Содержание

1. Вступительное слово.

Пособие предназначено для детей 5-6 классов, интересующихся программированием.

Здравствуйте!

Сейчас почти в каждой семье есть компьютер. И у многих из вас есть интерес к компьютерным играм. Не будем заострять внимание на вопросе о том плохо это или хорошо. Примем этот факт как данность. Но знаете ли вы, как создаются эти игры? Любая игра – это, прежде всего, программа. А для создания даже небольшой программы автору необходимо обладать хорошим багажом знаний, владеть определёнными навыками, уметь реализовывать идею с помощью специфического языка.

В любой профессии есть ремесло, а есть искусство. Так вот, настоящее программирование – это искусство. Конечно, те игры, в которые мы играем, создаются большими коллективами. Там есть и дизайнеры, и художники, и пиарщики, но основным ядром являются, конечно же, программисты. Почему речь зашла об играх? Дело в том, что решив много задач, выучив много команд и различных конструкций, написав множество алгоритмов, ваших знаний хватит на то, чтобы делать свои собственные, пусть небольшие, игры.

Вам предлагается лишь немного прикоснуться к этому миру и, как знать, возможно, в будущем вы станете выдающимися специалистами в этой области.

Изучать основы программирования мы будем в специальной учебной среде «Логомиры», основным персонажем которой является «Черепашка» – универсальный исполнитель алгоритмов.

Минимально необходимый для хорошей оценки уровень – это способность воспроизвести любую из изученных базовых программ. На отлично нужно знать и уметь воспроизводить алгоритмы и программы повышенной сложности. Те же, кто после изучения материала сможет не просто воспроизводить, но и создавать свои собственные сложные программы, конечно, будут поощряться дополнительно.

Желаю успеха!

2. Структура среды: меню, панель инструментов, рабочее поле, поле ввода команд.

Начнём с изучения структуры самой среды. Запустив программу Логомиры 3.0, мы увидим следующую картину:

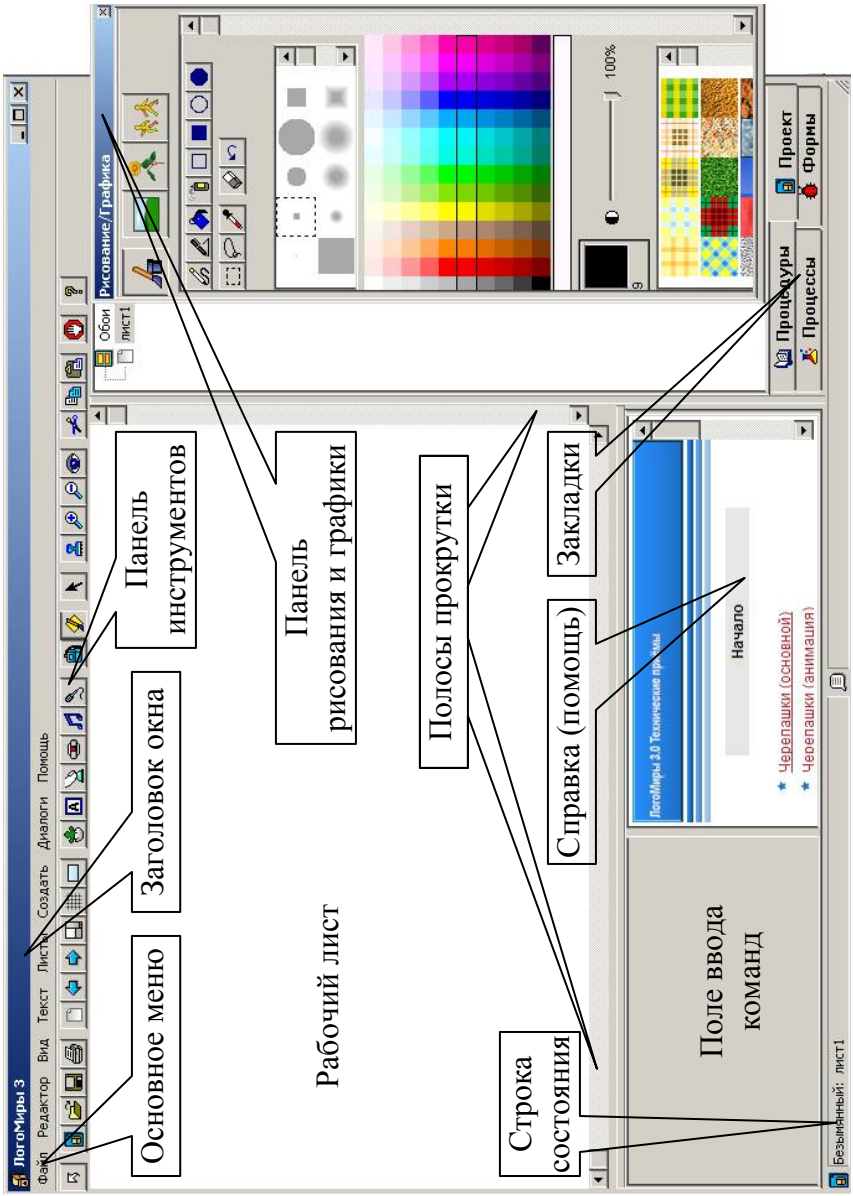


Рис. 1

Под заголовком окна находится строка меню. Чуть ниже располагается панель инструментов.

Как и в большинстве программ одни и те же действия можно сделать или с помощью кнопок на панели инструментов или из меню. Рассмотрим основные пункты меню, которыми воспользуемся в самое ближайшее время.

3. Меню, панель инструментов.

3.1. Пункт меню «Файл».

1) *Файл → Новый проект*.

Очевидно, что, вызвав этот пункт, откроется новый проект. При этом если вы вносили хотя бы какие-то изменения в текущий проект, то программа спросит, хотите ли вы сохранить текущие изменения или нет. На панели инструментов этому пункту меню соответствует кнопка  .

2) *Файл → Новый размер проекта*.

Размер проекта можно изменить только в самом начале. Если в проект внести изменения, то его размер изменить уже нельзя.

По умолчанию стоит стандартный размер. Чтобы избежать возможных недоразумений, при создании нового проекта следите за тем, чтобы его размер был стандартным.

3) *Файл → Открой проект*.

Если у вас уже есть сохранённый ранее проект, вы можете открыть его с помощью данного пункта меню. Например, у вас есть проект, сохранённый под именем *машинки.mwx* на столе. Выбираем пункт «Открой проект» и далее, как показано на рисунке 2, указываем путь к файлу с сохранённым проектом.

На панели инструментов этому пункту меню соответствует кнопка  .

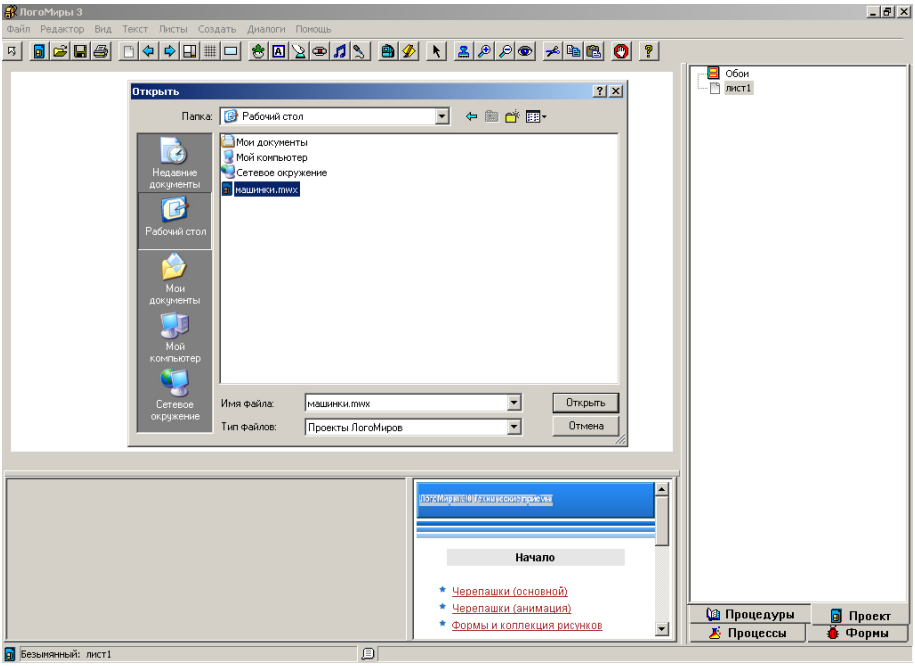


Рис. 2

4) *Файл → Сохрани проект*.

Чтобы сохранить созданный проект, нужно воспользоваться этим пунктом меню.

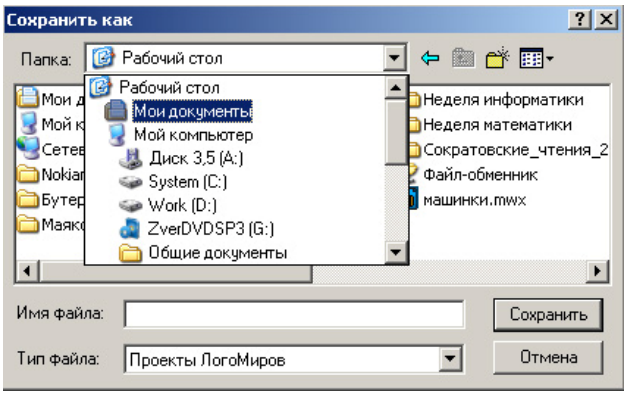


Рис. 3

После чего надо указать место сохранения: диск:\папка\имя_файла, где вы хотите сохранить проект, и написать имя файла. Если вы уже однажды сохранили свой

проект и выбрали этот пункт меню снова, то все изменения, внесённые в проект, сохранятся в том же месте в файле под тем же именем.

На панели инструментов этому пункту меню соответствует кнопка  .

5) *Файл → Сохрани проект под именем...*

Если нужно сохранить файл под другим именем и/или в другом месте, то выбираем именно этот пункт меню. Если же проект сохраняется первый раз, то разницы между пунктом «*Сохрани проект»* и «*Сохрани проект под именем...*» никакой нет.

В пункте меню «Файл» ограничимся пока только этими подпунктами.

3.2 Пункт меню «Редактор».

В этом пункте меню находятся подпункты, связанные с редактированием текста и других объектов.

Несколько слов о «буфере обмена».

Буфер обмена *– область оперативной памяти компьютера, в которой могут сохраняться данные различных форматов для переноса или копирования их между приложениями или частями одного приложения.*

Для того чтобы разобраться, что же это такое, рассмотрим конкретный пример. Предположим, что у вас есть часть текста, которая с небольшими изменениями должна несколько раз повториться. Для этого совершенно не нужно набирать этот текст каждый раз заново. Достаточно выделить его с помощью мыши или сочетания клавиш SHIFT + [→] (зажимаем клавишу SHIFT и, не отпуская её, двигаем курсор с помощью клавиш управления курсором, то есть стрелочками), предварительно установив мигающий курсор в нужное место. После этого надо выбрать в меню *Редактор → Копируй* (выделенный текст скопируется в буфер обмена). Если выбрать *Редактор → Вырежи*, то выделенный фрагмент исчезнет с экрана и переместится в буфер обмена. Затем вы ставите курсор в то место, куда нужно добавить требуемую часть текста и выбираете в меню *Редактор → Верни*.

Часть текста, которую вы выделяли, появится в нужном месте. При этом в промежутке между копированием и вставкой текст находился как раз в буфере обмена. Помимо этого на панели инструментов есть группа кнопок:

 – вырезать;  – копировать;  – вставить.

Так что, кому удобно, могут вместо пунктов меню использовать соответствующие кнопки.

3.2.1 «Горячие» клавиши.

Конечно, вы понимаете, что клавиши на самом деле никак не могут оказаться таким уж сильно нагретыми. Горячими клавишами называются либо отдельные клавиши, либо сочетания клавиш, с помощью которых можно быстро проделать те или иные операции. В описанном выше примере вместо пунктов меню *Редактор → Копируй* и *Редактор → Верни* можно воспользоваться сочетанием клавиш CTRL (ЯБЛ) + C и CTRL (ЯБЛ) + V соответственно. То есть, выделив нужный фрагмент текста или элемент рисунка, вы нажимаете CTRL (ЯБЛ) и, не отпуская эту кнопку, нажимаете один раз C для копирования и V для вставки фрагмента из буфера обмена.

Раскрыв меню, рядом с каждым подпунктом пункта меню «Редактор» вы увидите соответствующие сочетания горячих клавиш:

- **CTRL (ЯБЛ) + C** – скопировать выделенный фрагмент в буфер обмена;
- **CTRL (ЯБЛ) + X** – вырезать выделенный фрагмент в буфер обмена;
- **CTRL (ЯБЛ) + V** – вставить выделенный фрагмент из буфера обмена;
- **CTRL (ЯБЛ) + A** – выделить всё;
- **CTRL (ЯБЛ) + Z** – отменить последнее действие;
- **CTRL (ЯБЛ) + S** – сохранить проект;
- **Delete (Del)** – удаление одного символа после курсора (только для версии Логомиров под Windows);
- **BackSpace** (⟸) – удаление одного символа до курсора;

Использование горячих клавиш значительно облегчает набор текста, управление приложением и системой в целом. Поэтому настоятельно рекомендуется твёрдо заучить, по крайней мере, приведённые сочетания и постоянно тренироваться в их применении.

Замечания

1. Если вы работаете за обычным компьютером (IBM-совместимом), то вы используете в сочетаниях кнопку CTRL. Их на клавиатуре две. Они находятся в самом нижнем ряду клавиш ввода текста по краям. Если же вы занимаетесь на компьютере Macintosh (iBook G4 в нашем случае), то вам необходимо в сочетаниях клавиш использовать кнопку ЯБЛ (это кнопки слева и справа от пробела, рис. 4).

Рис. 4

2. На клавиатуре iBook отсутствует кнопка Delete (Del). Поэтому для удаления отдельных символов надо использовать кнопку ⇦.

3.3 Пункт меню «Вид».

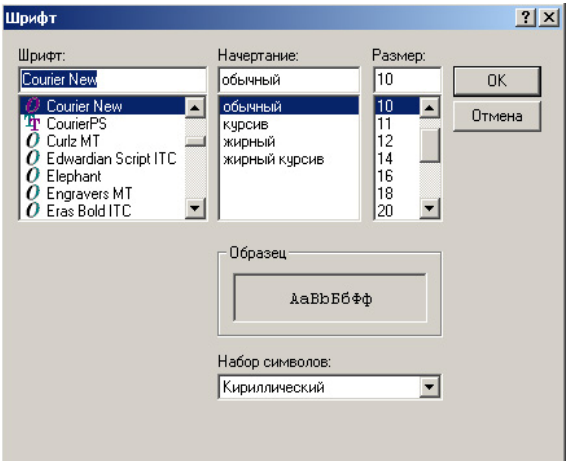
В этом пункте меню можно включить или выключить те или иные элементы среды, а также восстановить расположение элементов по умолчанию. Этому пункту меню на панели инструментов соответствует кнопка  .

Особенно часто нам понадобится включать и отключать панель «Рисование/Графика». Этому подпункту соответствует кнопка  на панели инструментов.

Попробуйте самостоятельно включить или наоборот выключить какой-либо элемент среды.

3.4 Пункт меню «Текст».

Не трудно догадаться, что здесь находятся элементы управления текстом. Перед тем как менять настройки текста, необходимо его или выделить, или сделать настройки до начала ввода.



Шрифт…

Здесь меняются тип шрифта, его начертание и размер.

Рис. 5

Цвет…

Изменение цвета.

Рис. 6

Замечание

Большинство из вас дома будут пользоваться версией Логомиров для Windows. Принося проекты, сохранённые дома и открывая их в версии Логомиров для Mac OS X на ноутбуках, вас неприятно удивит тот факт, что написанная программа будет выглядеть как набор квадратиков. Не нужно впадать в панику! Достаточно выделить весь текст программы и дальше с помощью пункта меню *Текст → Шрифт…* (рис. 5) изменить тип шрифта на *Courier New → Regular*. И вы снова увидите свой труд в нормальном виде. Также можно для удобства увеличить размер шрифта до 14.

3.5 Пункт меню «Листы».

В проекте может быть несколько листов. В этом пункте меню находятся подпункты, связанные с управлением листами. *Листы → Новый лист* создаёт новый лист. То же самое произойдёт и при нажатии на кнопку  на панели инструментов. Листы можно переименовывать: *Лист → Назови лист…* Переключаться между листами можно тремя разными способами:

1) с помощью кнопок на панели инструментов:  ;
2) с помощью пункта меню *Листы → Имя листа*;
3) на дополнительном поле в закладке «Проект» двойным щелчком левой кнопки мыши на имени листа.

Значительная часть проектов содержит сразу несколько задач. Поэтому целесообразно каждую маленькую задачу создавать на отдельном листе.

В этом же пункте меню содержится ещё один подпункт: *Эффекты…* Разберёмся зачем он нужен. В среде Логомиры предусмотрена возможность создания презентаций, где каждый лист – это один слайд. Так вот, для того чтобы презентация выглядела более привлекательно, можно настроить способ смены слайдов как раз с помощью этого пункта меню.

3.6 Пункт меню «Создать».

В Логомирах можно создавать самые разнообразные объекты. Начнём с самого главного – того, с которым вам постоянно придётся иметь дело, с «Черепашки». При выборе в меню *Создать → Черепашку* в середине поля появится невзрачная чёрная черепашка с лапками и головой. Ниже мы с вами подробно поговорим о том, что можно с ней делать. Сейчас же рассмотрим ещё один способ создания черепашки. На панели инструментов есть кнопка  . Чтобы создать черепашку таким способом, надо сначала нажать на эту кнопку, а затем щёлкнуть по рабочему полю.

Для работы нам понадобятся разные вещи, но в ближайшем будущем будут нужны ещё два объекта – это бегунки и кнопки. Как всегда можно создавать эти объекты с помощью соответствующих подпунктов пункта меню «Создать», а можно воспользоваться кнопками на панели инструментов:

 – создание кнопки;  – создание бегунка.

Об остальных подпунктах мы пока говорить не будем, поскольку для тех задач, которые будут предложены, они пока не потребуются.

Давайте закончим с меню.

3.7 Пункт меню «Помощь».

В версиях Логомиров под Window и Mac OS X помощь несколько отличается, однако, основная идея во всех версиях одна: описание работы с черепашкой и словарь так называемых примитивов (команд). Если вы подержите какое-то время мышку на примитиве в поле ввода команд или в закладке Программы, появится всплывающая справка с краткой информацией о примитиве. Но это относится только к Логомирам под Window.

Если вдруг вы что-то подзабыли или вам нужна информация о применении той или иной команды, то в этом пункте меню вы найдёте необходимые сведения.

Очень важное замечание!

В процессе работы чаще сохраняйте свой проект (для этого есть соответствующее сочетание горячих клавиш), так как бывают моменты, когда программа «виснет».

И запомните, что к учителю за справкой и советом следует обращаться в самую последнюю очередь! Учитесь действовать самостоятельно.

4.1 Основные команды черепашки.

Команда	Сокращ-е	Пример
вперед *число шагов*	***вп***	вперед 100 (или вп 100)
назад *число шагов*	***нд***	назад 200 (или нд 200)
направо *градусы*	***пр***	направо 30 (или пр 30)
налево *градусы*	***лв***	лв 310 (или пр 310)
по Опускает перо черепашки.	***по***	по
пп Поднимает перо черепашки.	***пп***	пп
домой Возвращает черепашку в середину экрана головой вверх.	***домой***	домой
сг Возвращает черепашку в середину экрана головой вверх и стирает графику.	***сг***	сг
нов_х Перемещает черепашку в точку с новой координатой по оси *х*.	***нов_х***	нов_х 100
нов_у Перемещает черепашку в точку с новой координатой по оси *у*.	***нов_у***	нов_у 100
нм [х у] Новое место. Перемещает черепашку в точку с заданными координатами.	***нм***	нм [50 80]
штамп Оставляет оттиск текущей формы черепашки на рабочем поле.	***штамп***	штамп
жди *число*	***жди***	жди 1
пч Показывает черепашку	***пч***	пч
сч Скрывает черепашку	***сч***	сч
останов Отключает все черепашки на листе проекта	***останов***	останов
повтори *число раз* [команды]	–	повтори 20 [вп 50 пр 18]

4.2 Способы ввода команд.

Начнём с ввода команд в поле команд (см. рис. 1). Для выполнения черепашкой набранной команды, надо нажать на клавишу Enter. Для того чтобы вставлять команды между уже введёнными, надо нажимать клавишу Enter, удерживая клавишу CTRL. Создадим черепашку (см. пункт 3.6) и попробуем сделать так, чтобы она нам что-нибудь нарисовала.

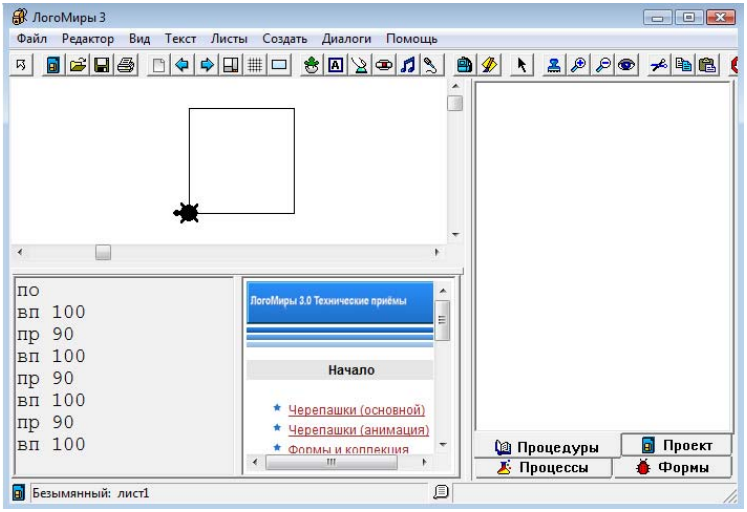


Рис. 7

Чтобы черепашка оставляла за собой линию на плоскости, нужно опустить перо. Для этого есть команда «*по*» (перо опусти). С помощью команд, задающих движение вперёд и команд поворота, рисуем квадрат (см. рис. 7).

Если потребовалось удалить всё, что было нарисовано на рабочем поле, нужно использовать команду «*сг*» (сотри графику).

сг
по
вп 100
пр 90
вп 100
пр 90
вп 100
пр 90
вп 100

Задание

Напишите программу, рисующую домик, не отрывая пера от плоскости (рис. 8). Длина стороны 100, длина диагонали 142.

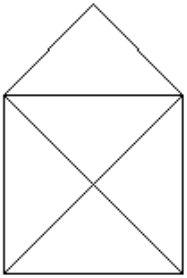


Рис. 8

Замечание

Тщательно следите за синтаксисом при вводе команд: между командой и числом обязательно должен быть пробел, если в команде есть символ нижнего подчёркивания, значит необходимо использовать именно его, а не пробел и т.п.

4.3. Процедуры. Собственные и несобственные процедуры.

Процедура – это подпрограмма, имеющая имя. Имя процедуры не должно содержать пробелы. Если вы хотите в имени процедуры использовать несколько слов, то используйте между ними либо символ нижнего подчёркивания «_», либо дефис «-». Называть процедуру следует так, чтобы по названию было понятно, что она делает. Внутри процедуры можно писать различные команды. Для вызова, находящихся внутри процедуры команд, нужно в тексте программы указать имя процедуры.

Синтаксис в Логомирах

Это ИМЯПРОЦЕДУРЫ
команды
Конец

Закрывать процедуру словом «конец» обязательно!

Пример:

это квадрат
по
нк 0
вп 100
пр 90
вп 100
пр 90
вп 100
пр 90
вп 100
конец

Процедура называется ***несобственной***, если она записана на закладке «Процедуры» в правой части экрана. Несобственные процедуры доступны для выполнения в любой момент любой черепашкой на любом листе в отличие от ***собственных*** процедур, записанных в рюкзаке конкретной черепашки.

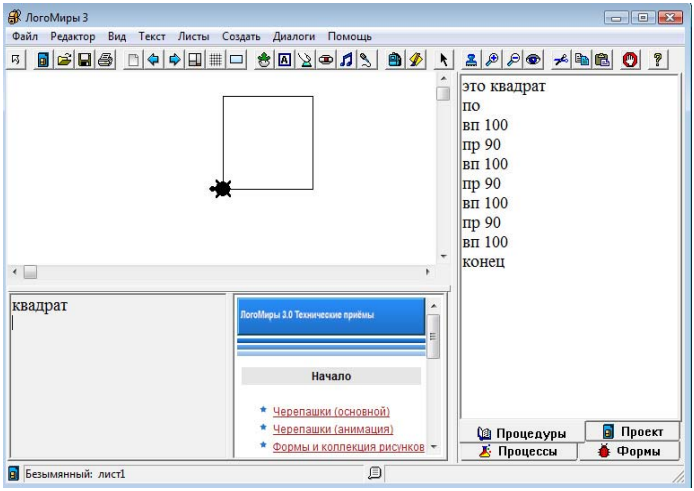


Рис. 9

Чтобы черепашка выполнила действия, находящиеся в несобственной процедуре, надо в поле команд написать имя процедуры и нажать Enter.

В основном мы будем использовать собственные процедуры, записывая их в рюкзак конкретной черепашки:

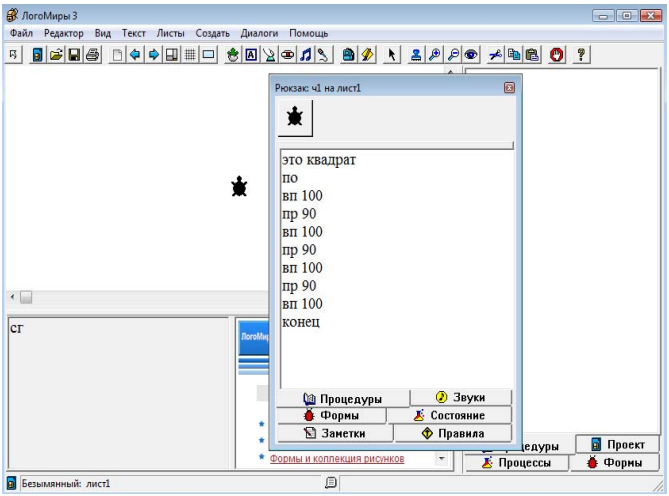


Рис. 10

Чтобы попасть в рюкзак черепашки, надо щёлкнуть правой кнопкой мыши по ней и выбрать в появившемся меню «Открыть рюкзак». Затем надо выбрать закладку «Процедуры».

Чтобы черепашка выполнила собственную процедуру, следует в закладке «Правила» поместить имя процедуры в самое верхнее поле, а затем оживить черепашку. Сделать это можно либо по щелчку левой кнопкой мыши на ней, либо из контекстного меню «Оживить.

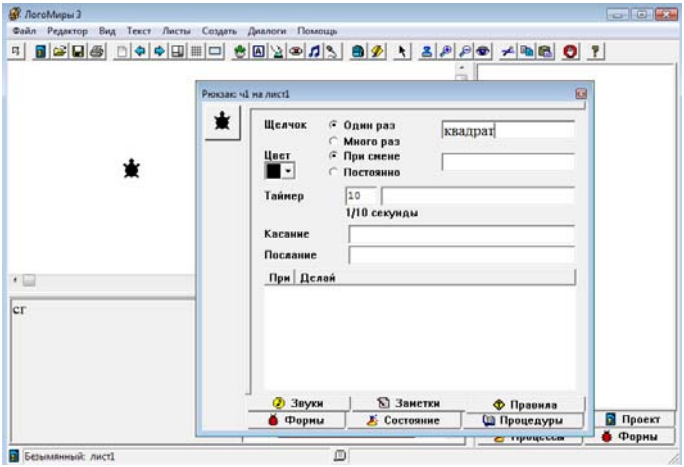


Рис. 11

Если вы допустите неточность в написании программы, хотя бы в одной команде, компьютер выдаст вам сообщение об ошибке.

Задание

Придумайте процедуру, рисующую карандаш, показанный на рисунке 12. Запишите её в рюкзаке черепашки. Поместите имя процедуры в правила. Оживите черепашку левой кнопкой мыши и посмотрите результат.

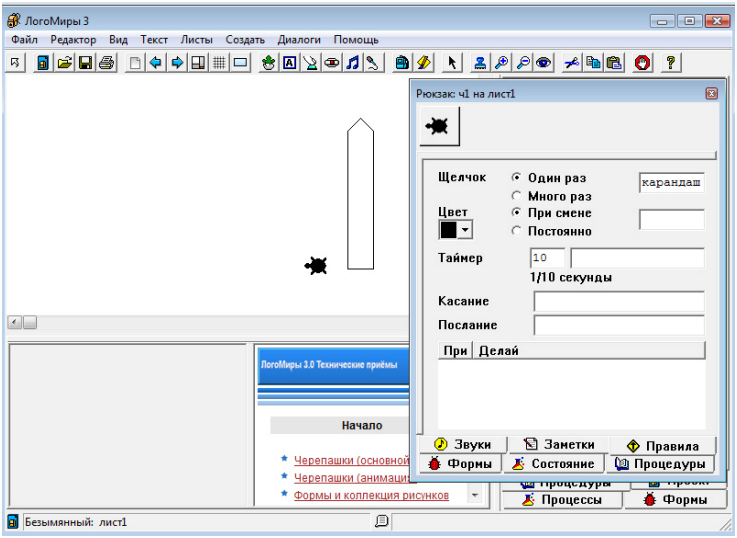


Рис. 12

5. Основы программирования. Алгоритм. Способы записи алгоритма. Виды алгоритмов. Задачи. Задачи повышенной сложности.

Алгоритм – это точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время.

5.1 Способы записи алгоритмов.

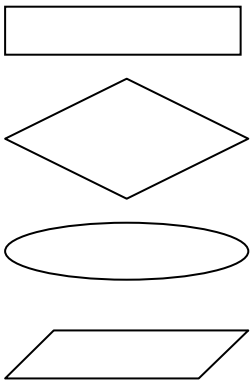
1. Словесный

Например, приготовление любого блюда в кулинарной книге записано именно этим способом.

2. Блок-схема

Это способ записи алгоритмов при помощи специальных символов.

Обозначения блок-схемы



– действия (команды);

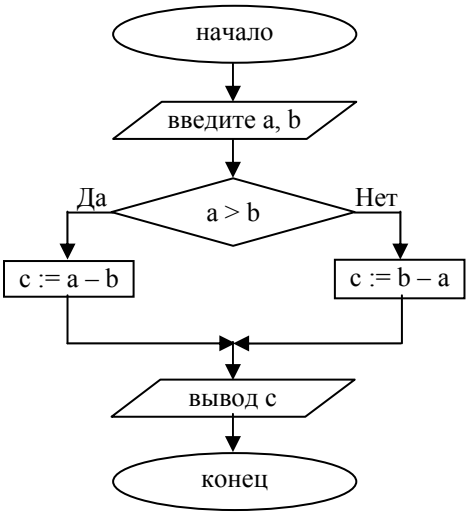
– условие;

– начало/конец;

– ввод/вывод данных.

На самом деле обозначений в блок-схемах гораздо больше. Но для наших целей достаточно этих четырёх.

Пример



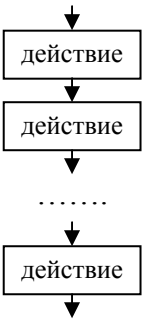
3. Программа, написанная на языке программирования

Существует множество самых разных языков программирования. Однако все их объединяют общие принципы. Отличаются они только синтаксисом, то есть названием тех или иных команд и способом их записи. Мы с вами будем учиться программировать в среде Логомиры, в которой есть свой специфический язык.

5.2 Виды алгоритмов.

1. Линейный

В линейном алгоритме действия (команды) следуют подряд друг за другом.

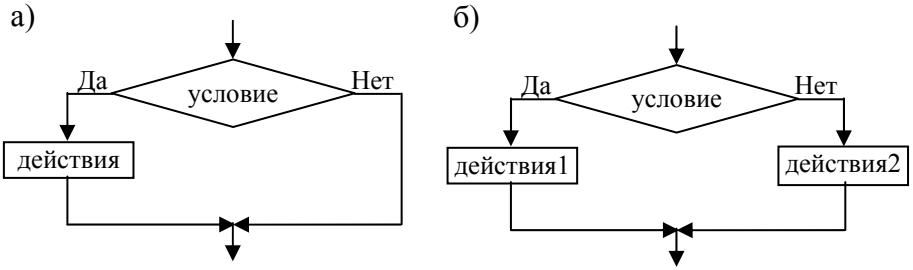


В качестве примера можно привести программы (процедуры) рисование квадрата (рис. 7), домика (рис. 8), карандаша (рис. 12).

2. Разветвляющийся

Это алгоритм, в котором есть условие.

Можно выделить два типа разветвляющихся алгоритмов:



В качестве примера приведём следующий.

Я лежу на диване. За окном идёт дождь.

а) *Если дождь прекратиться, то я пойду гулять.*

Здесь никаких действий в случае невыполнения условия не происходит!

б) *Если дождь прекратиться, то я пойду гулять, иначе – буду смотреть телевизор.*

В Логомирах есть две конструкции, соответствующие разветвляющемуся типу алгоритмов:

а) ***если условие [команды]***

б) ***если_иначе условие [команды1][команды2]***

В пункте б) обязательно использовать именно символ нижнего подчёркивания «_» между «если» и «иначе».

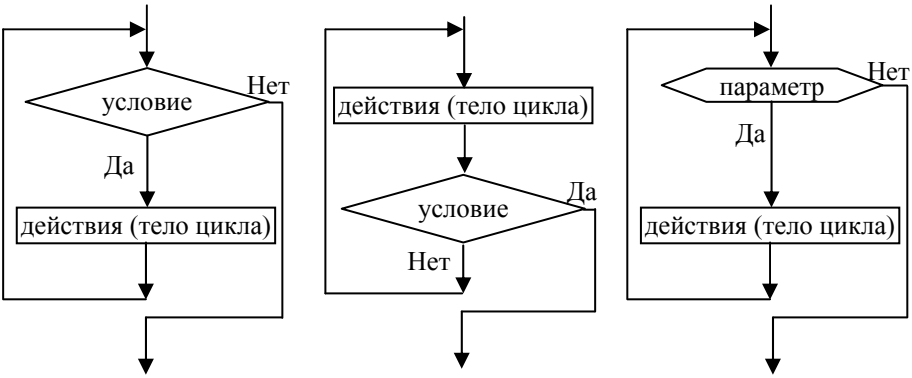
Примеры задач, на разветвляющийся вид алгоритмов мы рассмотрим позже.

3. Циклический

Это алгоритм, в котором есть повторяющиеся действия.

Во многих языках программирования есть три циклические конструкции (оператора):

а) цикл с предусловием б) цикл с постусловием в) цикл с параметром



Отметим различия между первыми двумя типами циклических алгоритмов:

1) У цикла а) условие находится в начале, у б) – в конце;
2) У цикла а) условие продолжения выполнения цикла должно быть истинно, у цикла б) – ложно;
3) Тело цикла а) может ни разу не выполниться (если условие сразу окажется ложным), тело цикла б) выполнится обязательно, по крайней мере, один раз;

Для наших задач пока достаточно использовать третий тип циклов. Он больше напоминает первую конструкцию, но с той разницей, что количество повторений будет жёстко фиксированным. Это фиксированное число повторений – ***параметр***.

Понять разницу между первым и третьим типом можно на таком примере:

Родители на даче дали вам такое задание – собрать упавшие с дерева яблоки.

Алгоритм а):

Пока *есть яблоки* подойди к яблоку, наклонись, подними, положи в мешок.

Алгоритм в):

Повтори *100 раз* подойди к яблоку, наклонись, подними, положи в мешок.

Очевидно, что при выполнении первого алгоритма вы будете собирать яблоки до тех пор, пока они есть. Во втором же случае в независимости от того какое имеется количество яблок вы выполните операции ровно 100 раз.

В Логомирах конструкции в) соответствует такая:

повтори <u>число раз</u> [команды]

Например, выполнив такую программу:

по повтори 4 [вп 100 пр 90],

черепашка нарисует квадрат.

Для того чтобы освоить циклический оператор (или оператор цикла) рассмотрим следующие задачи.

Введём три определения.

Ломаная – геометрическая фигура, состоящая из отрезков, последовательно соединенных своими концами, причём никакие два соседних отрезка не лежат на одной прямой.

Многоугольник – это замкнутая ломаная линия.

Многоугольник называется ***правильным***, если все его углы и стороны равны.

Мы уже знаем, как с помощью циклического оператора можно нарисовать квадрат. Квадрат – это правильный четырёхугольник. Давайте подумаем, как используя тот же подход, нарисовать правильные треугольник, 5-, 6-, 7-угольники.

Очевидно, что для рисования правильного треугольника нужно написать следующую процедуру:

```
это треугольник
   по
   повтори 3 [вп 100 пр 120]
конец
```

Здесь самая большая трудность заключается в расчёте угла поворота черепашки при каждом повторении. Откуда взялось число 120 в этой процедуре? Вспомним о том, что полный оборот окружности составляет 360º. Повернуть следует три раза. Вот и ответ: нужно поделить число 360 на 3.

Самостоятельно напишите процедуры для рисования остальных правильных многоугольников, показанных на рис. 13.

Итак, универсальный алгоритм для рисования любого правильного *n*-угольника (где *n* – число сторон или углов) выглядит так:

повтори n раз [вп 20 пр 360 / n]

Причём, чем больше число сторон, тем меньше следует брать длину стороны многоугольника.

5.3 Задачи.

Задача 5.3.1

Напишите алгоритм, с помощью которого черепашка нарисует окружность.

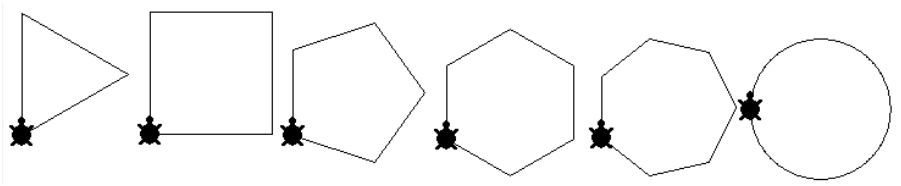


Рис. 13

Задача 5.3.2

Напишите алгоритм, с помощью которого черепашка нарисует 5-, 7-, 9- и 11-конечную звёзды, показанные на рис. 14.

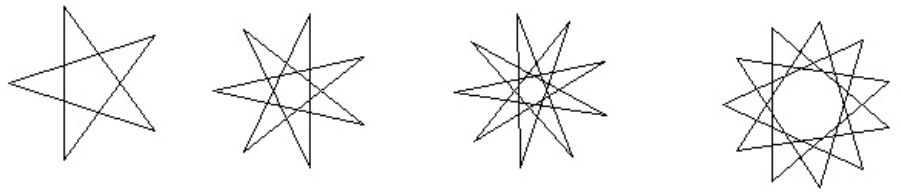


Рис. 14

Процедура для 7-конечной звезды:

```
это 7-звезда
   по
   повтори 7 [вп 100 пр 360 * 4 / 7]
конец
```

Используя оператор цикла, можно рисовать различные узоры.

Задача 5.3.3.

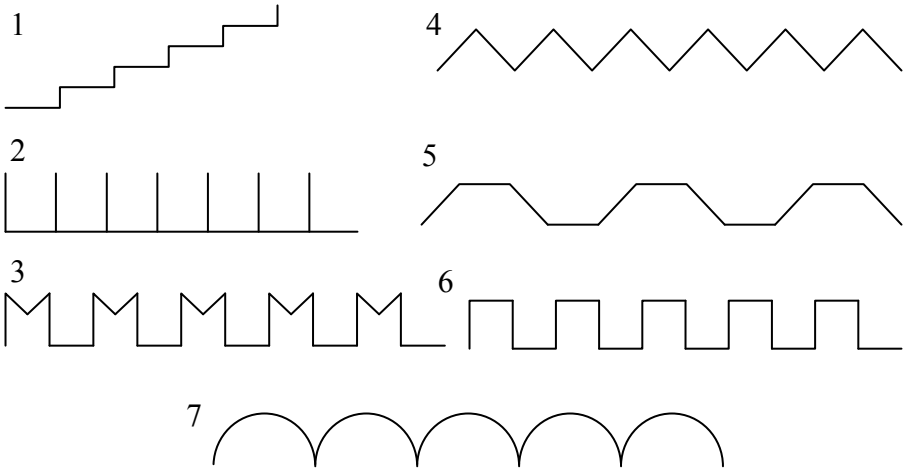


Рис. 15

Разберём подробно, как написать процедуру, рисующую узор под номером 3.

Наша задача нарисовать один элемент узора, а затем его повторить. Будем считать, что черепашка изначально находится головой вверх. Пусть длина вертикального отрезка и расстояние между элементами будет равна 50 точкам, а длина внутренних отрезков – 25 точек. Углы и направления поворота обозначены на рис. 16. Их расчёт не представляет существенных трудностей даже для пятиклассника.

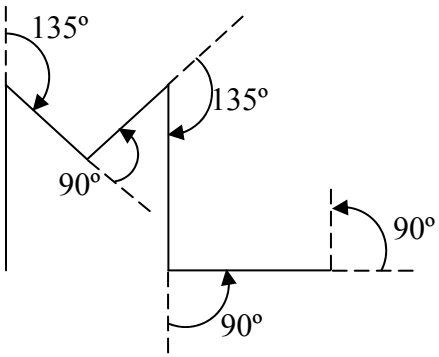


Рис. 16

Получим следующую процедуру.

это кремлёвская_стена
по
повтори 5 [вп 50 пр 135 вп 25 лв 90 вп 25 пр 135 вп 50 лв 90 вп 20 лв 90]
конец

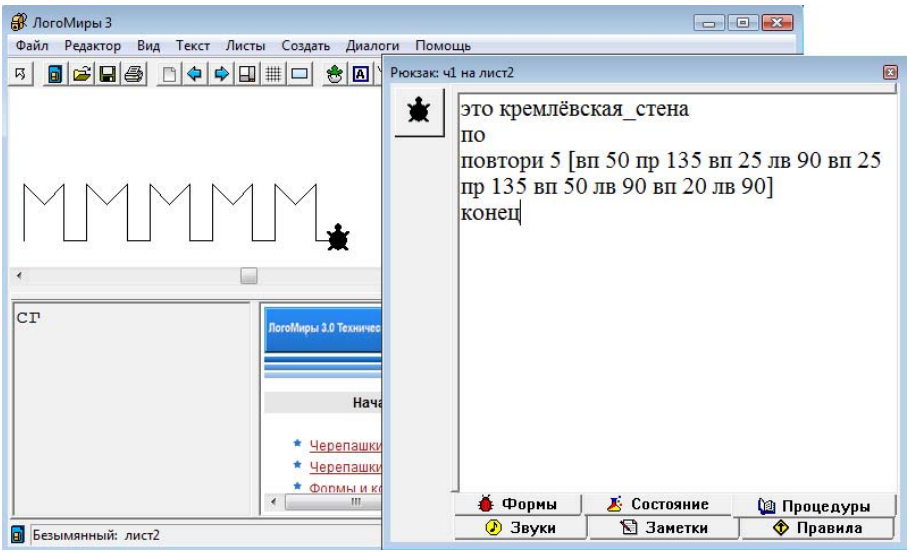


Рис. 17

Задача 5.3.3

Напишите алгоритмы, с помощью которых черепашка нарисует все оставшиеся на рис. 15 узоры. Создайте в одном проекте 7 листов (см. пункт 3.5), на каждом из которых разместите по одной черепашке с соответствующей программой.

В 7 узоре надо создать вложенный цикл: внутри будет цикл, рисующий одну полуокружность, а внешний цикл будет повторять эти полуокружности. Подумайте, как это сделать.

Задача 5.3.4

Напишите процедуры, рисующие снежинки, изображённые на рис. 18.

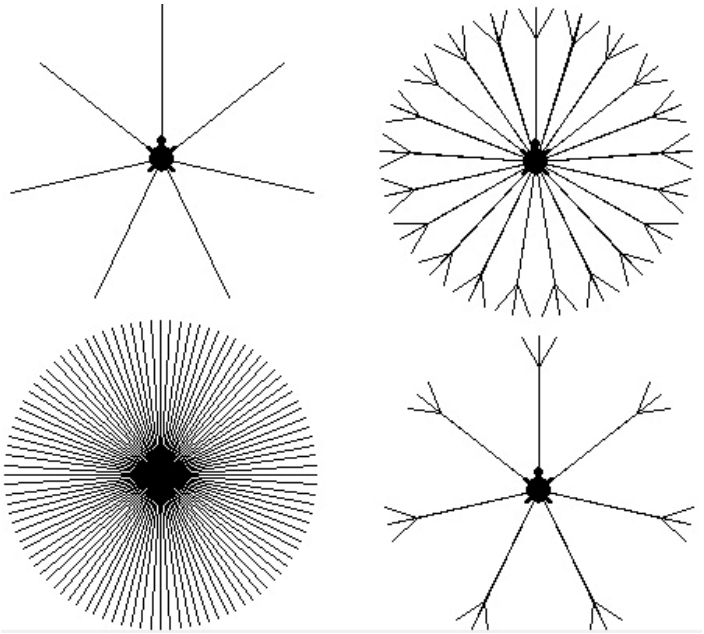


Рис. 18

Процедура, рисующая последнюю снежинку:

```
это снежинка4
    по
    повтори 7 [вп 100 нд 20 пр 30 вп 22 нд 22 лв 60 вп 22 нд 22 пр 30 нд
              80 пр 360 / 7]
конец
```

Задача 5.3.5

Придумайте свой или возьмите готовый узор или фигуру с повторяющимися элементами и напишите соответствующую процедуру, с помощью которой черепашка его нарисует.

Можно взять, например, классический греческий орнамент – меандр.

Рис. 19

О цвете и размере пера черепашки речь пойдёт чуть дальше.

5.4 Задачи повышенной сложности.

Все предложенные ниже задания выполняются по одной схеме: сначала пишется процедура, рисующая один элемент, затем во второй процедуре вызывается первая так, чтобы получился заданный узор.

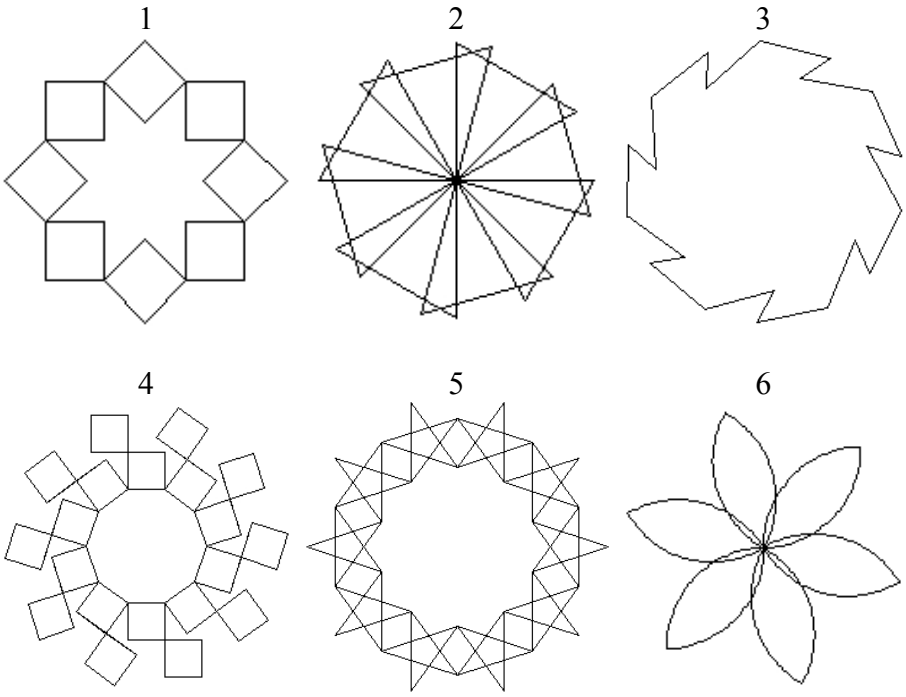


Рис. 20

Разберём подробно четвёртый рисунок.

Как видно, весь узор состоит из квадратов. Поэтому, используя предложенную выше схему, сначала создадим известную процедуру, рисующую один квадрат:

```
это квадрат
    повтори 4 [вп 30 пр 90]
конец
```

Теперь следует написать вторую процедуру, где в нужных местах будет вызываться первая:

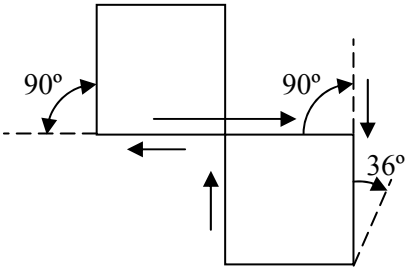


Рис. 21

Сначала рисуем нижний квадрат. Затем пройдём по уже нарисованной стороне квадрата, повернём налево и снова идём вперёд на длину стороны квадрата. Развернём черепашку направо на 90º и снова рисуем квадрат. Теперь осталось переместить черепашку в правый нижний угол нижнего квадрата и развернуть на угол 36º (см. рис 21). После чего все действия повторятся. В итоге получаем следующую процедуру:

```
это узор3
  по
  повтори 10 [квадрат вп 30 лв 90 вп 30 пр 90 квадрат лв 90
  нд 60 пр 90 нд 30 пр 36]
конец
```

Задача 5.4.1

Напишите процедуры, рисующие остальные узоры, изображённые на рисунке 20.

Задача 5.4.2

Придумайте свой сложный узор с повторяющимися элементами и напишите соответствующую процедуру.

6. Метод координат. Задачи.

В курсе математики 5 класса изучается координатная прямая, координаты и взаимное расположение точек на прямой. Теперь нам нужно перенести тот же принцип на плоскость. Добавим ещё одну координатную прямую. Проведём её перпендикулярно горизонтальной оси через точку 0 снизу вверх. Назовём горизонтальную прямую осью *x* (ось абсцисс), а вертикальную – осью *y* (ось ординат). Каждая точка плоскости задаётся двумя координатами: **первая координата указывается по оси *x*, а вторая – по оси *y***.

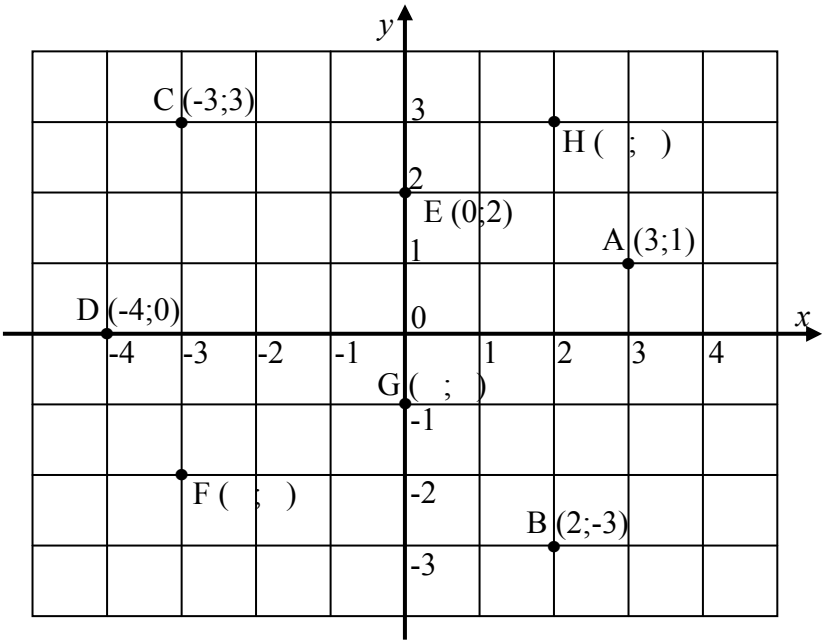


Рис. 22

Координаты точек: A (3;1), B (2;-3), C (-3;3), D (-4;0), E (0;-2).

Самостоятельно определите координаты остальных точек F, G, H. Не забывайте про порядок задания координат! Точки, лежащие на координатных осях, обязательно имеют одну из координат 0!

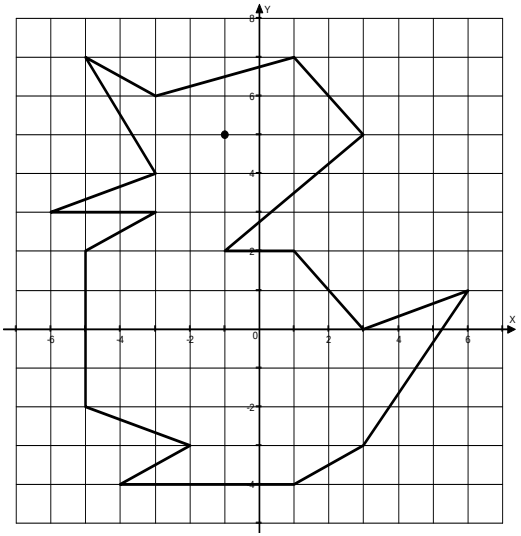
С помощью этого метода можно создавать различные рисунки, используя команду ***нм*** (новое место). Эта команда перемещает черепашку в точку с заданными координатами (см. таблицу команд пункта 4.1).

Формат команды: ***нм [x y]***.

В стандартном размере проекта значения координат по оси *x* от –372 до 372, по оси *y* от –213 до 213. Масштаб одной клетки при создании процедур для конкретных рисунков 10 точек.

Если координата имеет отрицательное значение, то между знаком «-» и числом пробел не ставится, так как это не знак арифметической операции, а знак числа!

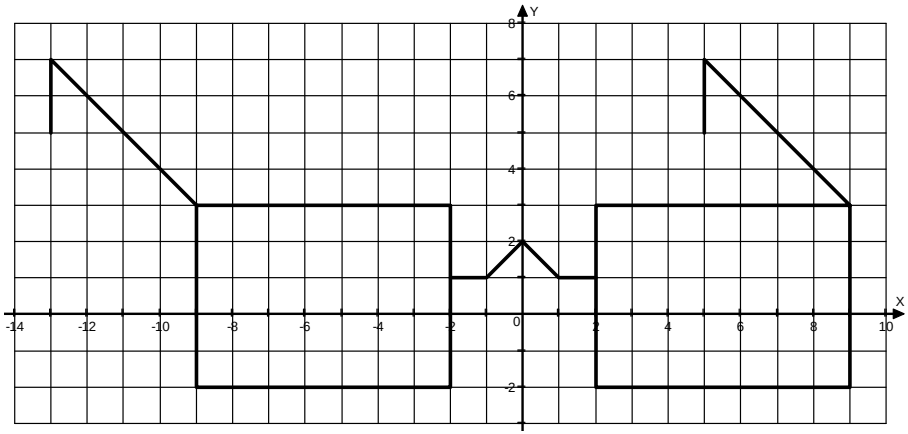
Пример процедуры, рисующей утёнка.



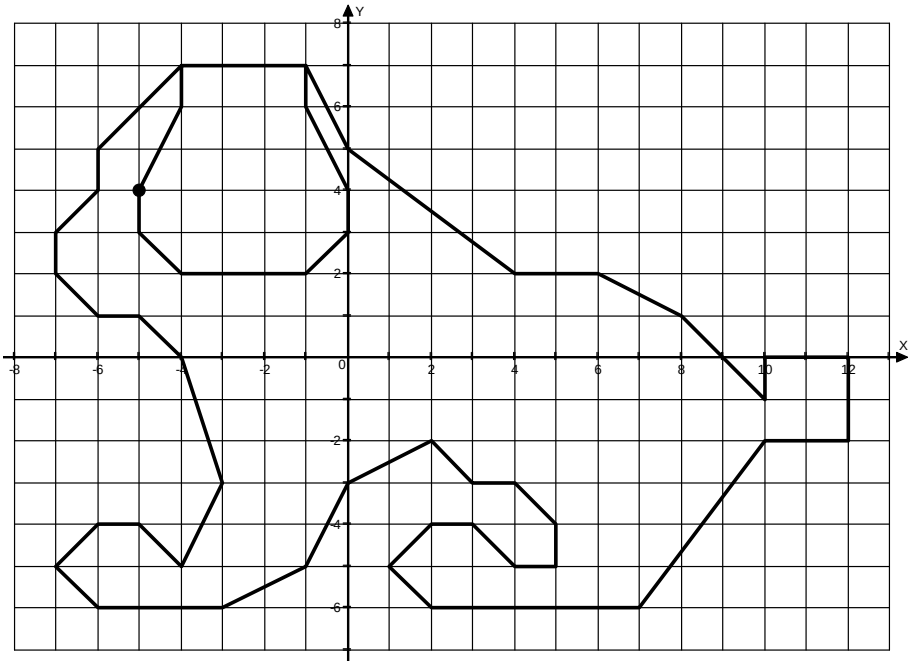

```
это утка        нм [-30 30]
пп              нм [-50 20]
нм [30 0]       нм [-50 -20]
по              нм [-20 -30]
нм [10 20]      нм [-40 -40]
нм [-10 20]     нм [10 -40]
нм [30 50]      нм [30 -30]
нм [10 80]      нм [60 10]
нм [-30 70]     нм [30 0]
нм [-50 80]     пп
нм [-30 40]     нм [-10 50]
нм [-60 30]     конец
```

Напишите процедуры, с помощью которых черепашка нарисует следующие картинки:

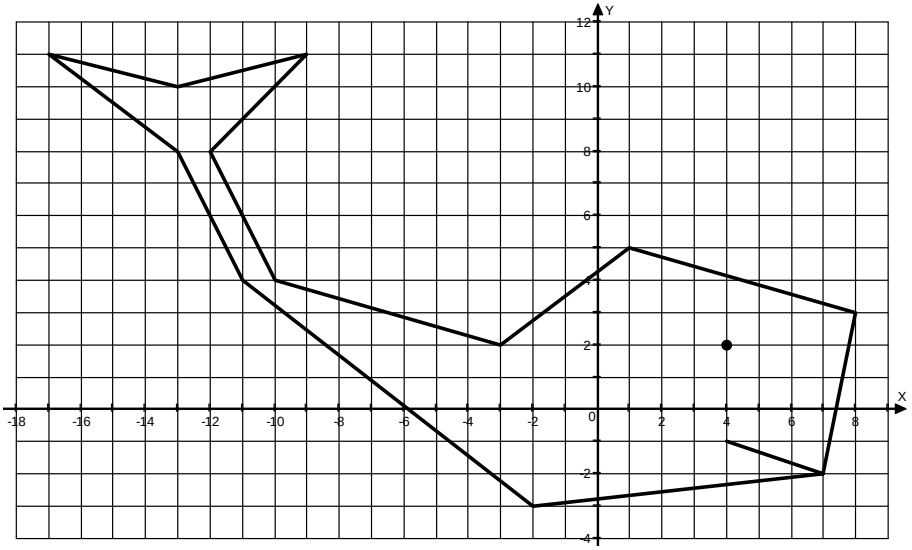
Задача 6.1



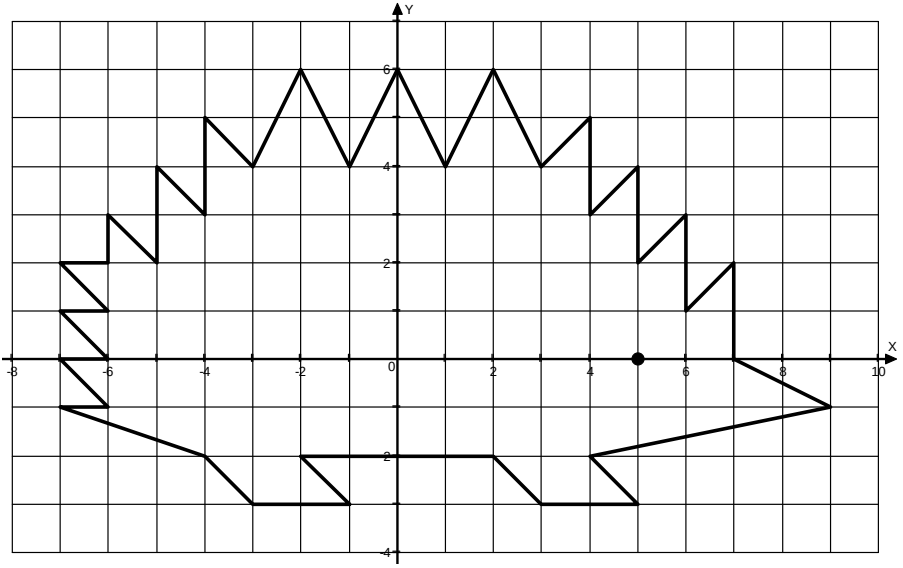
Задача 6.2



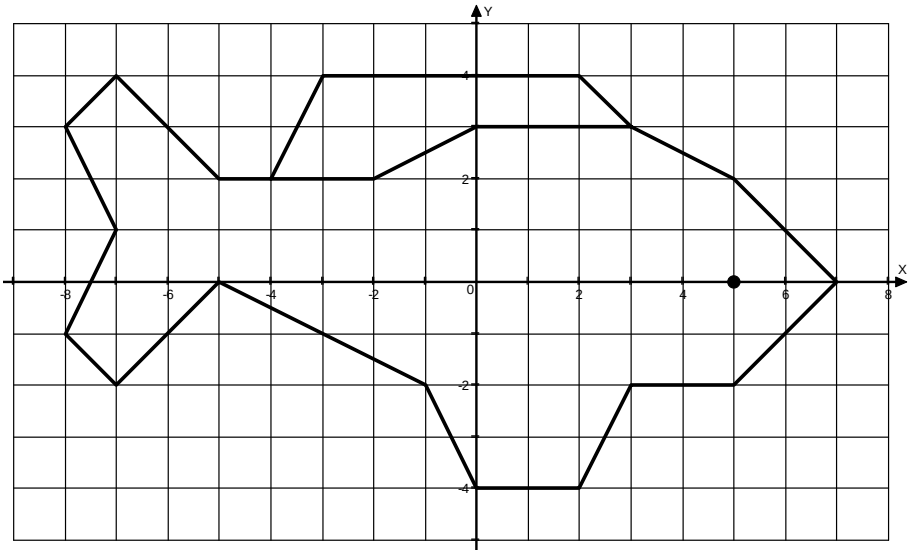
Задача 6.3



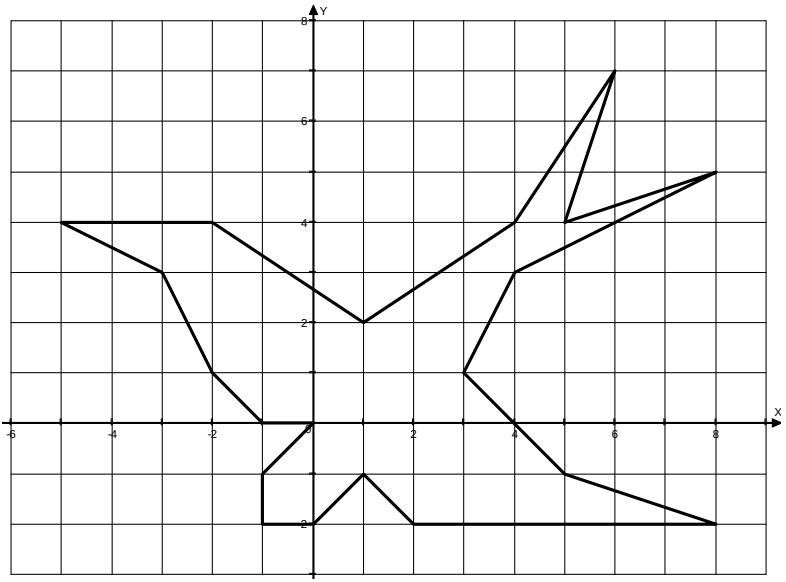
Задача 6.4



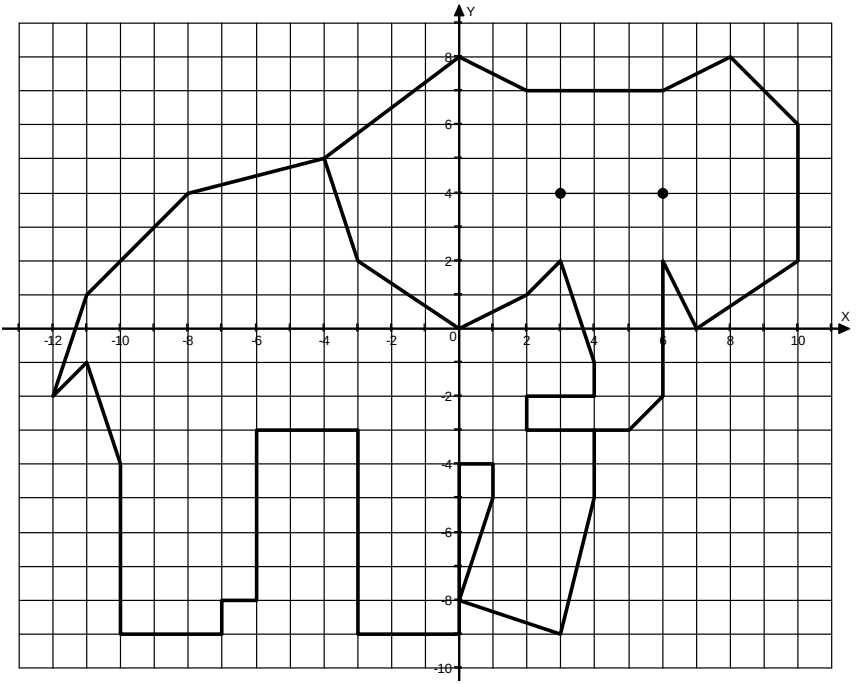
Задача 6.5



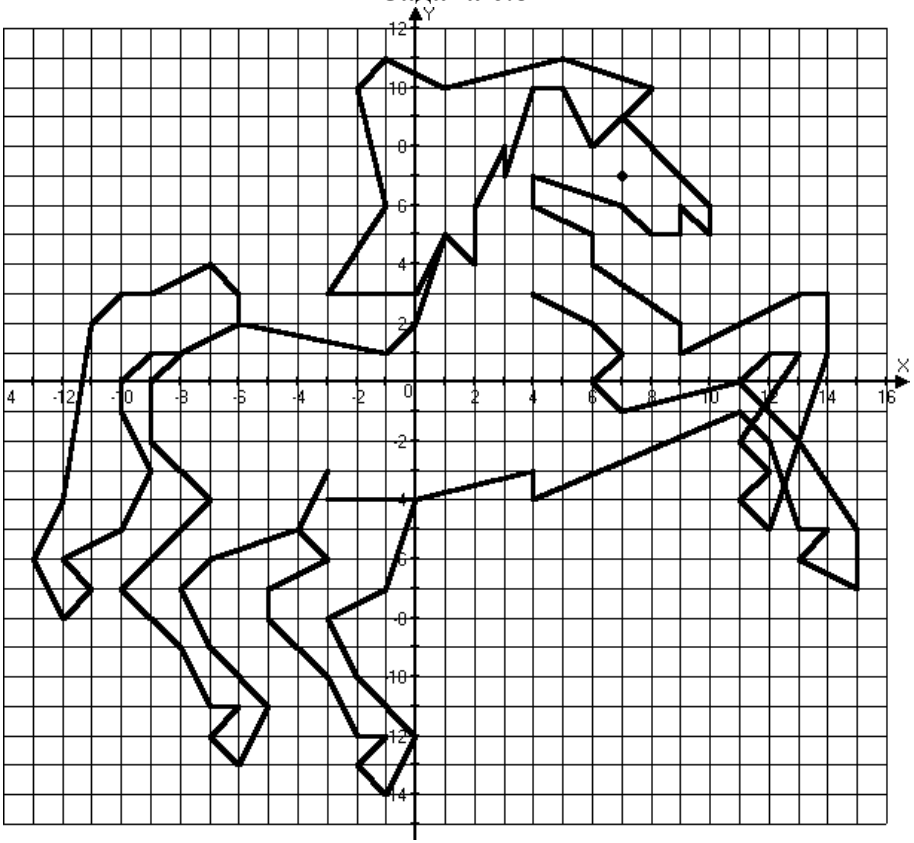
Задача 6.6



Задача 6.7



Задача 6.8



Задание

Придумайте свой рисунок и напишите соответствующую процедуру.

7. Черепашка. Формы черепашки. Собственные и несобственные формы. Анимация объектов.

Прежде чем говорить о формах черепашки, надо подробно разобрать панель «Рисование/Графика».

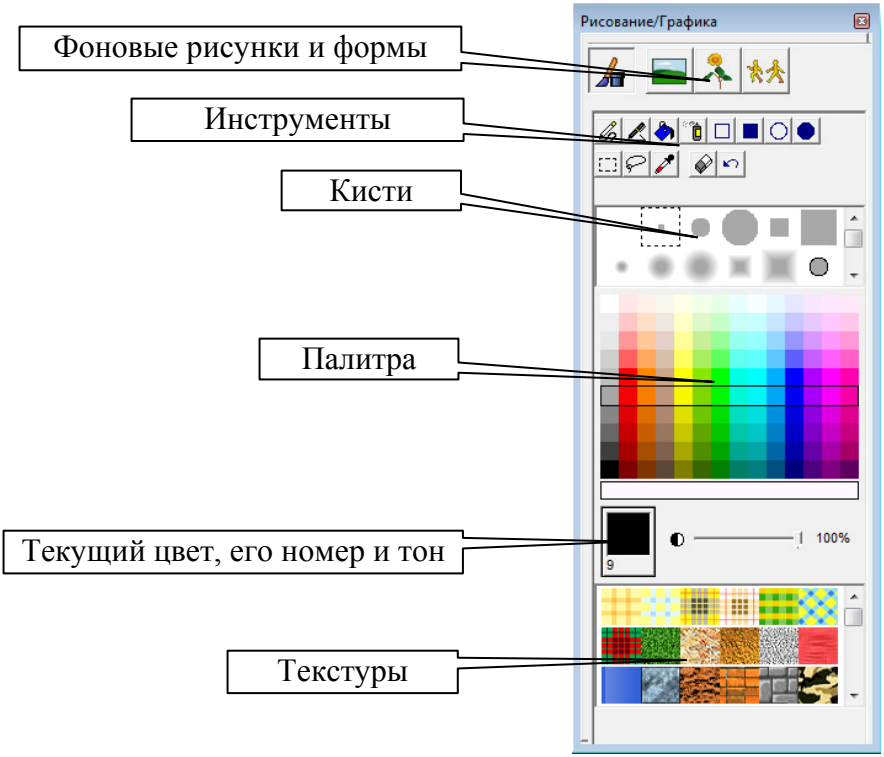


Рис. 23

В самом верхнем ряду находятся кнопки, с помощью которых можно выбирать режимы панели: рисование, фоновые рисунки, формы.

Ниже идут инструменты рисования, присущие любому растровому графическому редактору. Они вряд ли нуждаются в подробном описании. Если назначение того или иного инструмента не понятно, достаточно просто испробовать его на практике.

Далее мы можем увидеть различные виды кисти. Под ними находится палитра, где можно выбрать текущий цвет. Запомним, что у каждого цвета есть свой номер. В частности чёрный цвет имеет номер 9, белый – 0. Текущий цвет отображается под палитрой.

В нижней части панели «Рисование/Графика» располагаются различные текстуры, используя которые можно создать нужный вам фон.

Для того чтобы черепашка приняла ту или иную форму, сначала надо её выбрать, затем, захватив левой кнопкой мыши, переместить на центр черепашки:

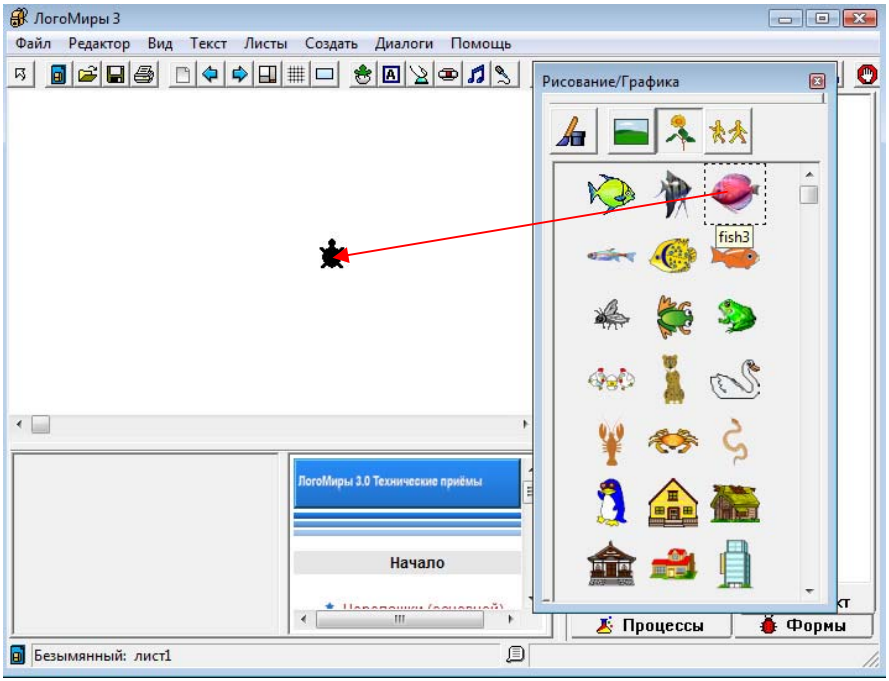


Рис. 24

После этой операции в рюкзаке у черепашки на закладке «Формы» на первой позиции окажется та форма, которую мы выбрали. Причём она будет яркой, такой же, как на панели «Рисование и графика». Это означает, что форма **собственная**, то есть она может быть использована только применительно к этой

черепашке. В правой части есть дополнительное поле, на котором тоже есть закладка «Формы». Если перетащить с панели «Рисования и графика» форму в эту область, а затем открыть рюкзак черепашки, то мы обнаружим, что на соответствующей позиции выбранная нами форма находится в виде силуэта, тени. Это означает, что форма **несобственная**, то есть доступна для использования любой черепашкой проекта.

Всего в рюкзаке черепашки может поместиться до 128 форм. Причём каждая позиция имеет свой номер. Чтобы переключить форму в тексте программы, есть специальная команда «*нф*» (новая форма).

нф номер формы

Пример: *нф 1*.

Если вы хотите вернуть форму черепашки по умолчанию, то надо задать номер 0 (*нф 0*).

Для того чтобы задать анимацию, то есть последовательную смену форм, необходимо на панели «Рисование и графика» выбрать пункт «Движение»:

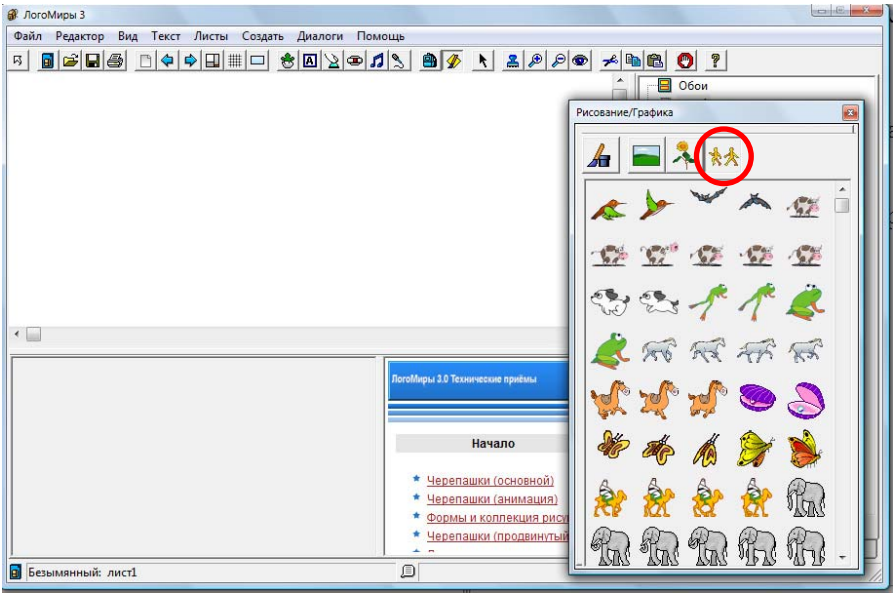


Рис. 25

Затем выбрать сразу несколько форм. Сделать это можно следующим образом: выбираем левой кнопкой мыши первую форму, нажимаем на кнопку SHIFT и, удерживая её, выбираем последнюю форму. После этого перетаскиваем все выбранные формы на черепашку. Открыв закладку «Формы» в рюкзаке, мы увидим, что все выбранные формы выстроились последовательно, начиная с первой. Теперь достаточно просто оживить черепашку. В результате она начнёт двигаться. При этом выбранные нами формы будут последовательно меняться, создавая тем самым мультипликационный эффект.

Когда мы оживили черепашку, в первом поле закладки «Правила» («по щелчку») автоматически появились команды |*вп 5 жди 1*| много раз, а в закладке «Состояние» в поле «Формы» появились имена выбранных нами форм. За счёт этого и создаётся эффект движения. Траекторию движения можно менять по своему усмотрению. Например, мы хотим, чтобы черепашка двигалась по кругу. Нет ничего проще – достаточно в закладке правила изменить команды таким образом: *вп 5 пр 2 жди 1*. Или если нам хочется, чтобы черепашка никуда не двигалась, но при этом меняла бы формы, нужно там же написать *вп 0 жди 1*.

Теперь мы без труда сможем создавать собственные анимированные картины. Осталось только подобрать соответствующий фон и узнать, каким образом его можно закрепить. Оказывается, это тоже очень просто! На панели «Рисование и графика» идём в пункт «Фоновые рисунки» и выбираем понравившийся нам фон. Перетаскиваем его на рабочее поле. Выделив рисунок левой кнопкой мыши, растягиваем его на всю рабочую область листа. Пока картинку по-прежнему можно двигать по экрану. При этом обратите внимание, что по периметру рисунка идёт пунктирная линия (рис. 26).

Чтобы «прилепить» рисунок к фону, нужно его отштамповать.

Сделать это можно двумя способами:

1. На панели инструментов есть инструмент «штамп». Берём его и щёлкаем по фоновому рисунку.
2. Вызвав щелчком правой кнопки мыши по рисунку контекстное меню, выбираем «Отштамповать».

Всё, рисунок окончательно приклеен к фону!

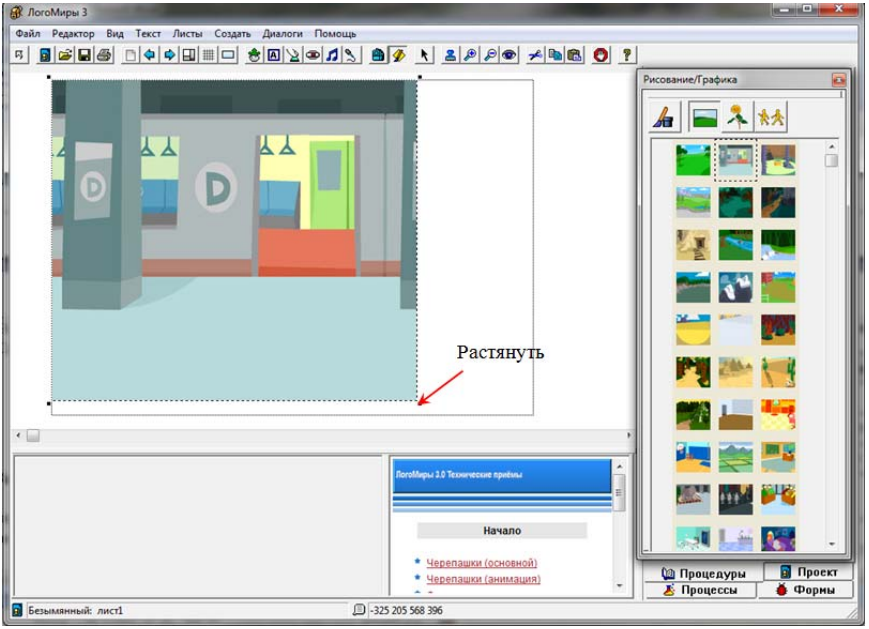


Рис. 26

Замечание

Если вы забудете отштамповать фоновый рисунок, то все черепашки, которые вы создадите, попадут под него. Их просто не будет видно!

Штамповать можно не только фоновые рисунки, но и, вообще говоря, любую форму из имеющегося набора. Кроме того, черепашка сама может оставлять след текущей формы по команде «*штамп*». Можно также штамповать формы черепашки пользуясь инструментом «штамп» вручную. Это особенно удобно, если нам понадобилось создать, например, группу деревьев или улицу из нескольких домов.

8. Команды черепашки, имеющие датчики. Задачи.

Команда (сокращение)	Датчик	Диапазон значений	Пример	Назначение
нов_форма (нф)	форма	0 – 128	нф 1	изменение формы
нов_курс (нк)	курс	0 – 359	нк 90	изменение курса
нов_размер (нрз)	размер	5 – 160	нрз 100	изменение размера
нов_размер_пера (нрп)	размер_пера	1 – 30	нрп 20	изменение размера пера
нов_цвет (нц)	цвет	0 – 139	нц 53	изменение цвета и цвета пера
нов_х	х_коор	-372 – 372 для стандартного размера проекта	нов_х 37	изменение положения по оси х
нов_у	у_коор	-213 – 213 для стандартного размера проекта	нов_у 79	изменение положения по оси у

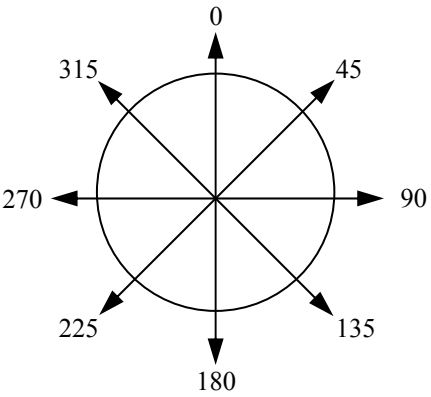


Рис. 27

Вторая команда «*нов_курс*» требует дополнительных пояснений. Мы уже знаем как поворачивать черепашку в нужную нам сторону с помощью команд «*направо*» (*пр*), «*налево*» (*лв*). Однако в этом случае поворот осуществляется относительно текущего направления. Это не всегда бывает удобно. По команде «*нк*» в каком бы положении ни находилась черепашка, она развернётся точно в заданном направлении. Так, например, по команде *нк* 197 черепашка развернётся следующим образом: . Всего направлений 360 (полный оборот окружности).

Рассмотрим задачи, где используются эти команды.

Задача 8.1

Напишите программу, при выполнении которой черепашка будет постепенно увеличиваться в размере (алгоритм роста репки).

Решение

Для того чтобы было интереснее, возьмём для черепашки любую форму, например, такую:  . В рюкзаке напишем следующую процедуру:

```
это репка
  нрз 5
  повтори 155 [нрз размер + 1]
конец
```

Не забудьте поставить имя процедуры на закладку правила (один раз)! В первой строке процедуры задаётся первоначальный размер черепашки, а именно, наименьший (см. диапазон значений в таблице). Затем при каждом шаге цикла размер черепашки будет увеличиваться на 1 до максимального размера 160.

Задание

Напишите аналогичный алгоритм для уменьшения размеров черепашки.

Запомним схему этого алгоритма:

команда → датчик → шаг.

Для закрепления сделаем ещё одну красивую задачу.

Задача 8.2

Напишите процедуру, рисующую цветовой круг, показанный на рис. 28. Постарайтесь сделать её самостоятельно.

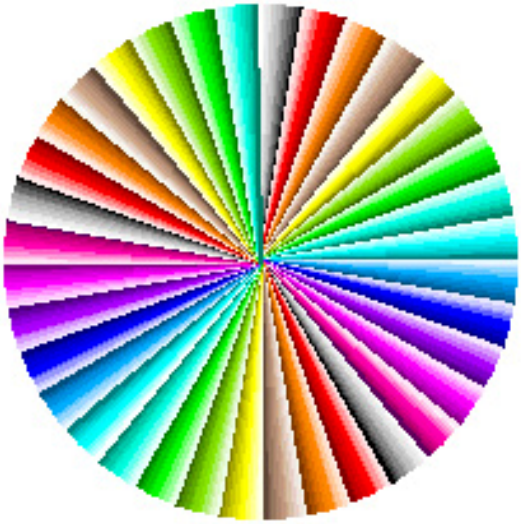


Рис. 28

Задача 8.3 (Весёлый маляр)

Создайте процедуру, при выполнении которой черепашка создаст следующий узор.

Рис. 29

Задача 8.4 (Автомобили на дороге)

Нарисуйте дорогу, небо и поля. На дороге поставьте две черепашки и создайте для каждой из них форму: одна черепашка – грузовик (вид сзади), вторая – легковая машина (вид спереди). Первая черепашка-грузовик должна двигаться вверх, отдаляясь уменьшаться в размере. Вторая черепашка-легковушка должна двигаться вниз, приближаясь увеличиваться в размере. Дополните проект готовыми формами домов и деревьев и другими объектами, которые вам подскажет фантазия. Создайте кнопки «Пуск» и «Стоп» для запуска и остановки черепашек.

На рис. 30 показан итог того, что должно получиться.

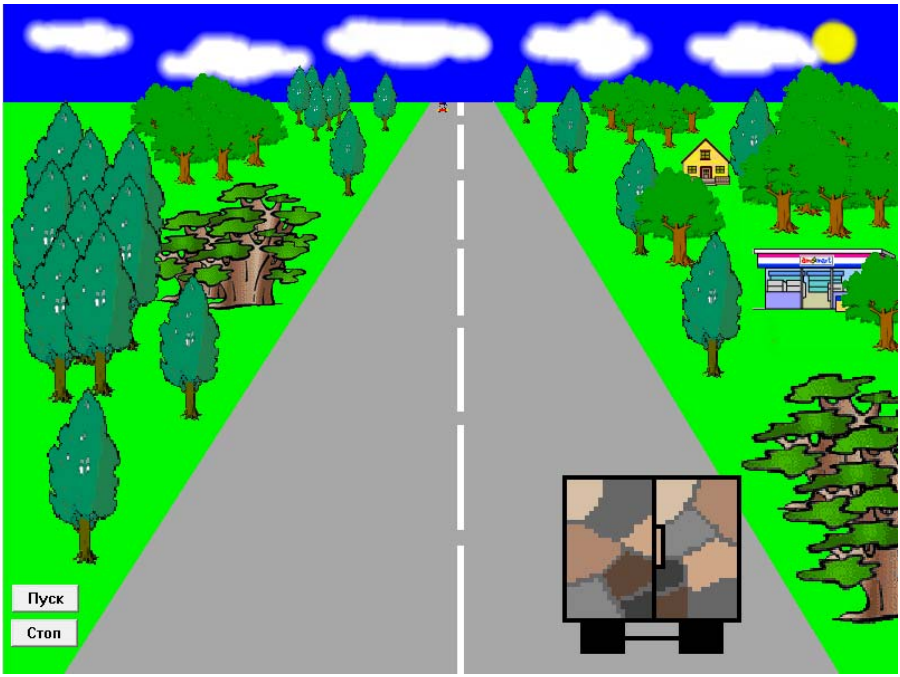


Рис. 30

Решение

Разберём пошагово все необходимые действия. В этом проекте почти всё придётся создать самим. Начнём с рисования неба. Выбираем на панели «Рисование/Графика» инструмент «сплошной прямоугольник» и цвет неба на ваше усмотрение,

например, 105. От верхнего левого или правого угла растягиваем прямоугольник примерно на 1/7 часть поля. Теперь выбираем цвет травы, например, 65 и с помощью инструмента «Заливка» закрашиваем всю оставшуюся часть поля зелёным. Снова выбираем «сплошной прямоугольник», цвет белый и размер кисти самый маленький. Начиная от границы неба, рисуем вниз разделительную полосу дороги посередине экрана. Дальше надо взять инструмент «Ручка» и цвет дороги (6 или 7). Теперь очень внимательно нужно от самой границы травы и неба, чуть отступив вправо и влево от сплошной разделительной полосы, нарисовать две симметричные линии до самого низа. Учтите, что если вы начнёте рисовать линии хотя бы на одну точку ниже границы с небом, то потом, заливая дорогу, вы зальёте лишнее. Так что даже лучше начать чуть-чуть выше уровня границы. Затем, конечно, надо залить обе части дороги тем же цветом, которым рисовали линии. В результате должна получиться следующая картина:

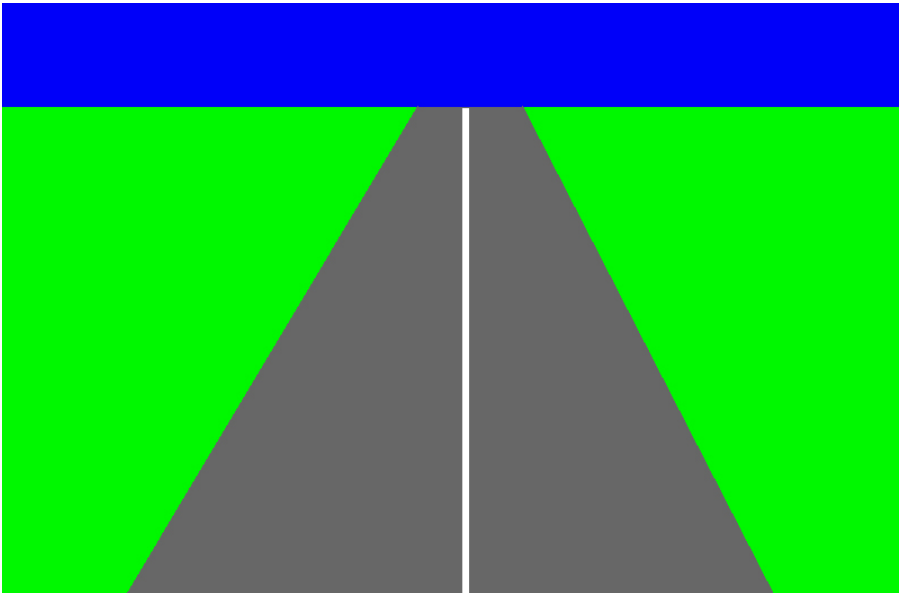


Рис. 31

Чтобы сделать разрывы в разделительной полосе, воспользуемся снова инструментом «Сплошной прямоугольник». Чтобы дорога выглядела более реалистично, во-первых, она должна довольно быстро сужаться к линии горизонта и, во-вторых, разрывы в разделительной полосе должны быть чем дальше, тем короче.

Теперь создадим черепашку и нарисуем форму грузовика. Для этого откроем её рюкзак и в закладке формы дважды щёлкнем по первой форме. Удалим маленький квадратик в середине с помощью инструмента «Прямоугольное выделение» и кнопки BackSpace или просто с помощью ластика. Возьмём инструмент «Прямоугольник» (незакрашенный), размер кисти 1 и, выбрав чёрный (9) цвет нарисуем на всю ширину, но оставив внизу примерно седьмую-восьмую часть области, прямоугольник – это кузов нашего грузовика. Разделим его с помощью инструмента «Ручка» пополам. Чтобы нарисовать колёса, воспользуемся инструментом «Сплошной прямоугольник». Между колёсами ручкой рисуем ось. Затем берём инструмент «Карандаш» и разбиваем внутреннюю часть прямоугольника на неравные области серым цветом, которые потом заливаем оттенками коричневых и серых цветов так, как будто это налипшая на заднюю часть грузовика грязь. В результате должна появиться такая форма:

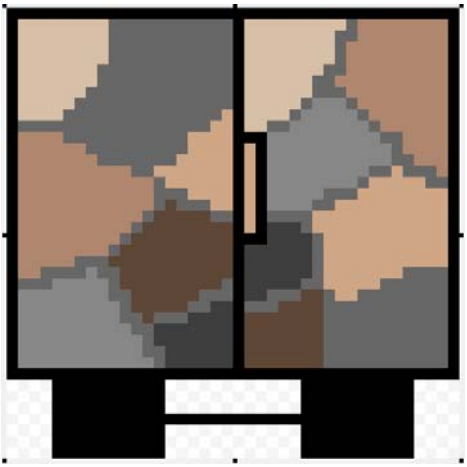


Рис. 32

Нажимаем на кнопку «ОК». Осталось написать программу. В поле ввода команд введите: *нф 1 нрз 160*. После чего поставьте черепашку в то место, откуда она начнёт движение. Переходим на закладку «Процедуры». Как почти в любой задаче программу можно написать по-разному. Рекомендуем, чтобы вспомнить, как использовать процедуру, написать их две: первая – начальное положение грузовика, вторая – процедура движения.

```
это грузовик_начало
    нк 340
    нф 1
    нрз 160
    нм [180 -202]
конец

это грузовик_движение
    грузовик_начало
    повтори 147 [вп 3 нрз размер - 1 жди 1]
    грузовик_начало
конец
```

В первой процедуре указываем все начальные параметры: форму, размер, местоположение, курс. Мы будем начинать движение грузовика с максимально возможного размера – 160 и с того места, куда мы его поместили. Координаты начального положения смотрим в закладке «Состояние». В зависимости от того, как вы нарисовали дорогу, курс черепашки может довольно существенно колебаться. Поэтому пока поставим примерно *нк 340*. Потом подберём его опытным путём. Обратите внимание, что первая процедура вызывается как до, так и после начала движения. Это нужно для того, чтобы по окончании движения грузовик возвращался на место. Количество повторений также может розниться в зависимости от размеров проекта и конфигурации дороги. Подберите нужное количество повторений самостоятельно. Следите за тем, чтобы ваш грузовик не уезжал ни на обочину, ни на полосу встречного движения, ни на небо. Не забудьте поместить имя второй процедуры в правила один раз.

Приведём ещё один вариант программы движения грузовика, записанной только одной процедурой. Форма и курс остаются неизменными, поэтому эти команды можно ввести

только один раз в поле ввода команд и в тексте программы их больше не указывать:

это грузовик
нрз 160
нм [180 -202]
повтори 147 [вп 3 нрз размер - 1 жди 1]
конец

Если мы будем повторять эту процедуру многократно, то наш грузовик будет постоянно возвращаться к началу и вновь двигаться по дороге к линии горизонта.

После того, как вы добьётесь, чтобы грузовик двигался чётко по дороге, можно переходить к созданию второго автомобиля – легкового. Так как структура программы для обеих машин одинакова, проще всего скопировать черепашку-грузовик и сделать её дубликат. После чего просто внести необходимые изменения в текст программы. Но до этого давайте создадим форму легковой машины.

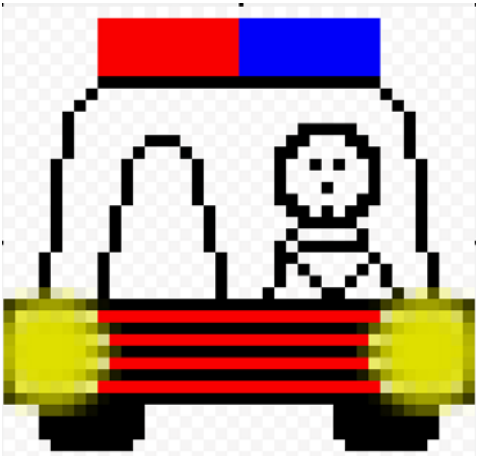


Рис. 33

Предварительно удалим у второй черепашки форму грузовика. Начнём рисовать машинку с незакрашенного прямоугольника. Затем нарисуем ручкой радиаторную решётку, боковые части и крышу. Закрашенным овалом рисуем колёса.

Фары рисуем карандашом с помощью размытия кисти 11. Можно дополнить рисунок мигалкой на крыше и человечком внутри.

После создания формы подправляем процедуры:

```
это легковая_начало
    нк 197
    нф 1
    нрз 10
    нм [-7 207]
конец

это легковая_движение
    легковая_начало
    повтори 147 [вп 3 нрз размер + 1 нф 2 жди 1 нф 1]
    легковая_начало
конец
```

Как и в случае с грузовиком программу можно существенно упростить. Попробуйте сделать это самостоятельно.

Добавим кнопки (смотри, как это сделать в п. 3.6): первая кнопка – начать движение («Пуск»), вторая – остановить движение. Инструкция для первой кнопки: *каждая [вкл]*, для второй – останов.

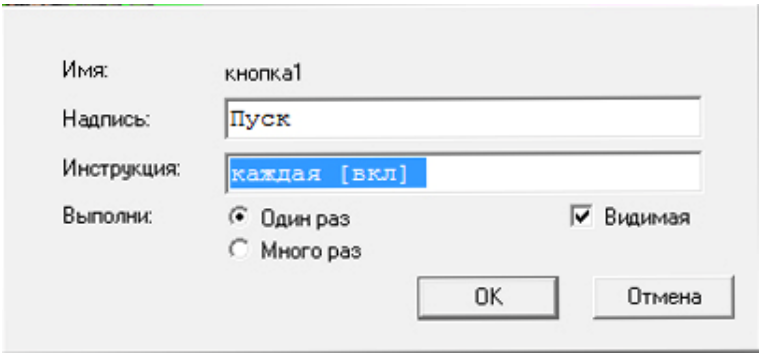


Рис. 34

9. Задачи, связанные с использованием условного оператора.

Условный оператор реализует разветвляющийся тип алгоритма (см. п. 5.2 Виды алгоритмов). Напомним, как производится проверка условий в среде «Логомиры»:

а) ***если условие [команды]***

б) ***если_иначе условие [команды1][команды2]***

Рассмотрим простейшие задачи, с использованием условия.

Задача 9.1

Разбейте всю рабочую область на 9 частей. Закрасьте все области разными цветами. Создайте черепашку и сделайте так, чтобы при её перетаскивании мышью с одного цвета на другой сама черепашка меняла бы цвет.

Решение

Для удобства разобьём всю плоскость на 9 частей, используя цвета, начиная с 5 и заканчивая 85 с шагом 10, то есть 5, 15, 25, …, 85 – это «чистые» цвета. Проще всего это сделать с помощью инструмента «сплошной прямоугольник».

Рис. 35

Напомним, что для изменения цвета черепашки есть команда нц (нов_цвет). Осталось сказать, с помощью какого условия черепашка может выяснить, какой цвет находится под ней. Это условие *цп = номер цвета*. Итак, процедура изменения цвета черепашки в зависимости от цвета поля под ней будет выглядеть следующим образом:

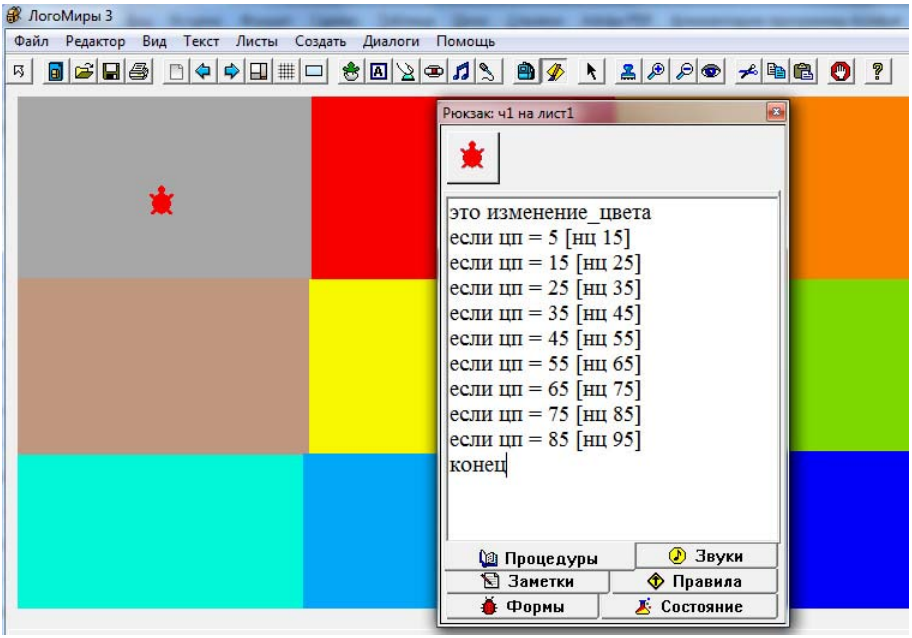


Рис. 36

```
это изменение_цвета
    если цп = 5 [нц 15]
    если цп = 15 [нц 25]
    если цп = 25 [нц 35]
    если цп = 35 [нц 45]
    если цп = 45 [нц 55]
    если цп = 55 [нц 65]
    если цп = 65 [нц 75]
    если цп = 75 [нц 85]
    если цп = 85 [нц 95]
конец
```

Не забудьте поместить имя процедуры «изменение_цвета» в первое поле закладки «Правила» и поставить «много раз».

Введём ещё одно полезное понятие. В качестве параметра для многих команд можно использовать ***случайный выбор*** числа из заданного диапазона, но этот выбор можно сделать только из целых положительных чисел, то есть из чисел 0, 1, …, *n*. Для

этого существует примитив «*случайный*» или в сокращённом варианте «*сл*». Например, для случайного выбора цвета из 50 цветов, следует написать *нц сл 50*. Решим ещё одну задачу.

Задача 9.2

Напишите программу, при выполнении которой черепашка начнёт движение со случайно выбранного места по оси *у* от 0 до 140 вверх до координаты *у*, равной 150.

Решение

```
это движение
    нов_х 0
    нов_у сл 140
    повтори 141 [вп 1 если у_коор > 140 [останов]]
конец
```

Можно перейти к более интересным задачам.

Задача 9.3 «Аквариум с рыбками»

Нарисуйте незакрашенный прямоугольник с толщиной границ равной 2 точки аквариум. Создайте две черепашки-рыбки. Напишите программу, при выполнении которой рыбки плавали бы в аквариуме по горизонтали на разных уровнях, отталкивались от стенок аквариума и меняли направление и форму. Добавьте в аквариум водоросли, ракушки, камни и раковины.

Решение

Проблем с рисованием незакрашенного прямоугольника быть не должно. Создаём черепаху и перетаскиваем на неё любую форму рыбки из имеющегося набора. Но по условию нам потребуются две зеркальные формы. Для этого в рюкзаке у черепашки на закладке «Формы» копируем добавленную нами форму и вставляем её на вторую позицию. Открываем скопированную форму в редакторе форм, дважды щёлкнув по ней. При помощи инструмента  зеркально отображаем форму. Нажимаем ОК. Теперь можно перейти к написанию процедуры.

Мы уже знаем, как проверить, какой цвет находится под черепашкой. Задумаемся, какого результата мы хотим достичь. Во-первых, черепашка должна двигаться, во-вторых, встретив

стенку, изменить форму и направление движения. Осталось перевести всё это на алгоритмический язык.

```
это рыба1
  вп 1
  если цп = 105 [нф 1 нк 90 вп 1]
  если цп = 105 [нф 2 нк 270 вп 1]
конец
```

Команда *вп 1* внутри условия нужна для того, чтобы после проверки условия и изменения направления движения при следующем повторении черепашка сместилась бы с границы.

Помещаем имя процедуры в правила и ставим «Много раз». В поле ввода команд зададим рыбке начальный курс *нк 90*. Теперь оживим черепашку, чтобы проверить, что всё работает правильно.

Вторую рыбку совершенно не обязательно создавать с самого начала, так как процедура у неё будет в точности такая же. Поэтому просто копируем первую. В её копии удаляем формы и на их место из готовых форм перетаскиваем любую другую рыбку. Как и у первой черепашки делаем зеркально отражённую форму. Изменим имя процедуры с «рыба1» на «рыба2» как на закладке «Процедуры» так и на закладке «Правила».

Добавим кнопки «Пуск» и «Стоп» как это рассказано в проекте «Автомобили на дороге». Дополним наш аквариум различными подходящими к теме объектами. Например, можно добавить анимированную раскрывающуюся раковину (см. п. 7), растения и т.п.

В принципе, проект готов. Смущает только, что рыбы перед тем как изменить направление движения выходят на половину за границу аквариума. Это связано с тем, что черепашка отслеживает цвет поля под ней точкой находящейся в точности на середине формы. Чтобы обойти подобный эффект, надо применить совершенно иной подход к решению данной задачи – координатный. То есть наша черепашка-рыба должна будет следить за изменением координат, а не цвета под ней.

Скопируем ещё раз любую из двух наших рыб и сместим её к правой или левой стенке аквариума (в зависимости от того куда она развёрнута) так, чтобы форма рыбки не вылезала за край. В рюкзаке черепашки на закладке «Состояние» смотрим её текущие координаты.

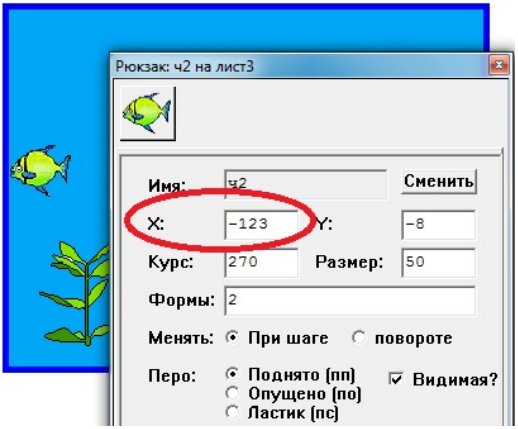


Рис. 37

Нас будет интересовать, конечно, координата по оси *x* – это и есть крайнее значение. Напомним, что текущее значение координаты хранится в соответствующем датчике – x_коор. Итак,

если x_коор > 123 [нк 90 нф 1 вп 1]

Аналогично поступаем с другой границей. Получим:

если x_коор < -123 [нк 270 нф 2 вп 1]

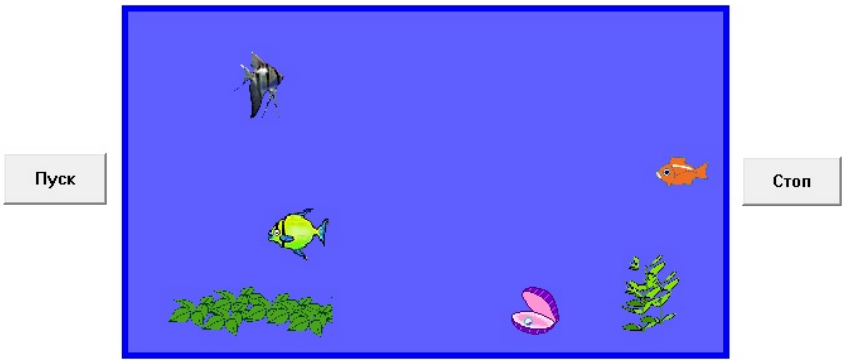


Рис. 38

Проект готов.

Задача 9.4 «Скачущий мячик»

Нарисуйте незакрашенный прямоугольник, у которого толщина границы была бы равна 4 точкам, растянув его по ширине на весь экран, а по высоте так, чтобы внизу осталось место для кнопок. Напишите программу, при выполнении которой по экрану будет летать шарик и отскакивать от стенок. Создайте кнопки запуска и остановки.

Решение

Для начала возьмём из имеющихся форм форму мячика или шарика и перетащим её, как показано на рисунке 24 на черепашку (см. п. 7).

Теперь подумаем над программой. Оказывается, что эта задача очень похожа на предыдущую. Разница только в том, что черепашка должна отталкиваться уже не от двух, а от четырёх границ и при этом она не должна менять форму. Поэтому первый вариант решения задачи даже проще, чем в предыдущем случае. Он будет выглядеть так:

```
это шарик1
  вп 3
  если цп = 105 [пр 90]
конец
```

Поместим имя этой процедуры в правила, поставим «Много раз». В поле ввода команд зададим черепашке курс 45, после чего оживим черепашку.

На самом деле решение должно выглядеть несколько сложнее:

```
это шарик1
  вп 3
  если цп = 105 [если_иначе курс = 0 - 180 [пр 90] лв[90]]
конец
```

Это связано с тем, что мячик не всегда может отскочить в нужную сторону.

Задание

Решите эту задачу в координатах (как в предыдущей задаче). Только в этом случае нужно задавать все четыре границы.

Задача 9.5 «Ряд кирпичей»

Напишите программу, выполняя которую черепашка нарисует ряд кирпичей.

Решение

Задача сводится к тому, что черепашка должна идти по горизонтали и оставлять оттиск кирпича. Для этого, как известно, есть команда «штамп».

В рюкзаке черепашки на закладке «Формы» откроем первую форму. С помощью инструмента «сплошной прямоугольник» нарисуем во всю ширину красный прямоугольник – это наш кирпич.

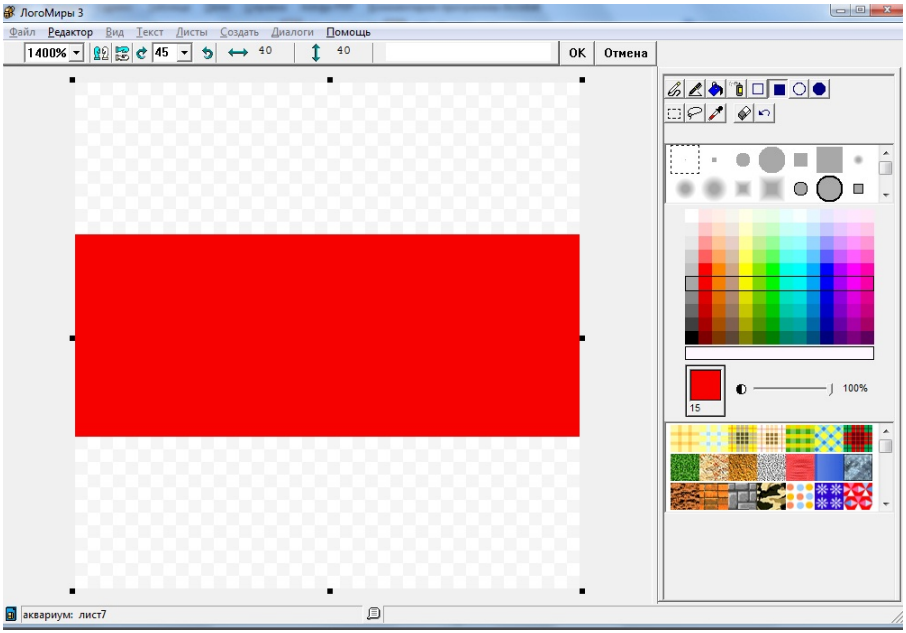


Рис. 39

Сделаем так, чтобы черепашка приняла нарисованную нами форму. Для этого или просто перетащим её из рюкзака на черепашку, или в поле ввода команд наберём *нф 1*. Увеличим размер черепашки до 50. Это можно сделать разными способами, например, можно задать команду *нрз 50* в поле ввода команд, а можно в рюкзаке на закладке «Состояние» ввести в поле «Размер» число 50.

Теперь поместим нашу черепашку в левый верхний угол и посмотрим её координаты в правилах. Первой командой в процедуре будет *нм[-330 160]*. Так как дальше черепашка должна двигаться направо, надо задать ей курс 90. Осталось совсем немного. Надо рассчитать на какое расстояние при каждом следующем шаге должна смещаться черепашка. Для этого в поле ввода команд напишем команду *штамп*. Черепашка оставит на плоскости оттиск текущей формы, т. е. кирпича. Сместим черепашку с помощью мыши направо так, чтобы между ней и кирпичом-штампом было небольшое расстояние. Посмотрим её координаты. Чтобы вычислить нужное нам расстояние смещения из текущего значения координаты *x* нужно вычесть предыдущее: -275 - (-330) = 55.

Запишем процедуру:

```
это кирпичи
 сг
 нм[-330 160]
 нк 90
 повтори 13 [штамп вп 55]
конец
```

Подберите начальное положение черепашки и расстояние, на которое она должна проходить.

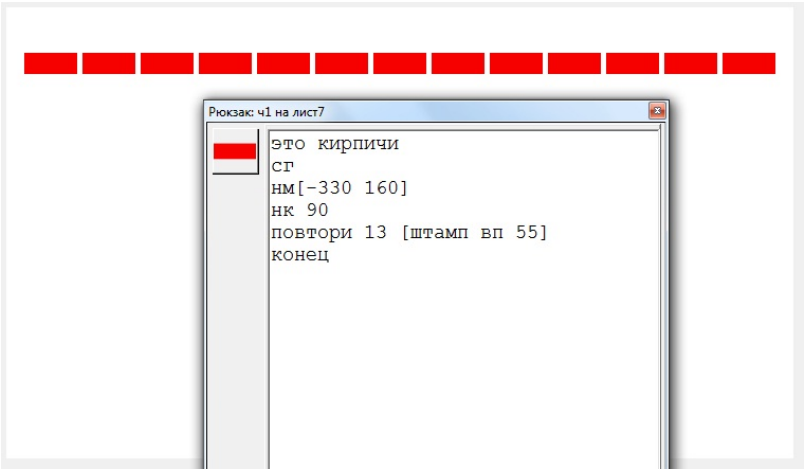


Рис. 40

Задача 9.6. Проект Игра «Арканоид»

Управляя с помощью бегунка движущейся по горизонтали палочкой, нужно шариком сбить все кирпичи.

Решение

Идея игры, мягко говоря, не нова. Она появилась на заре эры персональных компьютеров. Тем интереснее попробовать свои силы при создании собственной версии этой игры.

До этого момента мы не зря потрудились, так как последние задачи связаны как раз с этим проектом. Мы уже знаем, как нарисовать ряд кирпичей и как задать движение шарика. Нам осталось выяснить, как с помощью бегунка можно управлять палочкой. Давайте с этого и начнём. Создадим в новом проекте черепашку и нарисуем форму палочки примерно так:

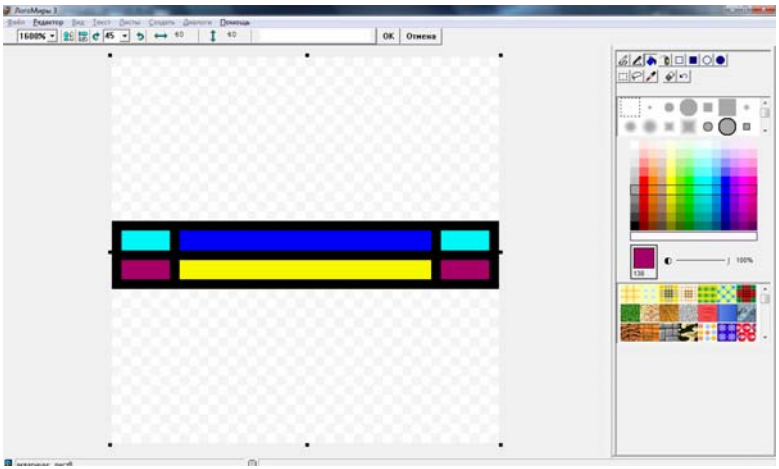


Рис. 41

Перетащим созданную форму на черепашку. Теперь, как и в задаче «Аквариум с рыбками», сместим палочку в крайне левое (или крайне правое) положение и посмотрим текущую координату по оси *x*. Создадим бегунок, назовём его «Двигай». В качестве минимального и максимального значений зададим крайние значения координаты *x* (см. рис. 42).

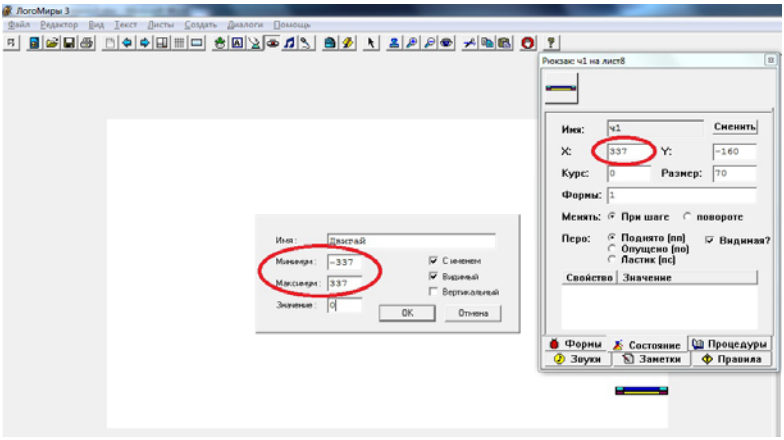


Рис. 42

Процедура у черепашки-палочки будет выглядеть очень просто: чётко фиксируем черепашку по оси *y* (*нов_y -160*), а по оси *x* координату надо брать из бегунка. Для этого нужно написать *нов_x Двигай*.

```
это палка
  нов_y -160
  нов_x Двигай
конец
```

Не забываем поместить имя процедуры в «Правила» и поставить «Много раз». Оживим черепашку и подвигаем бегунок. Если палочка при этом двигается, значит всё написано верно.

Следующим этапом будет создание черепашки, рисующей ряд кирпичей. Посмотрите задачу 8.5. Проделайте все необходимые действия, которые там описаны. Добавим в конец процедуры команды, скрывающие и отключающие черепашку.

```
это кирпичи
   сг
   нм[-330 160]
   нк 90
   повтори 13 [штамп вп 55]
   сч
   выкл
конец
```

Не хватает шарика. Обратимся к задаче 8.4. Воспользуемся вторым вариантом решения этой задачи (в координатах). Подводя последовательно шарик к правой, левой и верхней границе, запомним его крайние положения по соответствующим осям. Нужно чтобы от этих границ шарик отскакивал. Если же шарик пролетит мимо палочки (*у_коор < -160*), это означает, что игра проиграна. Надо показать окно с сообщением и остановить игру.

```
это шарик
   вп 1
   если х_коор >  364 [если_иначе курс = 0 - 180 [пр 90][лв 90]]
   если х_коор < -364 [если_иначе курс = 0 - 180 [пр 90][лв 90]]
   если у_коор >  205 [если_иначе курс = 0 - 180 [пр 90][лв 90]]
   если у_коор < -160 [сообщи [Проиграл!] останов]
конец
```

Чтобы шарик отбивался палочкой, надо добавить условие касания двух черепашек. Сначала приведём его в общем виде:

если коснулись? "имя1 "имя2 [действия]

В данном случае *действия* будут точно такими же, как при касании шариком любой границы:

```
если коснулись? "ч1 "ч3 [если_иначе курс = 0 - 180 [пр 90][лв 90]],
```

где *ч1* – имя черепашки-палочки, а *ч3* – имя черепашки-шарика.

Теперь надо подумать, как сделать так, чтобы кирпичи исчезали при попадании в них шарика. Оказывается, это довольно просто – надо перекрашивать их в цвет фона. То есть, как только под черепашкой-шариком окажется красный цвет кирпича, надо закрасить его белым цветом. При этом надо вести счёт попаданиям. Здесь есть некоторые трудности, так как нужен какой-то счётчик, в котором бы накапливалось число попаданий. Поскольку что такое переменная и как с ней работать мы пока не знаем, то выйти из этого затруднительного положения можно следующим образом. Воспользуемся в качестве такого счётчика размером пера шарика, которое в данной задаче мы вообще никак не используем. Зададим изначально в кнопке «Начать игру» минимальный размер пера – 1. При каждом попадании будем увеличивать его размер на 1, точно также как это сделано в

задаче 8.1 (Алгоритм роста репки). Только теперь будем увеличивать не размер черепашки и размер её пера. Не забудем, что при попадании шарика в кирпич, нужно чтобы он отскакивал от кирпича точно также как от стенок.

если цп = 15 [если_иначе курс = 0 - 180 [пр 90][лв 90] нц 0 красьнрп размер_пера + 1]

Добавим проверку условия победы:

если размер_пера = 14 [сообщи [Победа!] останов]

Почти готово. На всякий случай приведём процедуру для шарика ещё раз уже целиком:

```
это шарик
вп 1
если х_коор >  364 [если_иначе курс = 0 - 180 [пр 90][лв 90]]
если х_коор < -364 [если_иначе курс = 0 - 180 [пр 90][лв 90]]
если у_коор >  205 [если_иначе курс = 0 - 180 [пр 90][лв 90]]
если у_коор < -160 [сообщи [Проиграл!] останов]
если цп = 15 [если_иначе курс = 0 - 180 [пр 90][лв 90] нц 0 красьнрп размер_пера + 1]
если размер_пера = 14 [сообщи [Победа!] останов]
конец
```

Как всегда не забудьте поместить имя этой процедуры в «Правила» и поставить «Много раз». Добавьте кнопку «Начать игру». Инструкции в ней будут такими:

ч3, нрп 1 нк 45 нм [0 -150] каждая [вкл]

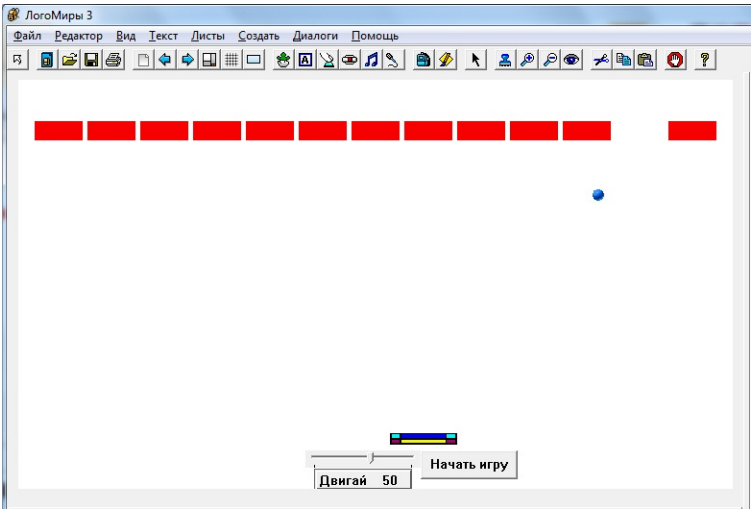


Рис. 43

Отладьте проект так, чтобы не было ошибок. Проверьте, появляются ли у вас сообщения о победе и поражении. Если всё нормально, то теперь можно дать кому-то поиграть в созданную вами игру.

10. Текстовые поля. Переменные. Проект «Калькулятор».

10.1 Текстовые окна.

Чтобы создать текстовое окно, как и в случае создания большинства других объектов, есть два способа: первый – из меню «Создать» → «Текстовое окно», второй – с помощью инструмента  .

После создания окно можно переместить в нужное место и растянуть до нужного размера, предварительно выделив его и схватив за любой из углов.

У тестового поля есть 4 атрибута, которыми можно управлять из пункта контекстного меню «Редактировать»:

- с именем;
- в одну строку;
- прозрачный;
- видимый.

Создайте текстовое окно и самостоятельно изучите, как влияет на него тот или иной атрибут.

Помимо этих атрибутов в текстовом поле можно менять все параметры отображающегося в нём текста, а именно: цвет, размер, тип, способ начертания.

Попробуйте написать в текстовом поле любой текст и, воспользовавшись соответствующими пунктами меню «Текст», измените тип шрифта, его размер и цвет.

Как и в случае с черепашкой вносить изменения в текстовое поле можно не только «вручную», но и с помощью команд. Нам понадобятся следующие:

Команда (сокращение)	Датчик	Пример	Назначение
нов_цвет_текста (нцт)	цт	нцт 15	изменяет цвета текста
нов_размер_шрифта (нрш)	рш	нк 90	изменяет размера шрифта
пиши	–	а, пиши "Текст а, пиши 5 (для цифр без ")	Печатает слово или список в активном текстовом окне.
ст	–	а, ст	Стирает текст.

Для того чтобы вносить изменения в конкретное текстовое поле, надо предварительно обратиться к нему по имени и после запятой написать нужные команды.

Пример

Создадим текстовое поле. С помощью щелчка правой кнопкой мыши по нему зайдём в меню «Редактировать». Зададим новое имя окна *m1*. Все остальные атрибуты пока оставим без изменения. Нажмём ОК. Теперь в поле ввода команд запишем следующее:

m1, нрш 20 нцт 105 пиши "Привет!

После выполнения этих команд в нашем текстовом окне мы увидим надпись «Привет!», причём размер букв будет равен 20, а цвет – синим.

Если теперь мы наберём в поле ввода команд: *m1, пиши 7* и выполним их, то увидим в нашем поле число 7. Для того чтобы удалить весь текст из поля «*m1*» напишем *m1, ст*.

Обладая этим нехитрым набором команд можно вполне создать самый настоящий...

10.2 Проект «Калькулятор». Первый вариант.

Создадим новый проект. Сделаем несколько текстовых полей: во-первых, заголовок «Калькулятор», во-вторых, подписи к полям, в которые будем вводить числа – «Первое число», «Второе число», «Ответ», в-третьих, поля, куда будем вносить сами числа и показывать результат. Давайте сразу переименуем поля: первое поле назовём «*a*», второе – «*b*», третье – «*otvet*». Имя «ответ» по-русски зарезервировано самой средой для других целей, поэтому в данном случае использовать его нельзя. Поставим для всех полей атрибут «одна строка».

Кроме этого нам понадобятся ещё 4 кнопки – арифметические операции. Мы их так и назовём: «+», «-», «*», «/». С инструкциями для кнопок немного сложнее. Однако догадаться, что там необходимо записать всё-таки достаточно просто. Задумаемся, что мы собираемся сделать. Нам нужно, предварительно очистив поле «otvet», взять одно число из первого текстового поля «a», другое – из второго «b», и результат вывести в третьем. Переведём это на алгоритмический язык. Например, для кнопки сложения надо записать следующие инструкции: *otvet, cm пиши a + b.*

Для остальных действий инструкции будут аналогичными: вычитание – «-», умножение – «*», деление «/». Не забывайте отделять знаки арифметических операций пробелами с двух сторон.

Добавим кнопку обнуления текстовых полей. Назовём её «C», а в инструкциях запишем: *a, cm b, cm otvet, cm*.

Осталось оформить нашу работу. Подберите самостоятельно размер и цвет шрифта для всех текстовых полей. Выберите фон.

Небольшое замечание: при вводе чисел следите за тем, чтобы в текстовых полях не было пробелов и переходов на следующую строку. Иначе возникнут ошибки!

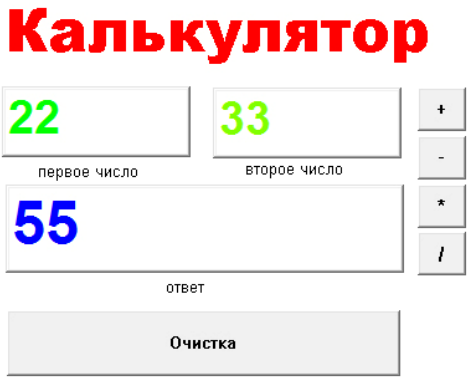


Рис. 44

10.3 Переменные.

Вообще говоря, ***переменная*** *– это величина, которая может изменять своё значение*. Для нас же важен вот какой момент: когда мы задаём переменную, в оперативной памяти компьютера выделяется специальная ячейка, где хранится значение переменной. Это значение всегда можно извлечь.

Чтобы задать переменную есть команда *пусть*.

пусть "имя_переменной значение

Например, *пусть "а 123*

Для того чтобы посмотреть значение переменной, нужно с помощью уже известной нам команды «пиши» обратиться к переменной по имени через «:».

пиши :а

Для тренировки последовательно в поле ввода команд напишите следующее:

```
пусть "число 100
покажи :число
пусть "число :число + 50
покажи :число
```

Как вы уже догадались, мы к имеющемуся значению переменной «*число*» 100 прибавили ещё 50 и вывели полученный результат (её текущее значение).

10.4 Проект «Калькулятор». Второй вариант.

На втором листе проекта «Калькулятор» создадим второй его вариант, который будет очень похож на самый обыкновенный калькулятор или на калькулятор, встроенный в ОС Windows (рис. 45).

Нам понадобится одно текстовое поле (назовём его «x», поставим атрибуты «без имени» и «одна строка»), кнопки для набора цифр, арифметических операций, знака равенства и некоторых других функций.

Рис. 45

При нажатии на кнопки с цифрами в поле «x» нужно передавать соответствующую цифру. Например, в кнопке «1» напишем: *x, пиши 1*.

В кнопке «,» следует написать: *x, пиши ",*

Подумаем, что должно происходить при нажатии на кнопки с арифметическими операциями. Нам нужно запомнить число из текстового поля и ту операцию, которую мы хотим выполнить. Для запоминания арифметического действия введём дополнительную переменную «действие». Запомнить первое введённое в текстовое поле число можно двумя способами: первый – используя переменную, второй – используя промежуточное текстовое поле. Если мы воспользуемся первым способом, тогда инструкция в кнопке «+» будет выглядеть следующим образом:

пусть "a x пусть "действие 1 x, ст

В переменную «а» мы передали значение из текстового поля «х», затем запомнили операцию, передав в переменную «действие» значение 1 и, наконец, очистили текстовое поле «х». Если же мы хотим воспользоваться вторым способом, то нам понадобятся два вспомогательных текстовых поля «а» и «б». Их надо сделать скрытыми. Тогда инструкция в кнопке будет выглядеть немного по-другому:

а, ст пиши х пусть "действие 1, х, ст

При нажатии на кнопку «=» нужно чтобы второе число передалось или во вторую переменную или во второе вспомогательное текстовое поле. Затем в зависимости от значения переменной «действие» нужно выполнить арифметическую операцию между двумя числами и результат передать в поле «х».

Создадим вспомогательную процедуру «равно» в правой части экрана:

Первый вариант (если используются только переменные).

```
это равно
пусть "б х
х, ст
если :действие = 1 [пусть "с :а + :б]
если :действие = 2 [пусть "с :а – :б]
если :действие = 3 [пусть "с :а * :б]
если :действие = 4 [пусть "с :а / :б]
х, пиши :с
конец
```

Второй вариант (если используются вспомогательные текстовые поля)

```
это равно
б, ст пиши х
х, ст
если :действие = 1 [х, пиши а + б]
если :действие = 2 [х, пиши а – б]
если :действие = 3 [х, пиши а * б]
если :действие = 4 [х, пиши а / б]
конец
```

Осталось вызвать эту процедуру в кнопке «=».

Кнопка «С» – очистка поля «х».

Кнопка «MS» запоминает число из поля «х», кнопка «MR» восстанавливает значение в этом поле. Чтобы запомнить число из поля «х» задайте любую переменную и в качестве её значения укажите имя поля «х». Чтобы число вновь появилось в поле «х», напишите

х, ст пиши :имя_вашей_переменной.

Остались кнопки с дополнительными функциями. Например, кнопка «+/–» меняет знак числа на противоположный. Чтобы поменять знак числа в Логомирах есть функция «минус». То есть в инструкциях этой кнопки будет следующее:

пусть "число минус х х, ст пиши :число

Сначала передаём во вспомогательную переменную «число» значение из поля «х» с минусом, затем очищаем и пишем новое значение переменной.

Тем, кому станет интересно, может добавить дополнительные кнопки, которые выполняют те или иные функции. Например, кнопку «1/х», при нажатии на которую в поле будет появляться результат деления 1 на число, находящееся в текущий момент в текстовом поле. Инструкции будут очень похожими на те, которые записаны в кнопке «+/–», только вместо «минус» надо написать «1 / х».

Отключите все элементы проекта для того чтобы случайно их не сдвинуть. Проверьте работу всех кнопок.

Задание.

Вернитесь к проекту «Арканоид» и исправьте его таким образом, чтобы количество попаданий шарика по кирпичам заносилось не в размер пера черепашки, а в переменную.

10.5. Задача. Проект Игра «ПВО»

Создайте ракетную установку и два самолёта. Сделайте так, чтобы при нажатии на кнопку «Огонь» из ракетной установки вылетала ракета. Если самолёт сбит, то он должен изменить форму, полететь к земле и взорваться.

Решение

Начнём с рисования ракетной установки.

Создадим черепашку, откроем её рюкзак. На вкладке «Состояние» изменим имя на «ру» (ракетная установка) и перейдём на закладку «Формы». Дважды щёлкнем по первой позиции, открыв редактор форм. Удалим из центра маленький квадратик. Теперь нарисуем незакрашенный прямоугольник тёмно-зелёного цвета на всю высоту формы. Разделим с помощью ручки тем же цветом этот прямоугольник на несколько неправильных областей. Зальём оттенками зелёного, серого и коричневых цветов сделанные области. Всё. Ракетная установка готова.

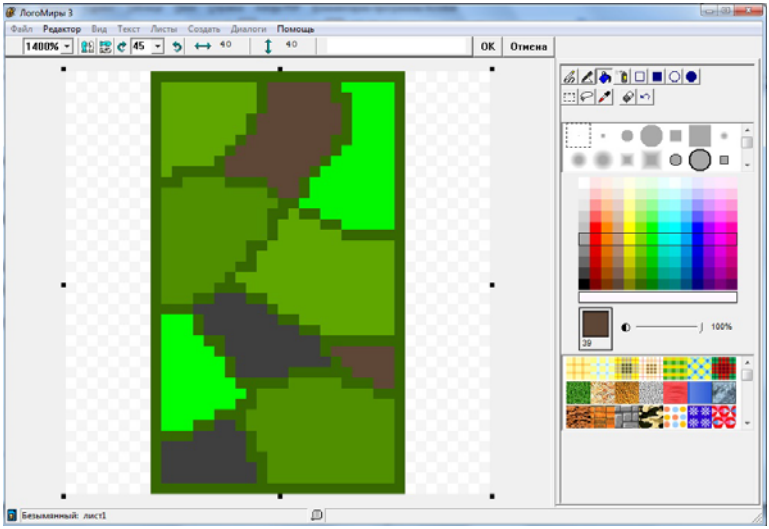
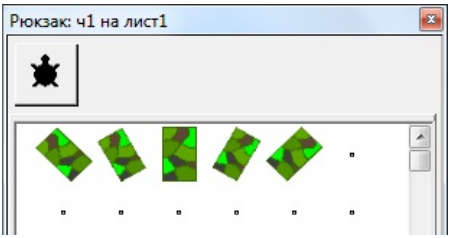


Рис. 46

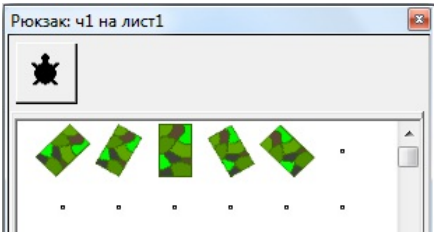
Нажимаем ОК. Теперь давайте скопируем эту форму ещё 4 раза. То есть всего у нас должно получиться 5 одинаковых форм. Снова откроем первую форму и развернём её на 45° налево с помощью инструмента 

. Вторую форму точно также

развернём на 30° налево, причём здесь надо в поле количества градусов ввести угол поворота с клавиатуры. Третью форму оставляем неизменной. Четвёртую поворачиваем на 30° направо. Пятую – соответственно на 45° направо. В результате у нас получится как бы веер форм:



Правильная картина



Неправильная картина

Рис. 47

Только не перепутайте, чтобы не получилась неправильная картина, показанная на рис. 47. Создадим бегунок и назовём его «Поворот». Мы хотим, чтобы формы менялись в зависимости от положения бегунка. Минимальное значение зададим 1, а максимальное – 5. Текущее значение бегунка можно задать любое из этого диапазона, например, 2. Поскольку установка никуда не должна двигаться, жёстко зафиксируем её с помощью команды *нм*. Теперь напишем простую процедуру:

```
это ракетная_установка
  нм [0 -150]
  нф Поворот
конец
```

Не забудьте о том, что имя процедуры не должно содержать пробелов. Поэтому между двумя словами в названии стоит символ «_». Помещаем имя процедуры в «Правила» и ставим «Много раз». Оживляем черепашку и двигаем бегунок. Если вы сделали всё правильно, то ракетная установка должна изменять угол наклона.

Точно также как мы рисовали установку, нарисуем ракету. Создадим вторую черепашку. Изменим её имя на «р». Также сделаем 5 форм и развернём их под теми же углами.

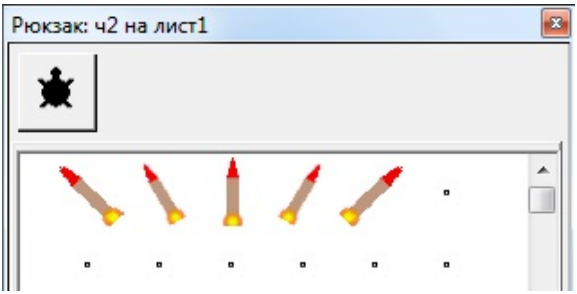


Рис. 48

Первая процедура для ракеты выглядит так:

```
это ракета
  нф Поворот
конец
```

В процедуре для ракеты мы не будем фиксировать её положение командой *нм*. Процедуру помещаем в «Правила» «Много раз». Дальше надо подумать, каким образом задать полёт ракеты. Во-первых, в зависимости от бегунка ракета должна лететь под определённым углом. То есть надо задавать курс. Во-вторых, в зависимости от того под каким углом будет лететь ракета, расстояние, которое она пролетит должно быть разным. Эту и другие процедуры будем писать в правой части экрана на вкладке «Процедуры».

```
это полёт_ракеты
   пч
   если Поворот = 1 [нк 315 повтори 240 [вп 2]]
   если Поворот = 2 [нк 330 повтори 200 [вп 2]]
   если Поворот = 3 [нк  0   повтори  170 [вп 2]]
   если Поворот = 4 [нк  30 повтори 200 [вп 2]]
   если Поворот = 5 [нк  45 повтори 240 [вп 2]]
   сч  нм [0 -150]
конец
```

Долетев до верхней точки и не попав в самолёт, ракета должна исчезнуть и вернуться в исходную точку, то есть на то же место, на котором находится установка. Но в начале процедуры, естественно, ракету надо показать.

Создадим кнопку «Огонь» и в инструкциях напишем: *р, полёт_ракеты*. Проверим, как всё работает. Ракета должна

вылетать при нажатии на кнопку «Огонь» и исчезать, не перелетая за границу окна. Изменяя количество повторений, добейтесь того, чтобы ракета не появлялась снизу экрана. Также надо внимательно следить за синтаксисом.

Можно перейти к самолётам. Сделаем сначала один. Создадим черепашку и выберем из имеющихся форм самолётик. Зайдём в её рюкзаке на вкладку «состояние» и изменим имя на «с1». Поставим черепашку в то место, с которого она должна начать движение. Так как в процессе выполнения программы, самолётик будет изменять параметры направления движения, формы, размера и местоположения, эти параметры мы будем возвращать в самом начале игры. Для этого напишем первую процедуру, задающую начальные установки самолётика:

```
это сам1_начало
    нов_х -320
    нов_у сл 180
    нк 90
    нф 1
    нрз 30
    пч
конец
```

Теперь заготовим формы для самолёта, которые будут включаться при попадании в него ракеты.

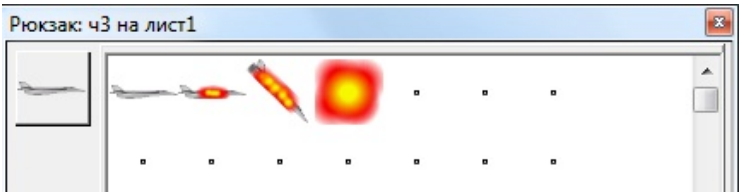


Рис. 49

Запишем, что должно произойти, когда самолёт сбит:

1) он должен поменять форму на вторую;
2) пролететь небольшое расстояние по горизонтали;
3) изменить форму на третью;
4) изменить курс;
5) полететь к земле;
6) взорваться;

Затем надо скрыть и выключить эту черепашку. При этом ракету сразу возвращаем на место. После чего снова вызываем процедуру начальных установок самолёта.

Запишем всё выше сказанное алгоритмическим языком:

```
это сам1_полёт
   вп 2
   если коснулись? "с1 "р
   [
      р, сч нм[0 -150]
      с1,нф 2
      повтори 20 [вп 2]
      нф 3
      нк 135
      повтори 200 [вп 2]
      нф 4
      повтори 130 [нрз размер + 1]
      сч
      сам1_начало
   ]
конец
```

Не забудьте поместить имя этой процедуры в «Правила» «Много раз». Проверьте работоспособность этой программы. Отладьте программу так, чтобы всё работало, как мы задумали. Если всё нормально, сделайте копию самолётика. После чего выполните…

Задание

Исправьте самостоятельно программу для второго самолётика так, чтобы он летал ниже первого и в противоположном направлении.

Осталось сделать кнопку запуска программы «Начать игру». Напишем процедуру «начало», в которой перечислим все действия, которые необходимо выполнить при запуске игры.

```
это начало
   с1, сам1_начало
   с2, сам2_начало
   каждая [вкл]
конец
```

В качестве инструкции в кнопке «Начать игру» напишем имя этой процедуры.

Дополним проект различными объектами: небом, землёй, облаками, деревья и т.п.

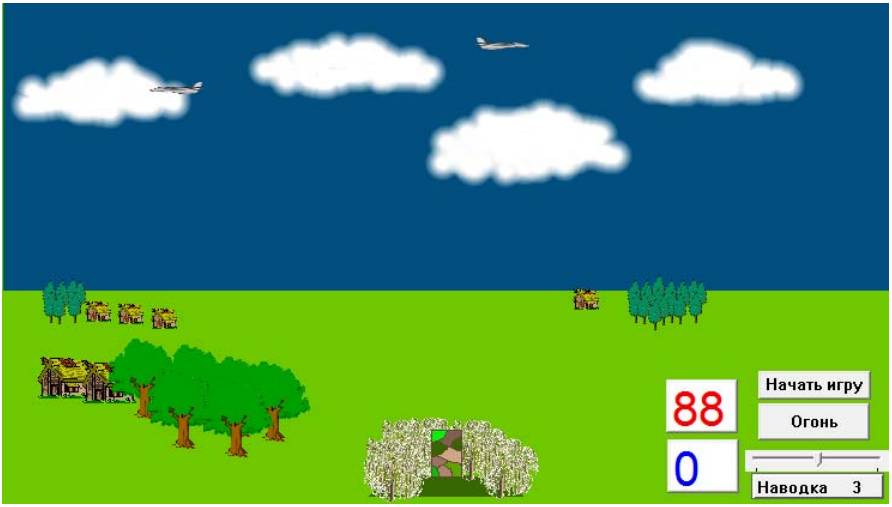


Рис. 50

Теперь, воспользовавшись знаниями о переменных и текстовых полях, добавим в проект обратный отсчёт времени и счётчик сбитых самолётов.

Создадим два текстовых поля. Одно назовём «время», второе – «сб_самолёты». Поставим для каждого поля атрибуты «без имени» и «одна строка». Добавим в процедуру «начало» команду объявления переменной «сб» со значением 0.

пусть "сб 90

Очистим текстовое поле «сб_самолёты»:

сб_самолёты, ст

Теперь в конце каждой процедуры полёта самолёта после цикла, в котором моделируется взрыв, добавим счётчик:

пусть "сб :сб + 1.

То есть при каждом попадании значение переменной «сб» будет увеличиваться на 1. И команду, передающую значение этой переменной в текстовое поле «сб_самолёты»:

сб_самолёты, ст нрш 30 нцт 65 пиши :сб

Кроме этого нам понадобится проверка условия победы:

если :сб = 10 [сообщи [Победа!]с1, сч с2, сч останов]

Чтобы задать в поле «время» обратный отсчёт, напишем процедуру «обр_отсч». В ней нужно задать переменную «вр» со значением 90, затем в цикле надо с задержкой передавать значение этой переменной в поле «время» и вычитать 1. Как только время закончится, выведем сообщение о поражении.

```
это обр_отсч
пусть "вр 90
нцт 15
нрш 30
повтори 90 [Время, ст Время, пиши :вр пусть "вр :вр - 1 жди 10]
если :вр = 0 [Время, пиши 0 сообщи [Проиграл!] сам1, сч сам2, сч останов]
конец
```

Обязательно нужно соблюсти порядок следования процедур:

- процедура полёта ракеты,
- процедуры начальных установок самолёта,
- процедура обратного отсчёта времени,
- процедура начала,
- процедуры полёта самолётов.

Самой последней строкой в процедуре «начало» вызываем «обр_отсч».

Если вы всё сделаете правильно, то проект будет выглядеть примерно так, как показано на рисунке 50. При нажатии на кнопку «Начать игру» самолёты начнут движение, в поле «время» будет идти обратный отсчёт, и при каждом попадании ракеты по самолёту во втором текстовом поле будет прибавляться единица.

10.6. Задача. Проект Игра «Сокобан»

Управляя персонажем (Сокобаном) в лабиринте нужно переместить сундучки (в нашем случае жёлтые квадратики) на заданные позиции – ключи (в нашем случае синие крестики). Необходимо сделать как минимум три уровня на разных листах.

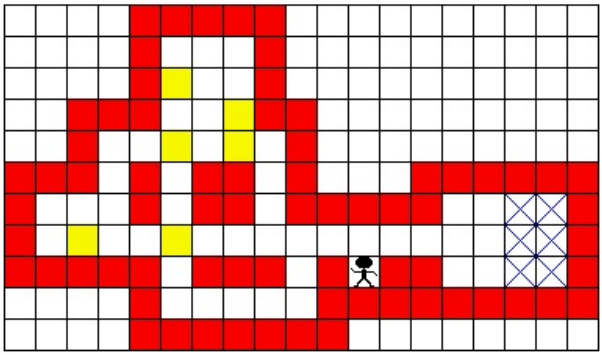


Рис. 51

Решение

Шаг 1. Создание форм.

1. Напишите процедуру, рисующую квадрат со стороной 20 точек.
2. Выделите нарисованный квадрат с помощью инструмента «Прямоугольное выделение» (см. рис. 50) на панели «Рисование/Графика» точно по границе и скопируйте его в буфер обмена.

Инструмент «Прямоугольное выделение»

Рис. 52

3. Откройте в рюкзаке черепашки на закладке «Формы» первую форму. Удалите оттуда всё лишнее и вставьте скопированный квадрат.
4. Перетащите форму на черепашку. Создайте ещё две копии этой формы. Второй квадрат залейте красным (15) цветом, третий – жёлтым (45) цветом.
5. Удалите процедуру, рисующую квадрат.
6. Поднимите перо черепашки, введя в поле ввода команд команду *пп*.

Шаг 2. Создание сетки.

1. Надо придумать алгоритм, выполняя который черепашка нарисует сетку 12×19 квадратов. Начнём с одного ряда. Оказывается, это в точности то же самое, что штамповать кирпичи. Шаг смещения равен длине стороны квадрата – 20 точек. То есть ряд из двадцати квадратиков черепашка нарисует, выполнив следующие действия: *повтори 19 [штамп вп 20]*.

2. Теперь надо нарисовать таких рядов много. Здесь возможны разные подходы. Наиболее общим будет такой: по оси *x* будем при каждом повторении смещать черепашку в начало строки, а по оси *y* надо изменять положение черепашки на одну позицию вниз. Однако, если *x* будет всегда одним и тем же, то *y* будет постоянно меняться. Поэтому при каждом шаге цикла из текущей координаты *y* будем вычитать 20. И так как всего рядов должно быть 11, получим:

повтори 11 [повтори 19 [штамп вп 20] нов_x -190 нов_y y_коор - 20]

3. Поскольку черепашка штампует ряды слева направо, то изначально надо задать её положение так, чтобы вся сетка располагалась ближе к верхнему краю, а по горизонтали в середине окна проекта *нм [-200 160]* и курс – *нк 90*. В итоге получилась такая процедура:

```
это лабиринт
   сг
   нк 90
   нм [-190 160]
   повтори 11 [повтори 20 [штамп вп 20] нов_x -190 нов_y y_коор - 20]
конец
```

Шаг 3. Рисуем лабиринт.

Черепашка должна автоматически рисовать лабиринт при нажатии на кнопку «С начала». Надо научить её это делать. Как всегда подход может быть разным: границу лабиринта можно рисовать, закрашивая квадратики с помощью команды «крась», а можно сделать красную форму квадрата и штамповать её. Используем второй подход, т. к. в этом случае скорость

рисования лабиринта будет гораздо выше. К тому же мы уже заготовили формы красного и жёлтого квадратов.

Постарайтесь нарисовать лабиринт самостоятельно. Хотя окончательный вариант процедуры, рисующей лабиринт, всё же стоит привести:

```
это лабиринт
    сг
    пч
    нф 1
    нк 90
    нм [-190 160]
    повтори 11 [повтори 19 [штамп вп 20] нов_х -190 нов_у
    у_коор - 20]
    нм [-190 0]
    нф 2
    нк 0   повтори 3 [вп 20 штамп]
    нк 90  повтори 2 [вп 20 штамп]
    нк 0   повтори 2 [вп 20 штамп]
    нк 90  повтори 2 [вп 20 штамп]
    нк 0   повтори 3 [вп 20 штамп]
    нк 90  повтори 4 [вп 20 штамп]
    нк 180 повтори 3 [вп 20 штамп]
    нк 90  вп 20 штамп
    нк 180 повтори 3 [вп 20 штамп]
    нк 90  повтори 4 [вп 20 штамп]
    нк 0   вп 20 штамп
    нк 90  повтори 5 [вп 20 штамп]
    нк 180 повтори 4 [вп 20 штамп]
    нк 270 повтори 8 [вп 20 штамп]
    нк 180 вп 20 штамп
    нк 270 повтори 6 [вп 20 штамп]
    нк 0   повтори 2 [вп 20 штамп]
    нк 270 повтори 4 [вп 20 штамп]
    нм [-110 60] штамп
    нм [-110 40] штамп
    нм [-70  60] нк 90 повтори 4 [штамп вп 20 пр 90]
    нм [-70   0] повтори 3 [штамп вп 20]
```

```
    нм [10 0] штамп
    нм [50 0] штамп
    нм [70 0] штамп
    нф 3
    нм [-150 20] штамп
    нм [-90 120] штамп
    нм [-90  80] штамп
    нм [-90  20] штамп
    нм [-50 100] штамп
    нм [-50  80] штамп
    сч
    выкл
конец
```

Несмотря на то, что это, пожалуй, самая длинная процедура, которую мы с вами создавали, она довольно проста по содержанию, так как в ней используются однотипные конструкции. Самое главное при написании не ошибиться в подсчёте числа повторений и синтаксисе.

Шаг 4. Создание черепашек-ключей.

На рис. 47 есть 6 синих крестиков. Это те места, куда надо переместить жёлтые квадраты-сундучки. На них нужно поставить 6 черепашек. Для начала сделаем одну. Скопируем форму незакрашенного квадрата на две первые позиции из рюкзака черепашки, рисующей лабиринт. На первой форме нарисуем ручкой две синие диагонали, на второй – инструментом «Сплошной овал» зелёную окружность (рис. 53).

Рис. 53

Функция каждой из шести черепашек предельно проста – проверка цвета поля под ней. Если цвет поля оказывается жёлтым, значит, на это место мы сдвинули сундучок. А поскольку они не должны сдвигаться с места, процедура будет выглядеть так:

```
это на_месте1
  нм [130 40]
  если цп = 0  [нф 1]
  если цп = 45 [нф 2]
конец
```

Естественно, процедура должна повторяться в правилах много раз.

Копируем первую черепашку-ключ, ставим её на следующую позицию, изменяем её координаты и название процедуры. Аналогично поступаем с остальными копиями.

Шаг 5. Сокобан.

Нам нужен самый главный персонаж – Сокобан, который будет перемещать сундучки. Создадим новую черепашку. Можно опять скопировать форму квадрата из рюкзака первой черепашки в рюкзак Сокобана, а затем внутри него нарисовать человечка. Добавим в проект кнопки перемещения: «Вверх», «Вниз», «Вправо», «Влево». Инструкция для каждой из них достаточно очевидна. Нужно задавать курс и смещение. Например, у кнопки «Вниз» инструкция будет: *ч8, нк 180 вп 20* (ч8 – это, конечно, черепашка-сокобан). Аналогично устроены и остальные кнопки.

Теперь самое сложное. Как заставить нашего главного героя не выходить за границы лабиринта и перемещать сундучки? С первой задачей мы справимся без труда. При каждом смещении кнопками будем проверять цвет поля. Если он окажется красным (15), то просто будем шагать назад. Вторая часть сложнее. Давайте разберём все возможные ситуации после того, как Сокобан сделает шаг:

1) Под ним окажется цвет границы. Тогда, как мы выяснили, надо сделать шаг назад.
2) Под ним окажется белый цвет. Ничего не делаем.
3) Под ним окажется жёлтый цвет. Тогда надо шагнуть ещё на одну позицию дальше и посмотреть, что находится за жёлтым квадратом. Если там белый квадрат, то мы красим эту клетку жёлтым, возвращаемся на один шаг назад и красим белым. Тем самым создадим иллюзию движения сундучка с одной позиции на другую. Если же там красный или жёлтый, то надо вернуться обратно на две позиции, так как Сокобан не должен двигать сразу два сундучка.

Теперь запишем всё это алгоритмическим языком:

это сокобан
если цп = 15 [нд 20]
если цп = 45 [вп 20 если_иначе цп = 0 [нц 45 крась нд 20 нц 0 крась][нд 40]]
конец

На сколько всё короче! Не забываем поместить имя процедуры в «Правила» «Много раз».

Шаг 6. Кнопка запуска программы.

Как обычно нужно сделать кнопку начала игры. Назовём её «С начала». Что должно произойти при нажатии на эту кнопку?

1) Нужно чтобы исчезли все черепашки, и стёрлась вся графика.
2) Первая черепашка должна нарисовать лабиринт.
3) Должны появиться черепашки-ключи.
4) На стартовой позиции должен появиться Сокобан.
5) Наконец, все черепашки кроме первой должны ожить.

Теперь можно записать инструкции для этой кнопки:

каждая [сч выкл] ч1, лабиринт ч8, нм [30 0] каждая [пч вкл] ч1, сч выкл

Отладьте проект так, чтобы не возникало ошибок и всё работало корректно.

Задание

Создайте ещё три уровня в том же проекте на других листах, показанные на рисунке. Назовите листы: «уровень1», «уровень2», «уровень3», «уровень4». В именах листов пробелов быть не должно! Сделайте кнопки переходов между ними, указав в качестве инструкции в каждой кнопке имя листа.

Совет

Для удобства можно скопировать лист с уже созданным уровнем и затем немного видоизменить программу рисования уровня, количество и места расположения черепашек-ключей.

Рис. 54

11. Переключатели. Процедура «startup». Проект «Тест».

Тесты окружают нас повсюду – начиная со страниц сети Интернет и заканчивая ЕГЭ, и охватывают практически все сферы человеческой деятельности – от устройства на работу и до выявления психологических и психических отклонений. Не смотря на ряд существенных недостатков, тесты обладают и огромным числом преимуществ, к которым можно отнести следующие: быстрота проверки, возможность машинной обработки, массовость.

Мы с вами создадим тест, который будет проверять знание материала по начальному уровню программирования в среде Логомиры. С помощью него вы сможете реально оценить уровень подготовки ваших одноклассников, и, может даже, протестируете родителей, которые вместе с вами изучали основы программирования. Заодно мы повторим весь тот материал, который прошли за год.

Для создания теста нам понадобятся ещё две вещи, с которыми мы ещё не познакомились: переключатели и процедура «sturtup».

11.1 Переключатели.

Переключатель – это объект, в котором можно задать несколько вариантов ответов и осуществлять выбор между ними.

В отличие от других объектов, создать переключатель можно только из меню «Создать» → «Переключатель».

Как и у любого объекта в Логомирах у переключателя есть имя. Имеются также и атрибуты: «видимый», «с именем», назначение которых очевидно.

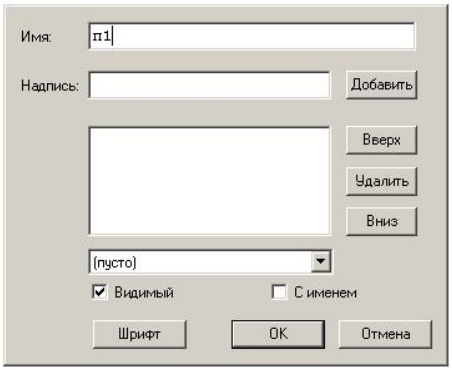


Рис. 55

В поле «Надпись» добавляются варианты ответов, после чего нужно нажимать на кнопку «Добавить». Кроме этого здесь можно задать изначальный выбор, который должен быть в переключателе (по умолчанию стоит «пусто»).

Поэкспериментируем. Создадим переключатель и добавим туда несколько вариантов ответов:

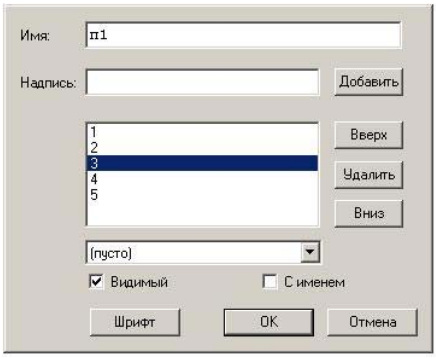


Рис. 56

После нажатия на кнопку ОК мы увидим следующую картину:

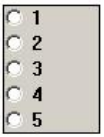


Рис. 57

Это так называемый radiobatton (радиобаттон) – переключатель, в котором можно выбрать лишь один вариант ответа. Значением переключателя можно управлять и с помощью команд. Например, для того чтобы обнулить (снять выбор) значение переключателя следует выполнить такую команду:

установи_имя_переключателя 0,

где в качестве имени в нашем случае следует написать «*n1*», то есть *установи_n1 0*. Попробуйте набрать эту команду в поле ввода команд. Выполнив её, вы убедитесь, что выбор обнулится. Помимо этой команды нам понадобится возвращать значение выбора переключателя. Как его узнать? Дело в том, что выбор в переключателе – это его номер по счёту сверху. Если в поле ввода команд набрать команду *пиши n1*, то строчкой ниже мы увидим число – это номер нашего выбора. Если выбор не сделан, мы увидим 0.

11.2 Процедура «sturtup».

Несобственная процедура (см. п. 4.3), имеющая специфическое имя startup, выполняется сразу при открытии проекта.

Если записать

```
это startup
    спроси [Как ваше имя?]
    сообщи пред [Привет,] ответ
конец
```

затем сохранить проект, закрыть его и открыть снова, то мы увидим окно, в котором нас спрашивают «Как ваше имя?» и поле для ввода ответа. После заполнения этого поля и нажатия на кнопку ОК, появится следующее окно с надписью «Привет, введённое имя».

Всё готово для того чтобы можно сделать последний…

11.3 Проект «Тест».

Структура проекта:

- титульный лист;
- листы с вопросами;
- итоговый лист.

Процедуры (несобственные):

- очистка текстовых полей на итоговом листе (очисти_текст);
- очистка переключателей на листах с вопросами (очисти_переключатели);
- процедура startup;
- процедура занесения значений переключателей в текстовые поля на итоговом листе (запись);
- процедура вычисления количества набранных балов и оценки (сч_оценку).

Начнём с выбора обоев. Учтите, фон должен быть максимально нейтральным, не раздражать и вместе с тем сделать возможным чтение текста вопросов.

Поскольку тест будет проверять знания по предмету «Информатика и ИКТ», на титульном листе создадим текстовое поле, в котором крупно напишем (рис. 53):

Итоговый тест по информатике и ИКТ
5 класс

Добавим кнопку «Начать тест». В инструкциях к этой кнопке введём:

очисти_переключатели вопрос1,

где *вопрос1* – имя листа с первым вопросом.

Рис. 58

Для того чтобы сохранить фон, скопируйте первый лист. Задайте ему имя *вопрос1*. Удалите с него все элементы. Создайте текстовое поле, в которое запишите «Вопрос №1». Теперь поместим в ещё одно текстовое поле сам вопрос. Следует заметить, что объём текста вопроса может весьма различаться. Поэтому нужно предусмотреть ситуацию, когда вопрос займёт практически всю площадь листа. Сделайте переключатель и внесите туда варианты ответов. Сразу измените имя переключателя – *n1*.

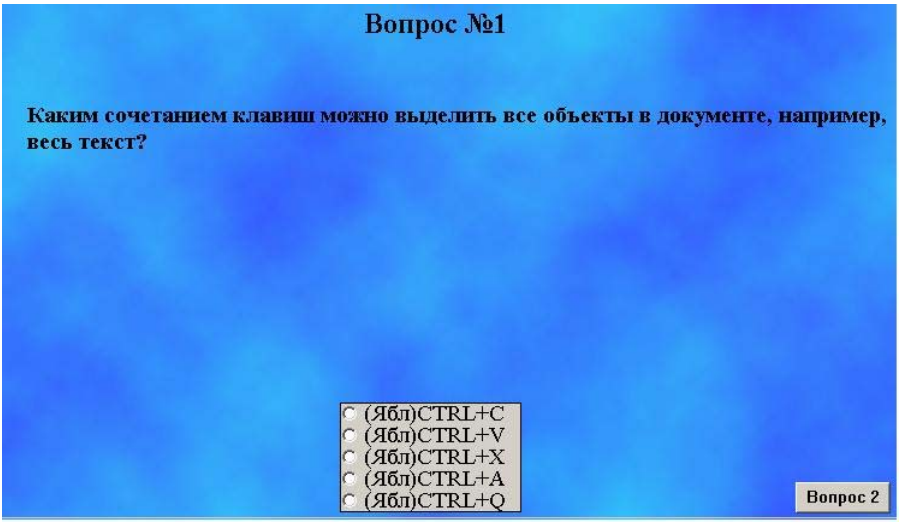


Рис. 59

Последнее – это кнопки. На первом листе нужна только кнопка перехода ко второму вопросу, на других – кнопки перехода к следующему и к предыдущему вопросам. Инструкции в кнопках – имена листов (*вопрос2*, *вопрос3* и т.д.)

Снова копируем лист. Изменяем имя переключателя на *n2*, варианты ответов, номер вопроса, текст вопроса, название и инструкции в кнопках. Таким образом, создадим пока три вопроса и итоговой лист. На последнем листе с вопросом вместо кнопки перехода к следующему вопросу сделаем кнопку «Закончить тест». В этой кнопке должна вызываться процедура очистки текстовых полей (*очисти_текст*) и сделан переход к итоговому листу (*итоги*). На итоговом листе делаем следующее – поле «*имя*», поля, в которые мы будем передавать значения переключателей (номера ответов на вопросы), поле для оценки и кнопка «Покажи оценку».

Рис. 60

Сразу изменим имена. Поле, куда будет передаваться имя тестируемого, так и назовём «*имя*». Имена полей с номерами ответов назовём соответственно «*отв1*», «*отв2*», … Поле с оценкой назовём «*оценка*». В инструкциях кнопки «Покажи оценку» будем вызывать процедуру вычисления и вывода оценки (*сч_оценку*).

В зависимости от того правильный сделан выбор ответа или нет, сделаем так чтобы в соответствующем текстовом поле цвет текста был зелёный, если выбор сделан верно и красный – в противном случае.

Ниже приведены процедуры, которые нужно написать в правой части экрана на вкладке «Процедуры» в порядке их следования.

```
это очисти_текст
    имя, ст
    оценка, ст
    отв1,  ст
    отв2,  ст
    отв3,  ст
конец
```

```
это очисти_переключатели
    установи_п1  0
    установи_п2  0
    установи_п3  0
конец

это startup
    спроси [Как ваше имя?]
    сообщи пред [Привет,] ответ
    очисти_переключатели
    Введение
конец

это запись
    имя, нрш 14 нцт 105 пиши ответ
    если_иначе  п1 =  4 [отв1,  нцт 65][отв1,  нцт 15] отв1, пиши п1
    если_иначе  п2 =  1 [отв2,  нцт 65][отв2,  нцт 15] отв2, пиши п2
    если_иначе  п3 =  5 [отв3,  нцт 65][отв3,  нцт 15] отв3, пиши п3
конец

это сч_оценку
    пусть "балл 0
    если  п1 = 4 [пусть "балл :балл + 1]
    если  п2 = 1 [пусть "балл :балл + 1]
    если  п3 = 5 [пусть "балл :балл + 1]
    пусть "балл округли :балл / 3
    оценка, нрш 40 нцт 25 ст пиши :балл
конец
```

Первые три процедуры в дополнительных пояснениях не нуждаются. Находящиеся там команды разобраны в пунктах 10.3, 10.4. В процедуре «*запись*» всё тоже достаточно прозрачно. В поле «*имя*» мы меняем размер (*нрш – нов_размер_шрифта*) и цвет (*нцт – нов_цвет_текста*) шрифта, после чего заносим в это поле имя, введённое нами в самом начале в окне «Как ваше имя?», и хранящееся в переменной «*ответ*». Дальше в зависимости от выбора в переключателе мы меняем цвет шрифта и передаём в соответствующее текстовое поле на итоговом листе значение выбора.

В последней процедуре «*сч_оценку*» задаётся переменная «*балл*», которую мы изначально обнуляем. Затем при каждом правильном выборе в соответствующем переключателе мы прибавляем в переменную «*балл*» единицу. В конце вычисляется оценка. В полноценном тесте будет 20 вопросов. Поэтому сумму накопленных баллов мы будем делить на 4, а потом округлять. Самая последняя строка в этой процедуре ответственна за вывод вычисленной оценки в текстовое поле «*оценка*».

Отладьте тест так, чтобы все процедуры работали корректно. После этого можно последовательно вносить в проект оставшиеся вопросы, добавляя в процедуры необходимые строки по аналогии уже имеющимися.

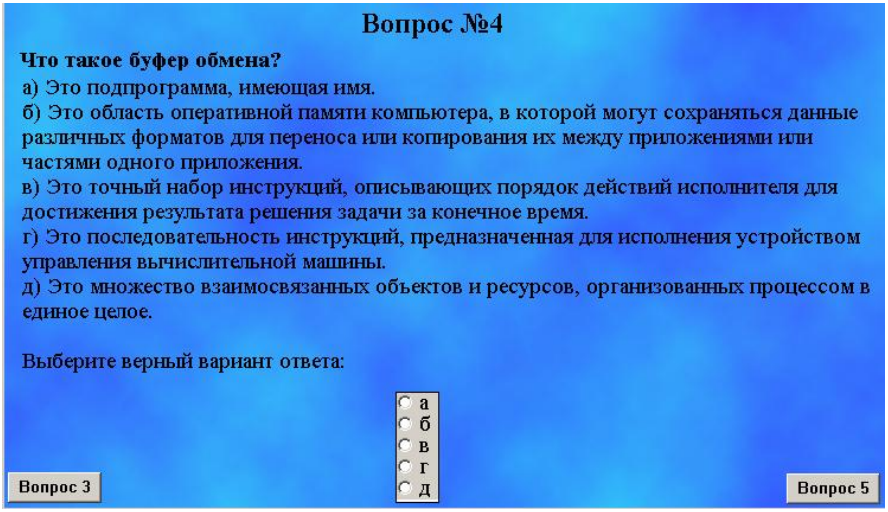


Рис. 61

Если ответ не помещается в переключатель, как, например, в 4 вопросе (рис. 59), можно поступить следующим образом: поместить варианты ответов в текстовое поле под буквами а), б), в), г), д), а в переключатель внести только буквы ответов.

Приложение
(Вопросы с вариантами ответов для итогового теста)

1 вариант

1. С помощью какого сочетания клавиш можно скопировать выделенные объекты в буфер обмена?

а) CTRL(Ябл)+X
б) CTRL(Ябл)+C
в) CTRL(Ябл)+S
г) CTRL(Ябл)+Q
д) CTRL(Ябл)+V

2. С помощью какого сочетания клавиш можно вставить скопированные объекты из буфера обмена?

а) CTRL(Ябл)+C
б) CTRL(Ябл)+V
в) CTRL(Ябл)+X
г) CTRL(Ябл)+Q
д) CTRL(Ябл)+S

3. С помощью какого сочетания клавиш можно завершить работу приложения в ОС MacOS X?

а) Ябл+C
б) Ябл+V
в) Ябл+X
г) Ябл+Q
д) Ябл+S

4. Что такое буфер обмена?

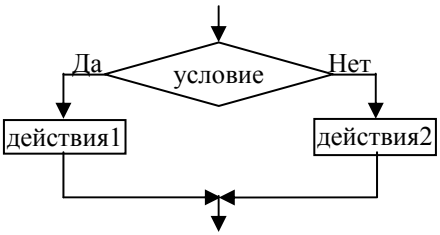
а) Это подпрограмма, имеющая имя.
б) Это область оперативной памяти компьютера, в которой могут сохраняться данные различных форматов для переноса или копирования их между приложениями или частями одного приложения.
в) Это точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время.

г) Это последовательность инструкций, предназначенная для исполнения устройством управления вычислительной машины.
д) Это множество взаимосвязанных объектов и ресурсов, организованных процессом в единое целое.

5. Что такое процедура?
а) Это точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время.
б) Это область оперативной памяти компьютера, в которой могут сохраняться данные различных форматов для переноса или копирования их между приложениями или частями одного приложения.
в) Это множество взаимосвязанных объектов и ресурсов, организованных процессом в единое целое.
г) Это последовательность инструкций, предназначенная для исполнения устройством управления вычислительной машины.
д) Это подпрограмма, имеющая имя.

6. Какие существуют способы записи алгоритмов?
а) Лингвистический, математический, программный.
б) С предусловием, с постусловием, с параметром.
в) Рациональный, нерациональный.
г) Словесный, блок-схема, программа.
д) Линейный, разветвляющийся, циклический.

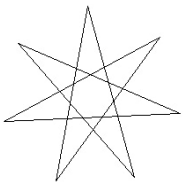
7. Какой вид алгоритма изображён на рисунке?
а) Линейный.
б) Циклический.
в) Нерациональный.
г) Рациональный.
д) Разветвляющийся.



8. При выполнении какой программы черепашка нарисует квадрат со стороной 50 точек?

а) повтори 4 [лв 90 нд 50]
б) повтори 4 [пр 90]
в) повтори 0 [вп 50 пр 90]
г) повтори 1 [вп 50 лв 90]
д) повтори 8 [вп 25 пр 90]

9. При выполнении каких команд черепашка нарисует семиконечную звезду, показанную на рисунке?

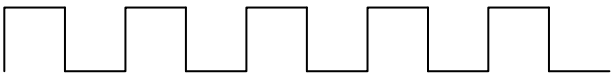


а) по повтори 7 [вп 100 пр 360 * 2 / 7]
б) по повтори 7 [вп 100 пр 360 * 8 / 7]
в) по повтори 7 [вп 100 пр 360 * 4 / 3]
г) по повтори 7 [вп 100 пр 360 * 4 / 7]
д) по повтори 7 [вп 100 пр 360 / 7]

10. При выполнении каких команд черепашка нарисует узор, показанный на рисунке?

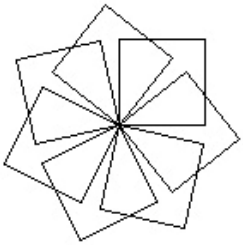
а) по повтори 5 [повтори 3 [вп 40 пр 90] лв 180 вп 40 лв 90]
б) по повтори 5 [вп 50 пр 135 вп 25 лв 90 вп 25 пр 135 вп 50 лв 90 вп 20 лв 90]
в) по пр 45 повтори 5 [вп 50 пр 90 вп 50 лв 90]
г) по пр 90 повтори 5 [вп 50 лв 90 вп 20 пр 90]
д) по повтори 5 [вп 50 нд 50 пр 90 вп 50 лв 90]

11. При выполнении каких команд черепашка нарисует узор, показанный на рисунке?



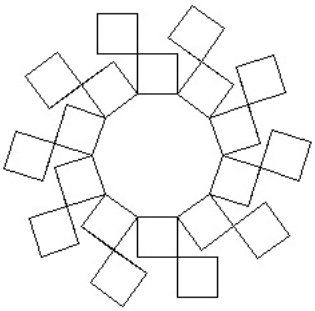
а) по пр 90 повтори 5 [вп 50 лв 90 вп 20 пр 90]
б) по повтори 5 [вп 50 нд 50 пр 90 вп 50 лв 90]
в) по пр 45 повтори 5 [вп 50 пр 90 вп 50 лв 90]
г) по повтори 5 [повтори 3 [вп 40 пр 90] лв 180 вп 40 лв 90]
д) по повтори 5 [вп 50 пр 135 вп 25 лв 90 вп 25 пр 135 вп 50 лв 90 вп 20 лв 90]

12. При выполнении какой программы черепашка нарисует узор, показанный на рисунке (в правилах написано имя «узор_из_кв»)?



а)	б)	в)	г)	д)
это квадрат повтори 4 [вп 30 пр 90] конец это узор_из_кв по повтори 8 [квадрат вп 30 пр 90 вп 30 лв 45] конец	это квадрат повтори 4 [вп 30 пр 90] конец это узор_из_кв по повтори 7 [квадрат пр 360 / 7] конец	это узор_из_кв по повтори 7 [вп 30 пр 360 * 4 / 7] конец	это квадрат повтори 4 [вп 30 пр 90] конец это узор_из_кв по повтори 10 [квадрат вп 30 лв 90 вп 30 пр 90 квадрат лв 90 нд 60 пр 90 нд 30 пр 36] конец	это квадрат повтори 5 [вп 30 пр 72] конец это узор_из_кв по повтори 7 [квадрат пр 360 / 7] конец

13. При выполнении какой программы черепашка нарисует узор, показанный на рисунке (в правилах написано имя «узор_из_кв»)?



а)	б)	в)	г)	д)
это квадрат повтори 4 [вп 50 пр 90] конец это узор_из_кв по повтори 7 [квадрат пр 360 / 7] конец	это узор_из_кв по повтори 7 [вп 50 пр 360 * 4 / 7] конец	это квадрат повтори 5 [вп 50 пр 72] конец это узор_из_кв по повтори 7 [квадрат пр 360 / 7] конец	это квадрат повтори 4 [вп 30 пр 90] конец это узор_из_кв по повтори 10 [квадрат вп 30 лв 90 вп 30 пр 90 квадрат лв 90 нд 60 пр 90 нд 30 пр 36] конец	это квадрат повтори 4 [вп 30 пр 90] конец это узор_из_кв по повтори 8 [квадрат вп 30 пр 90 вп 30 лв 45] конец

14. Какая из приведённых ниже команд изменяет цвет черепашки?

а) нц
б) цп
в) нк
г) нрп
д) нрз

15. Какая из приведённых ниже команд изменяет размер черепашки?

а) нк
б) нц
в) нрп
г) цп
д) нрз

16. Как задать список команд таким образом, чтобы их выполнила только черепашка с именем ч1 на текущем листе?

а) список команд, ч1
б) список команд, "ч1
в) "ч1, список команд
г) ч1 - список команд
д) ч1, список команд

17. Чтобы команды одновременно выполнили все черепашки на текущем листе, надо написать так:

а) ч1, ч2, ..., чn [список команд]
б) список команд - каждая
в) список команд
г) все [список команд]
д) каждая [список команд]

18. Условие касания двух черепашек (с использованием условного оператора 1-го типа) выглядит следующим образом:

а) если коснулись? ч1 ч2 [список команд]
б) если коснулись? "ч1" "ч2" [список команд]
в) если коснулись "ч1 "ч2 [список команд]
г) если коснулись? "ч1 "ч2 [список команд]
д) если коснулись ч1 ч2 [список команд]

19. При выполнении каких команд черепашка будет изменять свой размер с течением времени, не выходя за диапазон допустимых значений размера?

а) нрз 5 повтори 155 [размер + 1]
б) нрз 1 повтори 159 [нрз размер + 1]
в) нрз 5 повтори 155 [нрз + 1]
г) нрз 5 повтори 155 [нрз размер + 1]
д) нрз 5 повтори 160 [нрз размер + 1]

20. Черепашка имеет текущую форму квадрата со стороной 20 точек. Для того чтобы черепашка нарисовала сетку из таких квадратов размером 10х10, ей необходимо задать следующие команды:

а) нк 90 нм [-150 150] повтори 10 [повтори 10 [штамп вп 20] нов_х -150 нов_у у_коор - 20]
б) нк 90 нм [-150 150] повтори 10 [штамп вп 20] нов_х -150 нов_у у_коор - 20
в) нк 90 нм [-150 150] повтори 10 [повтори 10 [штамп нов_х -150 нов_у у_коор - 20]]
г) нк 90 нм [-150 150] повтори 10 [повтори 10 [штамп вп 20 пр 90] нов_х -150 нов_у у_коор - 20]
д) нк 90 нм [-150 150] повтори 10 [повтори 10 [штамп вп 20] нов_х -150 нов_у -20]

2 вариант

1. С помощью какого сочетания клавиш можно выделить все объекты в текущем документе (например, текст в текстовом редакторе)?

а) CTRL(Ябл)+C
б) CTRL(Ябл)+S
в) CTRL(Ябл)+X
г) CTRL(Ябл)+A
д) CTRL(Ябл)+Q

2. С помощью какого сочетания клавиш можно вырезать выделенные объекты в буфер обмена?

а) CTRL(Ябл)+C
б) CTRL(Ябл)+S
в) CTRL(Ябл)+X
г) CTRL(Ябл)+Q
д) CTRL(Ябл)+A

3. С помощью какого сочетания клавиш можно сохранить текущие изменения документа в ОС MacOS X?

а) Ябл+C
б) Ябл+S
в) Ябл+X
г) Ябл+Q
д) Ябл+A

4. Что такое файл?

а) Это часть системы памяти ЭВМ, в которую процессор может обратиться за одну операцию.
б) Это носитель информации, предназначенный для записи и хранения данных.
в) Это часть вычислительной машины, физическое устройство или среда для хранения данных, используемых в вычислениях, в течение определённого времени.
г) Это поименованная совокупность байтов на носителе информации, содержащая название подкаталогов и файлов.
д) Это упорядоченная совокупность данных, хранимая на диске и занимающая поименованную область внешней памяти.

5. Что такое алгоритм?

а) Это множество взаимосвязанных объектов и ресурсов, организованных процессом в единое целое.

б) Это область оперативной памяти компьютера, в которой могут сохраняться данные различных форматов для переноса или копирования их между приложениями или частями одного приложения.

в) Это подпрограмма, имеющая имя.

г) Это последовательность инструкций, предназначенная для исполнения устройством управления вычислительной машины.

д) Это точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время.

6. Какие существуют виды алгоритмов?

а) С предусловием, с постусловием, с параметром.

б) Лингвистический, математический, программный.

в) Рациональный, нерациональный.

г) Линейный, разветвляющийся, циклический.

д) Словесный, блок-схема, программа.

7. Какой вид алгоритма изображён на рисунке?

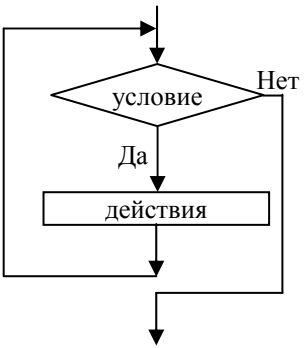
а) Циклический

б) Разветвляющийся

в) Нерациональный

г) Линейный

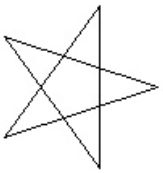
д) Рациональный



8. При выполнении какой программы черепашка нарисует треугольник со стороной 50 точек?

а) повтори 1 [вп 50 лв 120]
б) повтори 3 [пр 90]
в) повтори 0 [вп 50 пр 120]
г) повтори 3 [лв 120 нд 50]
д) повтори 6 [вп 25 пр 60]

9. При выполнении каких команд черепашка нарисует пятиконечную звезду, показанную на рисунке?



а) по повтори 5 [вп 100 пр 360 * 2 / 5]
б) по повтори 5 [вп 100 пр 360 / 5]
в) по повтори 5 [вп 100 пр 360 * 4 / 5]
г) по повтори 5 [вп 100 пр 360 * 8 / 5]
д) по повтори 5 [вп 100 пр 360 * 5]

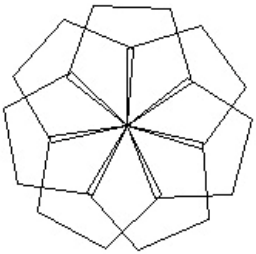
10. При выполнении каких команд черепашка нарисует узор, показанный на рисунке?

а) по повтори 5 [вп 50 нд 50 пр 90 вп 50 лв 90]
б) по пр 90 повтори 5 [вп 50 лв 90 вп 20 пр 90]
в) по повтори 5 [вп 50 пр 135 вп 25 лв 90 вп 25 пр 135 вп 50 лв 90 вп 20 лв 90]
г) по повтори 5 [повтори 3 [вп 40 пр 90] лв 180 вп 40 лв 90]
д) по пр 45 повтори 5 [вп 50 пр 90 вп 50 лв 90]

11. При выполнении каких команд черепашка нарисует узор, показанный на рисунке?

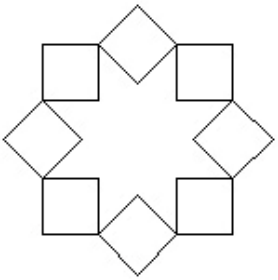
а) по пр 90 повтори 5 [вп 50 лв 90 вп 20 пр 90]
б) по повтори 5 [вп 50 нд 50 пр 90 вп 50 лв 90]
в) по пр 45 повтори 5 [вп 50 пр 90 вп 50 лв 90]
г) по повтори 5 [повтори 3 [вп 40 пр 90] лв 180 вп 40 лв 90]
д) по повтори 5 [вп 50 пр 135 вп 25 лв 90 вп 25 пр 135 вп 50 лв 90 вп 20 лв 90]

12. При выполнении какой программы черепашка нарисует узор, показанный на рисунке (в правилах написано имя процедуры «узор»)?



а)	б)	в)	г)	д)
это пятиугольник повтори 5 [вп 30 пр 360 / 5] конец это узор по повтори 8 [пятиугольник вп 30 пр 90 вп 30 лв 45] конец	это пятиугольник повтори 5 [вп 30 пр 90] конец это узор по повтори 7 [пятиугольник пр 360 / 7] конец	это узор по повтори 7 [вп 30 пр 360 * 4 / 7] конец	это пятиугольник повтори 5 [вп 30 пр 360 / 5] конец это узор по повтори 10 [пятиугольник вп 30 лв 90 вп 30 пр 90 пятиугольник лв 90 нд 60 пр 90 нд 30 пр 36] конец	это пятиугольник повтори 5 [вп 30 пр 72] конец это узор по повтори 7 [пятиугольник пр 360 / 7] конец

13. При выполнении какой программы черепашка нарисует узор, показанный на рисунке (в правилах написано имя процедуры «узор_из_кв»)?



а)	б)	в)	г)	д)
это квадрат повтори 4 [вп 50 пр 90] конец это узор_из_кв по повтори 8 [квадрат вп 30 пр 90 вп 30 лв 45] конец	это квадрат повтори 4 [вп 50 пр 90] конец это узор_из_кв по по повтори 7 [квадрат пр 360 / 7] конец	это квадрат повтори 5 [вп 50 пр 72] конец это узор_из_кв по повтори 7 [квадрат пр 360 / 7] конец	это квадрат повтори 4 [вп 50 пр 90] конец это узор_из_кв по повтори 10 [квадрат вп 30 лв 90 вп 30 пр 90 квадрат лв 90 нд 60 пр 90 нд 30 пр 36] конец	это узор_из_кв по повтори 7 [вп 50 пр 360 * 4 / 7] конец

14. Какая из приведённых ниже команд изменяет направление движения черепашки?

а) цп
б) нрп
в) нрз
г) нц
д) нк

15. Какая из приведённых ниже команд изменяет размер пера черепашки?

а) нк
б) нрз
в) нрп
г) нц
д) цп

16. Экран имеет прямоугольную форму размерами 720х340 точек. Центр экрана - начало координат. Черепашка имеет текущую форму шарика и летит по курсу 45. Какое условие нужно проверить черепашке, для того чтобы отскочить от верхней границы экрана?

а) у_коор > 170
б) х_коор > 360
в) у_коор < -170
г) х_коор < -360
д) х_коор > -360

17. Задана переменная «число», значение которой равно 75. Какой результат мы увидим на экране после выполнения следующих команд:

```
пусть "число (:число - 15) / 3
покажи :число ?
```

а) 10
б) 20
в) 75
г) 30
д) 70

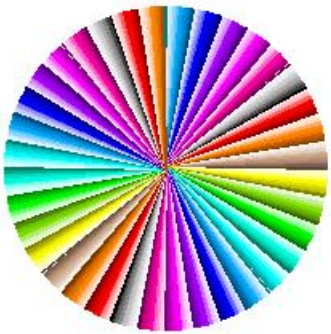
18. Черепашка должна нарисовать круг, граница которого красная размером 3 точки, а внутренняя область закрашена жёлтым. Какие команды ей нужно для этого выполнить?

а) нрп 3 нц 15 по повтори 180 [вп 2 пр 2] лв 90 пп вп 10 нц 45 крась
б) нрп 3 нц 45 по повтори 180 [вп 2 пр 1] пр 90 пп вп 10 нц 15 крась
в) нрп 3 нц 15 по повтори 180 [вп 2 пр 1] пр 90 пп вп 10 нц 45 крась
г) нрп 3 нц 45 по повтори 180 [вп 2 пр 2] пр 90 пп вп 10 нц 15 крась
д) нрп 3 нц 15 по повтори 180 [вп 2 пр 2] пр 90 пп вп 10 нц 45 крась

19. Черепашка имеет текущую форму закрашенного прямоугольника, размер которого 20x50 точек. При выполнении каких команд она нарисует в верхней части экрана горизонтальный ряд из 12 таких прямоугольников?

а) нм [-350 170] нк 0 повтори 12 [штамп вп 60]
б) нм [-350 -170] нк 90 повтори 12 [штамп вп 50]
в) нм [-350 170] нк 0 повтори 12 [штамп вп 20]
г) нм [-350 -170] нк 90 повтори 12 [штамп вп 60]
д) нм [-350 170] нк 90 повтори 12 [штамп вп 60]

20. Для того чтобы черепашка нарисовала цветовой круг, показанный на рисунке, она должна выполнить следующие команды:



а) нк 0 по нрп 2 повтори 360 [вп 100 нд 100 нц + 1]
б) нк 0 по нрп 2 повтори 360 [вп 100 нд 100 нц цвет + 1]
в) нк 0 по нрп 2 повтори 360 [вп 100 нд 100 пр 1 нц цвет + 1]
г) нк 0 по нрп 2 повтори 360 [вп 100 нд 100 пр 1 цвет + 1]
д) нк 0 по нрп 2 повтори 360 [вп 100 нд 100 пр 1 лв 1 нц цвет + 1]

Список литературы

1. Информатика и ИКТ. Учебник. Начальный уровень / Под ред. проф. Н.В. Макаровой. – СПб.: Питер, 2007. – 160 с.: ил.
2. Электронное издание. Проект «Координатная планета». Петров Сергей Михайлович.
3. Демонстрационные проекты и справка, встроенные в среду Логомиры.