Основные конструкции алгоритмов

Для работы с этим разделом вам потребуется среда Роб (программа Roo), которую вы можете установить из раздела Дистрибутивы этого диска.

Тема 1. Алгоритмы, программы и среды

В этой теме мы займемся написанием алгоритмов. Будем рассматривать:

- алгоритмы, которые решают задачи на построение и выкладывание различных узоров;
- алгоритмы, прохождения по лабиринтам.

При этом мы разберем многие приемы, которые характерны для современных алгоритмических языков.

Как мы знаем, для написания алгоритмов нам будет необходим язык, а к нему – исполнитель. Исполнителем, конечно, будет компьютер, а осуществлять перевод из нашего языка в язык компьютера будет соответствующая программа-компилятор. Как работает этот компилятор и как устроен внутренний язык компьютера, мы здесь разбирать не будем.

Но ограничиться только языком и компилятором нам будет не удобно. Составление алгоритмов, запись их на алгоритмическом языке будет осуществлять человек и создание удобств для работы человека является очень важным. Поэтому к языку и компилятору будут добавлены различные системы, создающие такие удобства. Например, система для написания текстов алгоритмов и внесения в них изменений (систему редактирования текстов).

Также необходима система для просмотра результатов работы алгоритмов (она относится к средствам визуализации, т.е. просмотра, придания наглядности алгоритмам), а также система, которая будет удобна при поиске и исправлении ошибок в программах (система отладки программ).

Все это, вместе взятое, будет называться средой программирования. То есть среда программирования – это алгоритмический язык и целый набор средств для составления и обработки алгоритмов.

Само слово программирование, конечно, происходит от слова программа, а последнее используется как синоним для выражения «алгоритм для компьютера». Также обычно говорят «язык программирования» вместо «алгоритмический язык» в случаях, когда такой язык предназначен для применения на компьютерах. Человека, пишущего программы на языке программирования, часто называют программистом.

В настоящее время почти все алгоритмические языки являются языками программирования. Для каждого такого языка обязательно создается своя среда программирования, без которой ныне любой язык программирования не получит практического применения, потому что удобство использования языка человеком ныне является основным требованиям для использования.

Тема 2. Среда Роба – Мир, в котором живет маленький робот Роб

Мы начинаем работать в среде **Роб**. Эта среда предназначена для рисования узоров на клетчатых полях различного размера и прохождению по лабиринтам.

Узоры будут составляться из линий, соединяющих центры соседних клеток. Линии будут только вертикальными и горизонтальными.

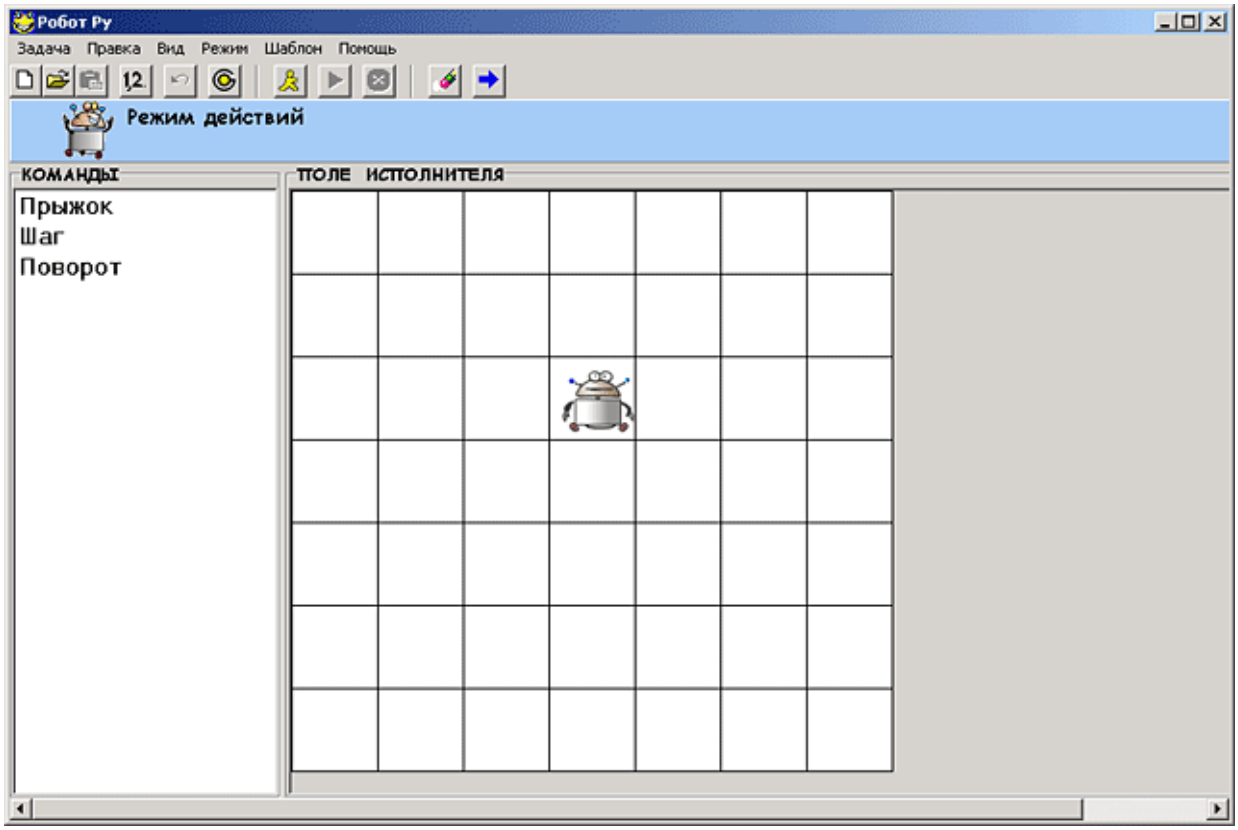
В среде Роб исполнителя алгоритмов также зовут Роб. Полностью его зовут Роб Желтоглазый, но мы его будем звать просто Роб. Это Роб рисует необходимые нам линии. Вообще-то Роб много что умеет делать, но мы используем его сейчас только для рисования. Роб ждет наши команды и исполняет их – если эти команды исполнимы.

На самом деле, конечно, исполнителем будет компьютерная система, но в целях наглядности исполнять команды написанных Вами алгоритмов на экране будет именно маленький Роб. Создавать рисунок Роб будет, прочерчивая отрезки своим собственными следами. Через промежутки, в которых рисовать ничего не надо, Роб может перепрыгивать. Поворачиваясь, Роб может выбирать направление движения.

В этой теме рассмотрим:

- Непосредственное выполнение команд
- Запись программы в среде Роб
- Выполнение программы
- Выполнение по шагам
- Создание шаблона
- Загрузка шаблона
- Сохранение программы
- Открыть готовую программу

Непосредственное выполнение команд



См. фильм "Непосредственное выполнение команд"

Какие инструкции (команды) может выполнять Роб?

Откроем среду Роб. Создадим новую задачу. Не будем загружать готовый шаблон – создадим свой, щелкув «нет». Зададим размер поля. Щелкнем OK.

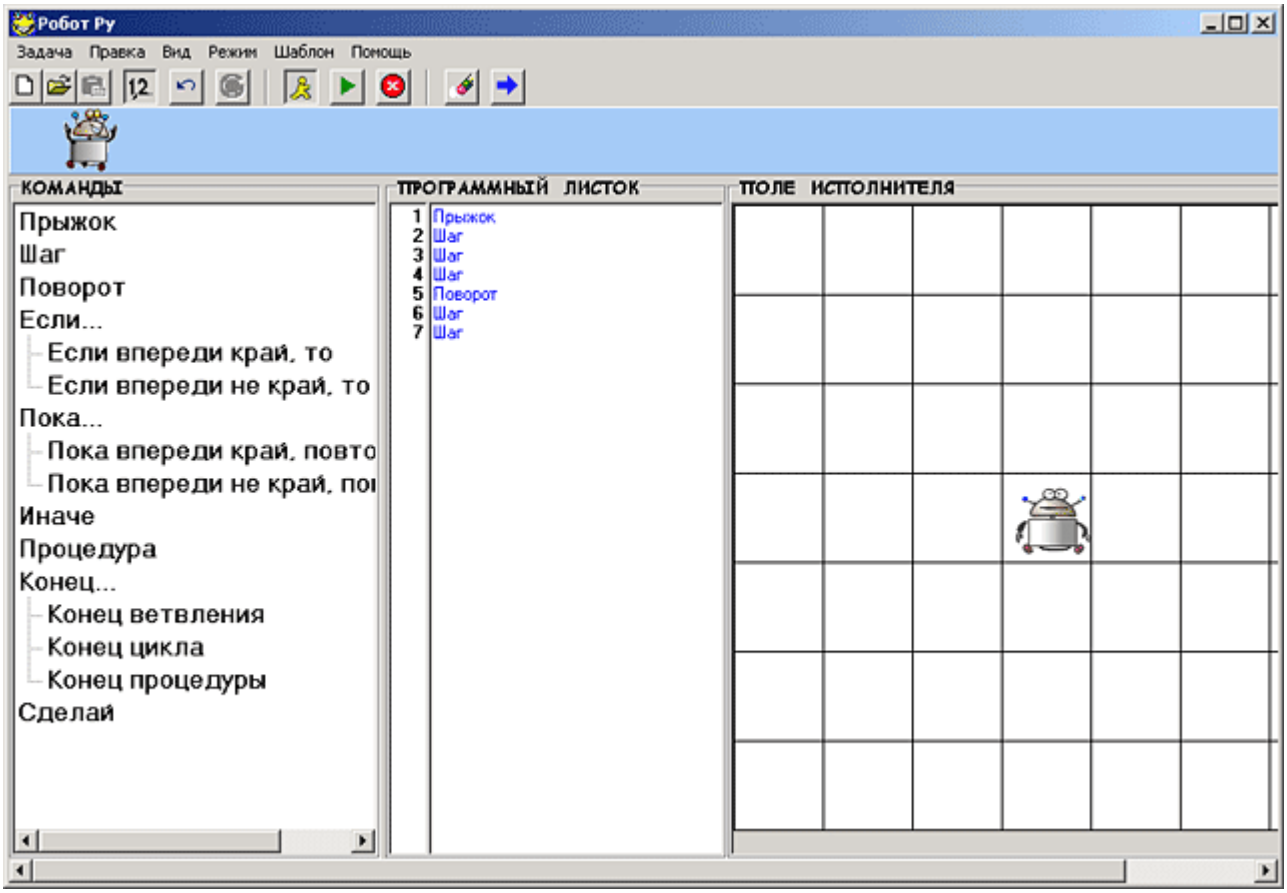
Появилось поле Роба. Установим с помощью мышки начальное положение Роба и с помощью стрелок клавиатуры – начальное направление. В завершении нажмем клавишу Enter.

Чтобы продемонстрировать команды, которые в точности по нашему предписанию будет выполнять Роб, перейдем в Режим действий. Щелкая на команды, мы видим, как их выполняет Роб.

1. **Поворот** – только по часовой стрелке.
2. **Шаг** – перемещает Роба на одну клеточку и оставляет след.
3. **Прыжок** – перемещает Роба на одну клеточку и не оставляет следа.

Если дать команду Робу шагнуть за пределы поля ...Роб в недоумении остановится и будет ждать правильную команду. Для очистки поля от узоров Роба имеется кнопка «Очистить». Для установки Роба в начальное положение имеется кнопка «Установка робота». Щелкнем эту кнопку – вместо Роба появилась стрелка. Установим ее мышкой в требуемую позицию, а направление зададим клавишами со стрелками. В завершении нажмем клавишу Enter. Роб встал в требуемое начальное положение.Потренеруйтесь с выполнением команд.

Запись программы в среде Роб



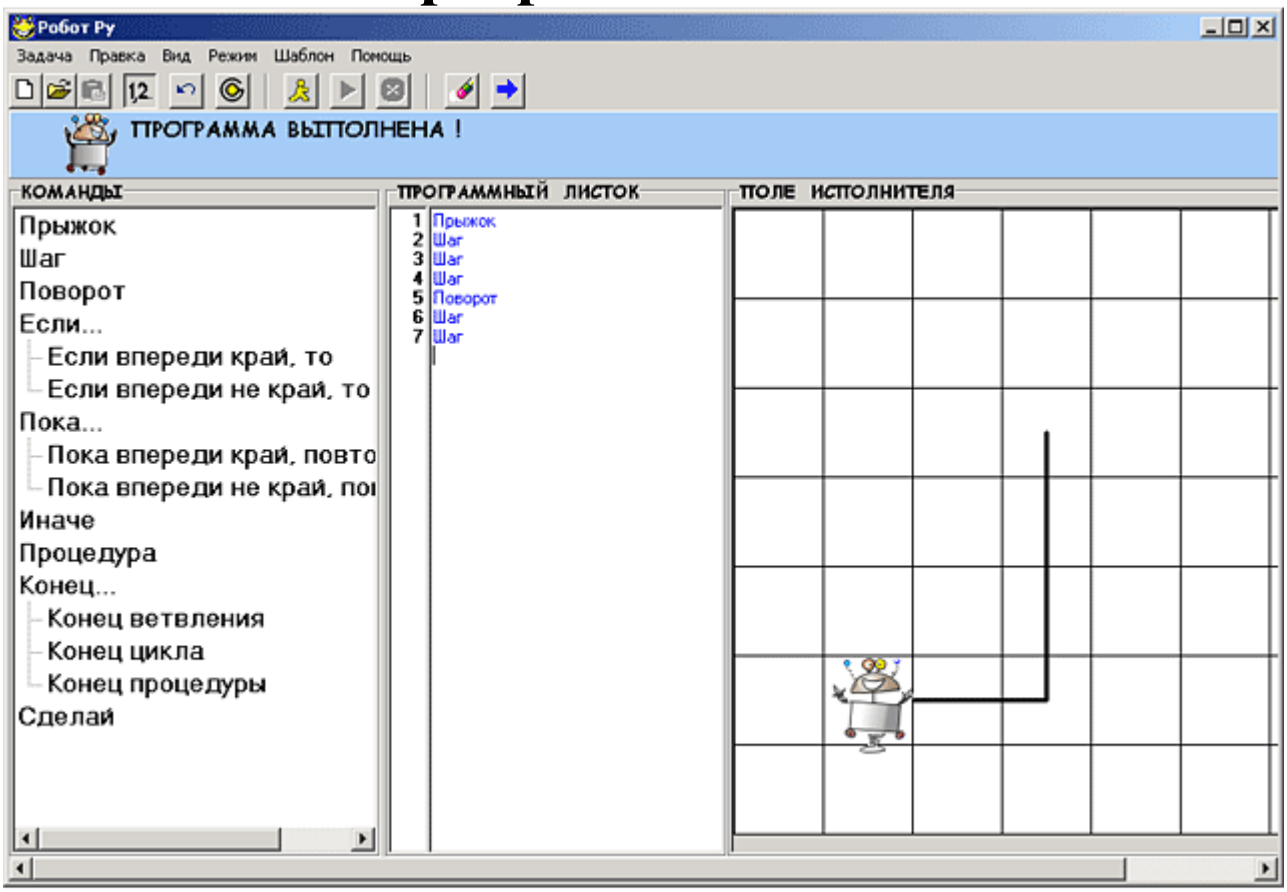
См. фильм "Запись программы в среде Роб"

Перейдем в режим выполнения. Появилось поле для ввода команд – программный листок. Поле команд расширилось – там появились команды:

- Если
- Пока
- Процедура
- Сделай.

Некоторые команды сложные, например если. Чтобы раскрыть команду, 2 раза щелкнем по слову Если. Команда раскрылась.
Чтобы занести инструкцию в программный листок, достаточно щелкнуть на нужной команде в полк «Команды».
В программном листке можно перемещать команды, удалять, вставлять новые.
Программа – это обычный текст. Ее можно скопировать в Блокнот и записать на диск. Затем ее можно вставить из блокнота.

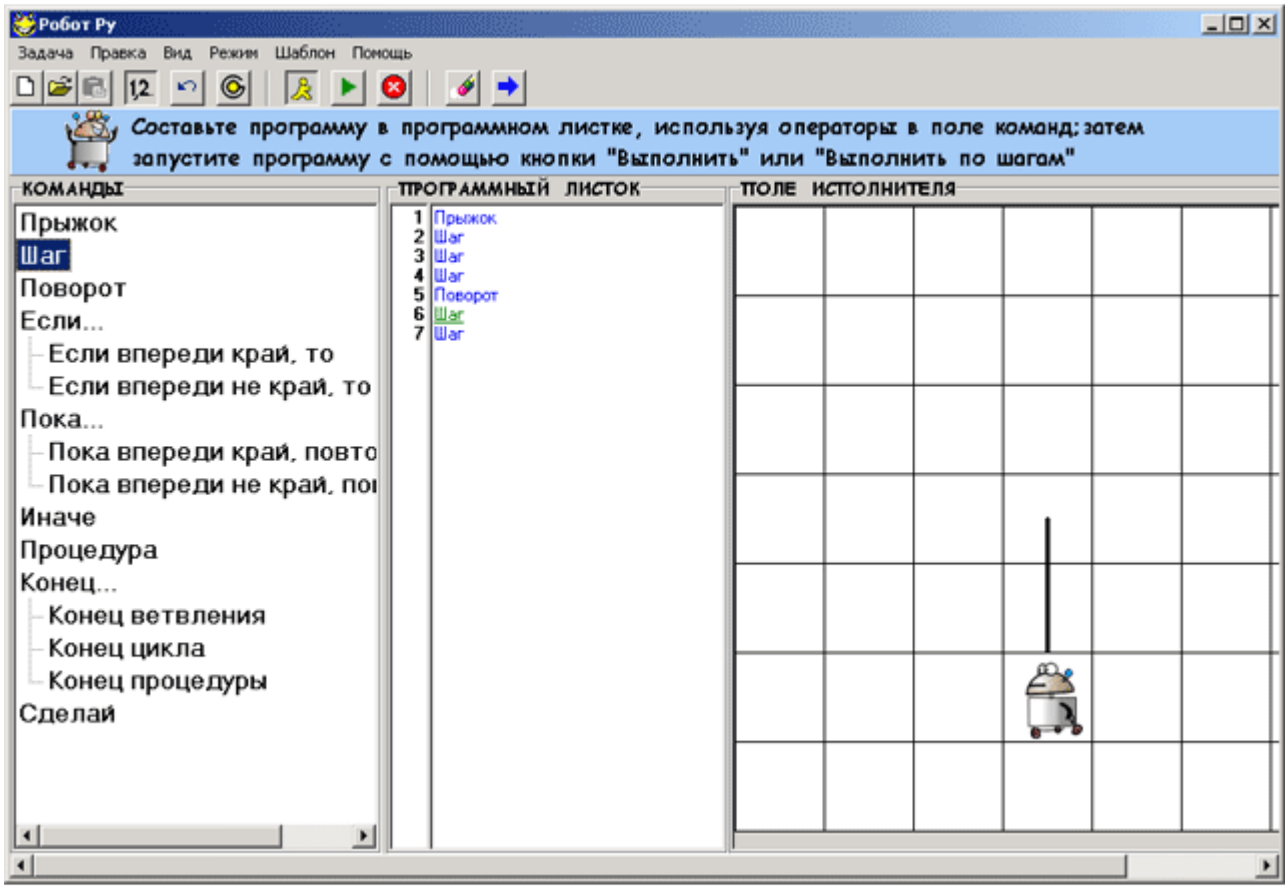
Выполнение программы



См. фильм "Выполнение программы"

После того, как программа записана, ее можно исполнить. Для этого щелкнем кнопку «Выполнить».

Выполнение по шагам

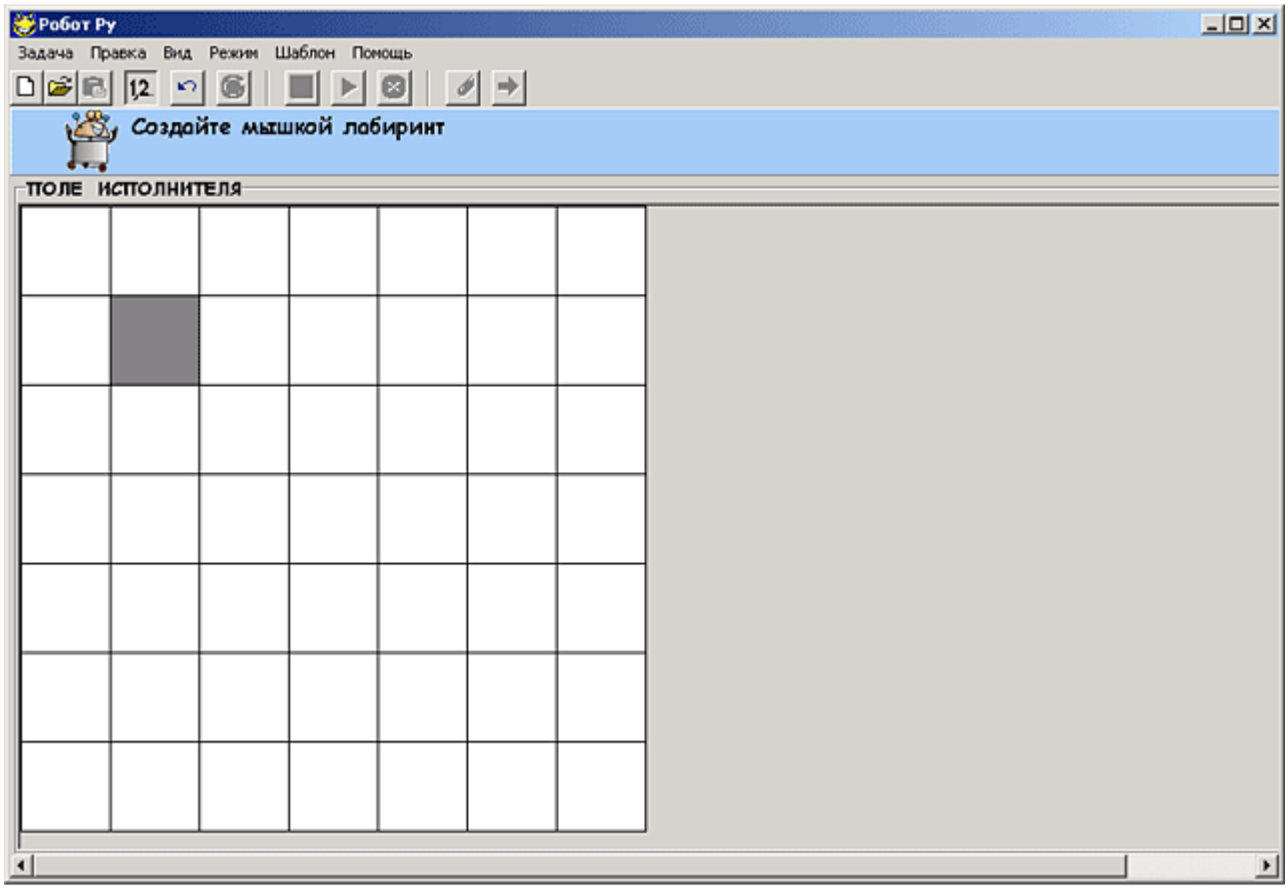


См. фильм "Выполнение по шагам"

Программу можно выполнить «по шагам». Для этого надо щелкнуть кнопку "Выполнить по шагам". Активизируется кнопка «Выполнить шаг». На каждый щелчок по этой кнопке Роб выполняет одну команду, причем сама эта команда выделяется в Командном листке.

Кнопка Стоп служит для принудительной остановки программы, например, если программа зациклится.

Создание шаблона

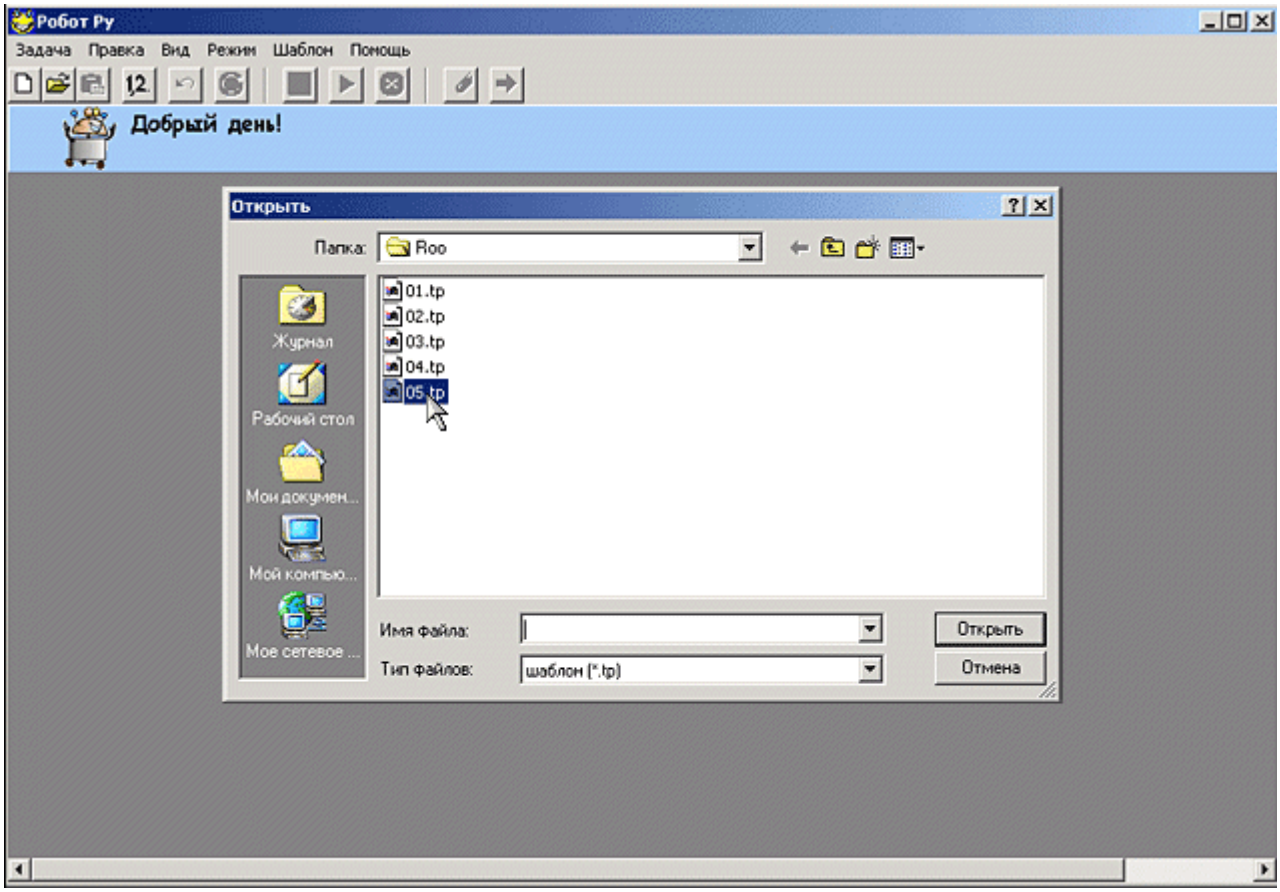


См. фильм "Создание шаблона"

Вы можете создать свой собственный шаблон поля, например, лабиринт, по которому будет перемещаться Роб.

Для этого выберем в меню Шаблон пункт Создать. Зададим размеры поля. Нарисуем лабиринт. Сохраним шаблое, исползуя команду «Сохранить».

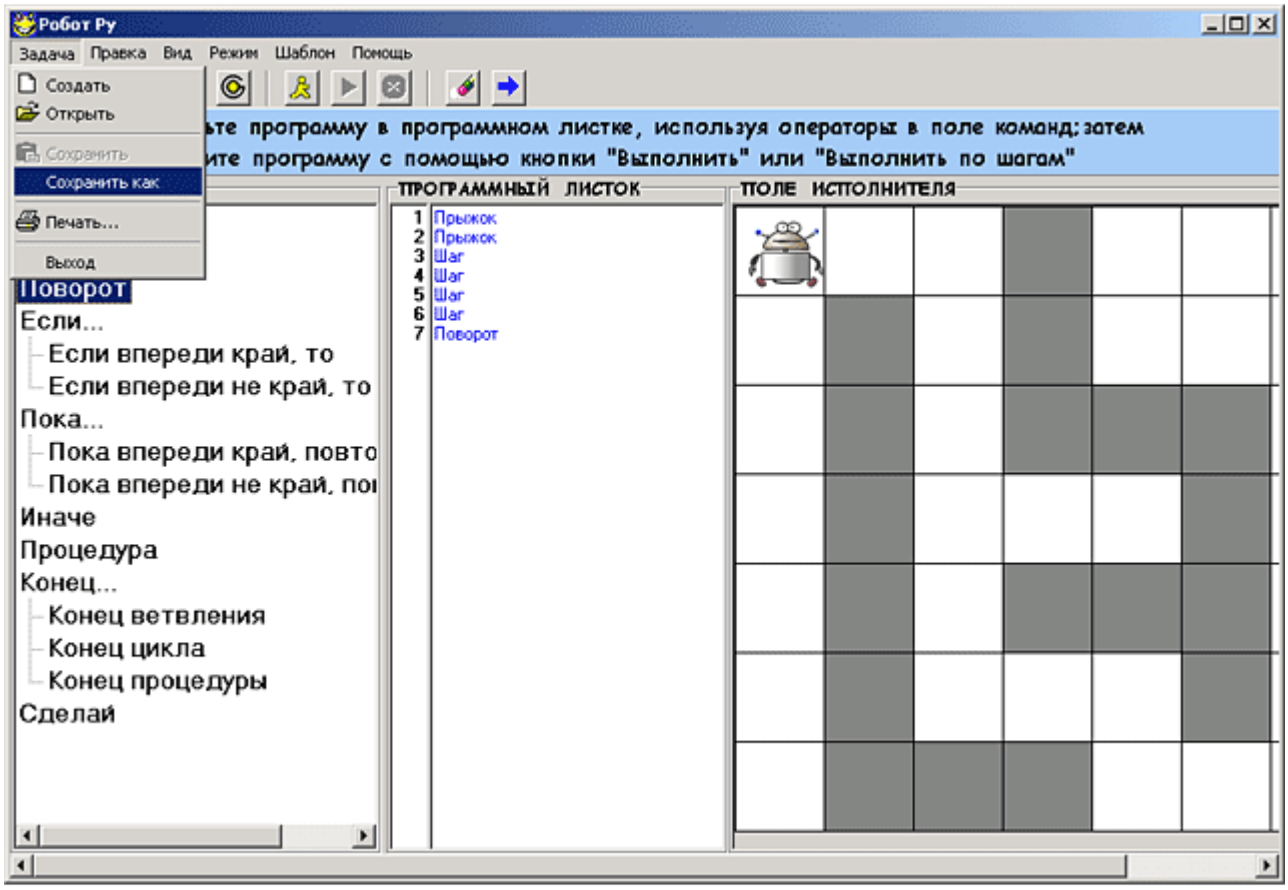
Загрузка шаблона



См. фильм "Загрузка шаблона"

Созданный шаблон можно использовать в задачах. Для этого надо щелкнуть кнопку Создать новую задачу. На запрос «Выбрать готовый шаблон поля?» щелкнуть «Да».
Откроется окно доступных шаблонов.
Выбрать шаблон.
С помощью мышки установим начальное положение Роба. А с помощью клавиш – направление.
В завершение нажмем клавишу Enter.

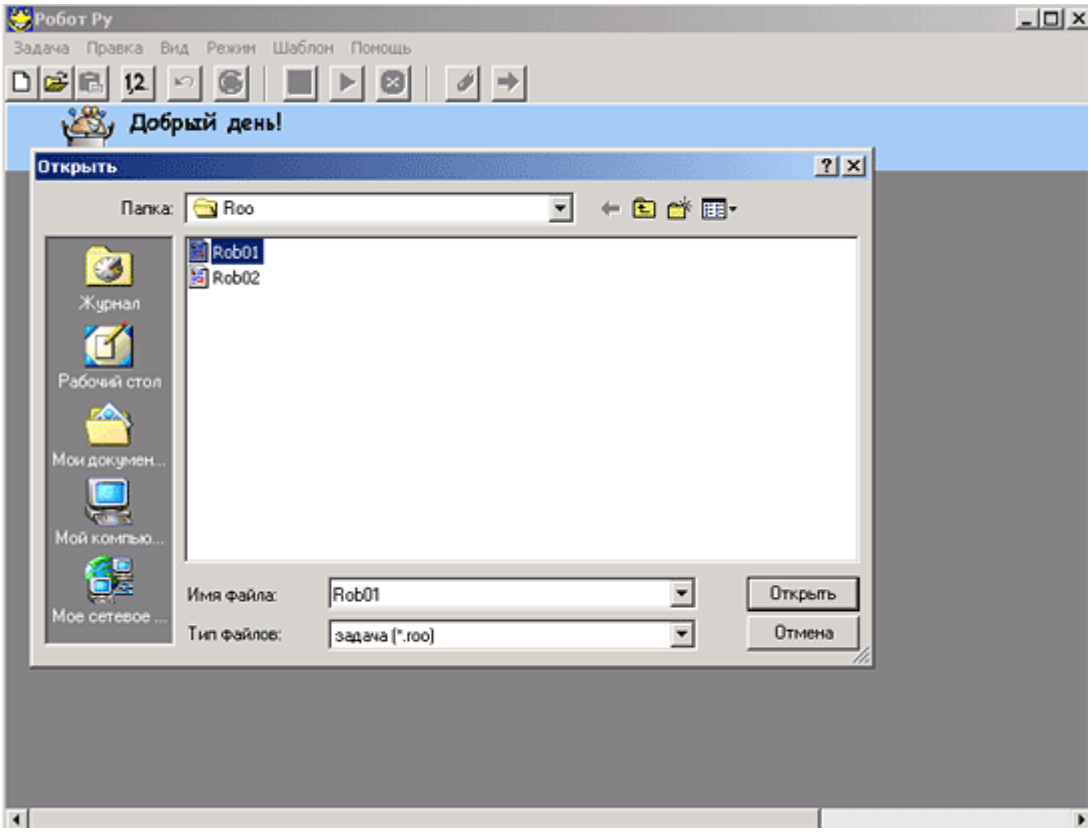
Сохранение программы



См. фильм "Сохранение задачи"

Написанную программу можно сохранить. Для этого выберем в меню «Задача» пункт «Сохранить как...». В появившемся окне дадим имя программе и щелкнем сохранить. Программа будет сохранена, включая текст программы, шаблон и начальное положение Роба.

Открыть готовую программу



См. фильм "Открыть готовую задачу"

Чтобы открыть сохраненную ранее программу, надо выбрать в меню «Задача» пункт «открыть»

В появившемся меню выбрать подходящую программу. Щелнуть «Открыть».

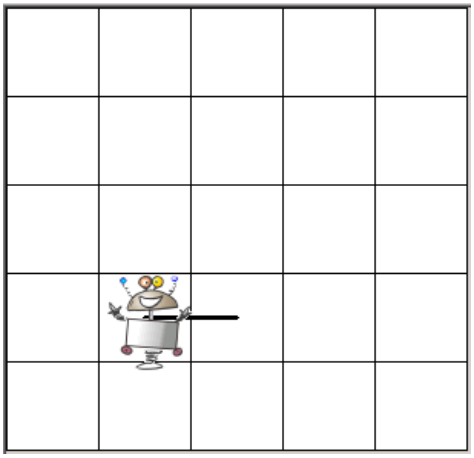
Тема 3. Программы и линейные программы

Напомним, что алгоритмы для Роба мы будем называть программами. Как мы знаем, для исполнителя алгоритма, конечно, необходимо указать, в каком порядке должны выполняться команды программы. Правило здесь обычное: команды программы исполняются по порядку, т.е. в том порядке, в котором они записаны. У нас порядок записи команд — "сверху вниз”.

Пример 1

Запустите среду Roo и откройте задачу Ro1. Или скопируйте в программный листок текст программы Ro1:

Шаг
Поворот
Поворот
Шаг



При выполнении этой программы Роб (пусть он находится в середине поля и смотрит вправо) шагает вправо, поворачивается вверх, поворачивается налево, шагает влево:

Программы, состоящие только из простых команд (шаг, прыжок и поворот), будем называть линейными. Линейные программы составляются тогда, когда в алгоритме нет повторяющихся элементов (или таких мало) или когда в ней нет действий, которые зависят от условий.

Выполняя линейную программу, Роб делает ровно столько действий, сколько в программе команд (строк). Например, линейной является программа из примера RO1.

Задачи в этом занятии все простые и могут быть без труда записаны с использованием только инструкций шаг и поворот. Обратим внимание на то, что при написании программы надо сперва придумать алгоритм, который бы решал задачу. Этот алгоритм можно записать даже «своими словами». И уже потом переписать этот алгоритм в инструкциях для Ру. Как правило, не надо пытаться писать программу сразу в языке программирования!

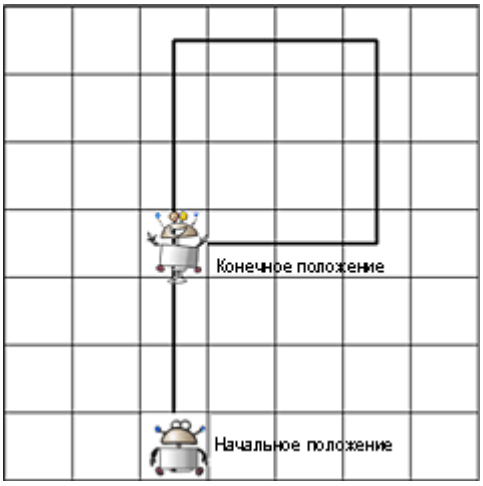
Обратите внимание на название в условии задачи, например, RO5A — так именуется задача в памяти компьютера.

Задачи к Теме 3

Задача Ro2

Загрузите Шаблон "01tp".

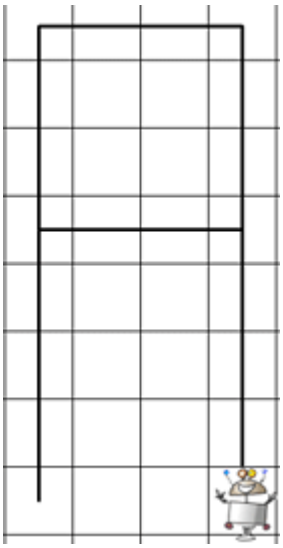
Напишите программу для Роба. Пусть Роб нарисует букву Р:



Задача Ro3

Загрузите шаблон "02.tp"

Напишите программу для Роба. Пусть Роб нарисует букву А:



Задача Ro4

Загрузите шаблон "02.tp"

Напишите программу, по которой Ру рисует буквы:

1. Ж
2. Л
3. У
4. Ф

Роба можно научить писать и слова. Составляя сложные программы, следует разбивать их на более простые части, в том числе те, писать которые вы уже умеете, а затем "собирать" из приготовленных таким образом кусков всю программу.

Задача Ro9

Напишите программу, под управлением которой Роб напишет слова:

1. РОЗА
2. ЛЕНА
3. ТАРАС
4. БАРАН

Тема 4. Условия и условный оператор

Мы знаем, что Роб может “жить” только в своем поле, попытки выхода за границы поля означают ошибку исполнения программы. Поэтому Робу просто необходимо уметь проверять условия выхода на границу поля (правда, их всего два) и действовать по-разному в зависимости от того, выполняются эти условия или нет.

Условия, которые может проверять Роб — это:

"впереди край"

и

"впереди не край"

Оба этих условия используются в условных операторах. (Слово «оператор» означает нечто вроде «длинная инструкция (команда)». Действительно, как мы сейчас увидим, длина оператора, измеренная в числе слов, может быть любой.)

Условные операторы имеют следующий вид:

если впереди край, то...

иначе

...

конец ветвления

или

если впереди не край, то

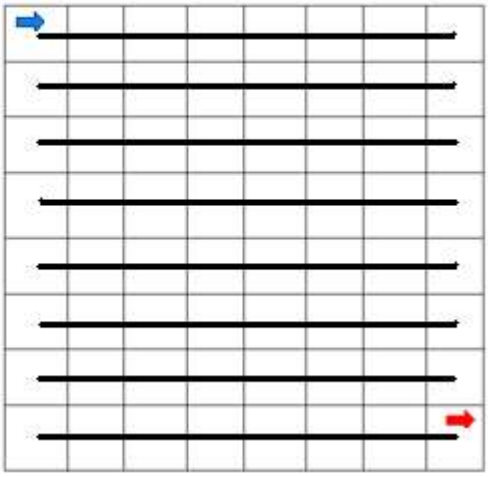
...

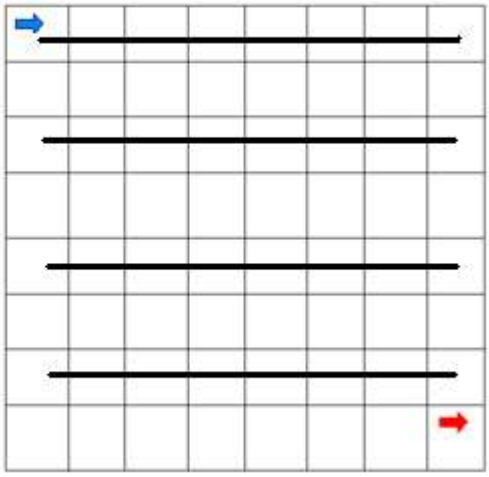
иначе

...

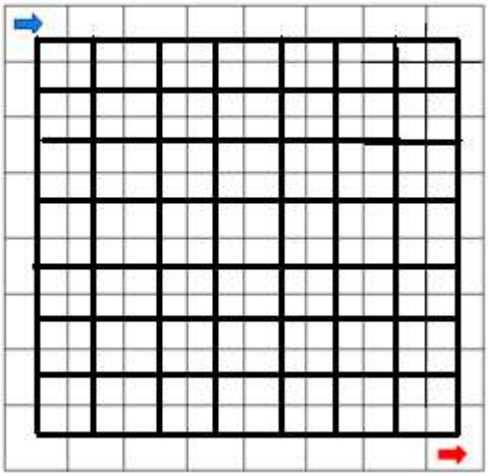
конец ветвления

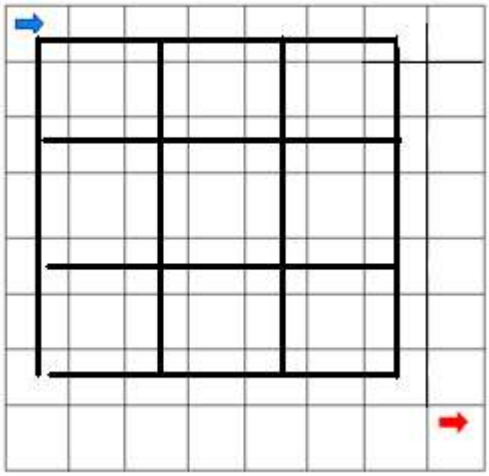
Если Роб был не перед краем:





Если Роб был перед краем:





Во многих случаях без проверки условий в программе не обойтись. Конечно, в том случае, когда Роб стартует в заранее известной точке поля известного заранее размера, можно также заранее все очень аккуратно подсчитать. А как быть
в том случае, если эта начальная точка неизвестна?

Пример RS3. Пусть программа состоит из 20 одинаковых, написанных подряд друг за другом условных операторов:

1. **если впереди не край, то**
2. **шаг**
3. **иначе**
4. **конец ветвления**

После выполнения этой программы, где бы Роб не находился в начальный момент, он прочертит прямую и окажется стоящим у края поля, но на край не вступит! Конечно, если величина поля не превышает 20x20.

Задачи к Теме 4

Задача Ro11b

Объясните, почему программа из примера RS3 работает именно так, как описано выше.

Задача Ro11c

Как, начиная в произвольной точке поля 10x10 клеток, прочертить прямую линию от одного края поля до другого (или вертикальную, или горизонтальную)? (Указание. Пройти до ближайшего края, развернуться на 180 градусов и прочертить.) Конечно, какая линия получится – горизонтальная или вертикальная, будет зависеть от начального положения Роба.

Задача Ro12

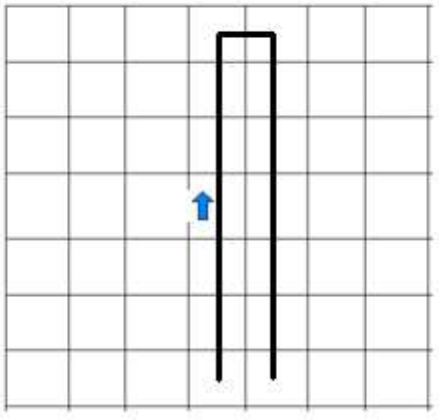
а) Поле для Роба имеет размер 8x8, начальное положение Роба неизвестно (то есть произвольное). Написать программу, по которой Роб начертит «рамку» по краям поля (то есть прочертит замкнутую линию, проходящую по крайним рядам поля).

б) Будет ли та же самая программа правильно работать на поле размером 6x6? 12x12?

в) Написать вариант той же программы для поля 12x12.

Задача Ro13

Поле для Роба имеет размер 2n+1 x 2n+1 (то есть число клеток по горизонтали и вертикали – нечетное. Начальное положение Роба – в центре поля, Роб смотрит вверх. Написать программу, по которой Роб прочертит вертикальную «скобу» в размер поля:

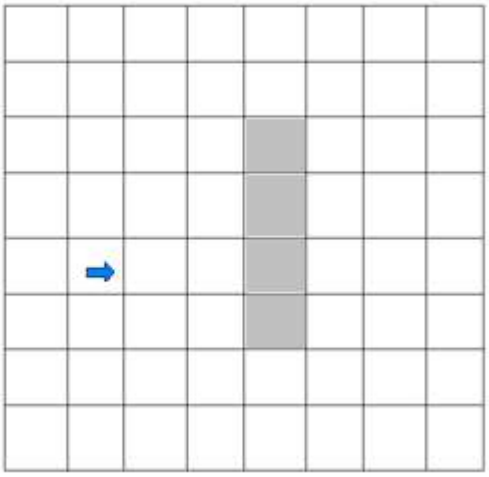


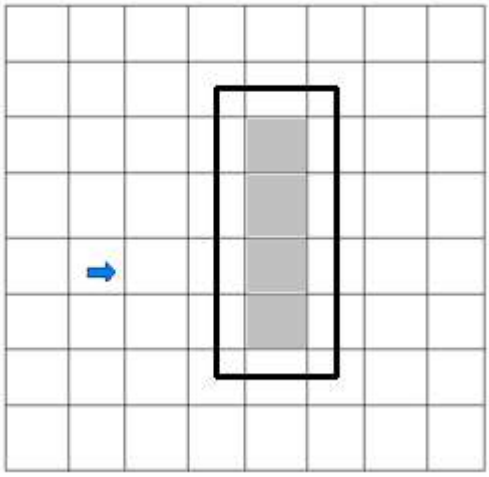
Задача Ro14

Как в задаче RO13, только «скоба» должна быть горизонтальна.

Задача Ro15

На поле некоторого (неизвестного) размера, не превышающего 12x12, имеется одна «стенка» лабиринта. В начальный момент Роб смотрит в направлении этой стенки. Робу требуется описать «рамку» вокруг этой стенки. Например, так:





Тема 5. Приемы написания алгоритмов

С помощью условных операторов уже возможно строить алгоритмы для решения весьма замысловатых задач. С составлением алгоритмов, однако, могут возникнуть проблемы. Возникает вопрос: имеются ли какие-либо специальные приемы для построения не совсем простых алгоритмов?

Сначала попробуем определить, что должны такие приемы давать разработчику программы?

Приемы и методы разработки программ должны, конечно, обеспечивать

- «направленность» («организованность») разработки программ, чтобы такая разработка не велась наугад;
- обеспечение надежности разрабатываемых программ (уменьшение количества ошибок при разработке);
- обеспечить «читаемость» программы (алгоритма), т.е. определенную естественность его записи (последовательности действий в алгоритме, разбиения алгоритма на отдельные части, обладающие отчетливыми целями и др.)

Основными же простыми приемами разработки являются:

- разработка (проектирование) программ «снизу вверх»;
- разработка (проектирование) программ «сверху вниз»;
- применение графических схем для представления программ (в т.ч. блок-схемы и расширенные блок-схемы);
- применение так называемых объективно-ориентированных методов.

Объектно-ориентированные методы мы отложим до следующей Темы (их область применения – разработка действительно сложных алгоритмов), а остальные приемы рассмотрим ниже. Кратко можно охарактеризовать эти методы так

Разработка программ. Метод «снизу вверх»

Метод «снизу вверх» - самый естественный, «простой» и потому распространенный у неопытных составителей алгоритмов. Состоит он в том, что мы постепенно накапливаем знания решений одних («более мелких») задач, а затем конструируем решения более сложных в виде несложных комбинаций этих более простых решений. Это и есть суть метода «снизу вверх»: от более простого к более сложному постепенно, с накоплением результатов.

Пример 1

Как составить букву Г из двух отрезков? Очевидно – нарисовать отрезок, сделать поворот – и прорисовать второй отрезок (могут быть варианты – смотря, откуда начинаем). Поэтому если мы знаем, как рисовать отрезок – можем нарисовать и букву Г.

Фильм "Проектирование написания буквы Г"

Пример 2

Нарисовать букву П. Букву Г и отрезки рисовать уже умеем.

Фильм 2.5.2. "Проектирование написания буквы П из буквы Г"

Заметьте: такой же метод проектирования решения всегда - почти без исключений - применяется при решении задач в школьной математике, физике и т.д. Узнаете? То есть на самом деле такой метод вам хорошо знаком!

Обычно метод «снизу вверх» применяется в форме последовательного построения решения, т.е. «начать с начала, потом решать проблемы построения решения (алгоритма) по мере их возникновения». Как это происходит – хорошо видно из приведенных примеров. Еще пример

Фильм 2.5.3. "Алгоритм слова ПОП"

Пока что мы проектировали методом «снизу вверх» только программы, у которых составляющие куски шли последовательно, без ветвлений. Как быть с ветвлениями? Да так же само, только из ветвления будет два выхода, а не один, как при строго последовательном ходе программы.

Пример 3

Если впереди край, отойти назад на три клетки и написать букву О. Если спереди не край, отойти на две клетки и начертить букву П. Роб смотрит вправо.

Фильм 2.5.4. "Разработка алгоритма написания"

Разработка программ методом «сверху вниз»

Метод проектирования «снизу вверх» хорошо работает на задачах, которые решаются более или менее последовательно. Другими словами, если задача состоит из «частей», то эти части не должны быть связаны сколь-нибудь сложным образом.

Для задач, решение которых имеет сложные связи между частями подходит метод проектирования алгоритмов «сверху вниз». Этот метод состоит из «последовательного дробления» исходной задачи на более мелкие (с указанием связей между ними). Разделение на более мелкие, простые задачи продолжается до тех пор, пока не будет достигнут уровень, на котором очевидным образом работает способ «снизу вверх». То есть основная задача разделяется до такого уровня, когда составляющие задачи уже достаточно простые, их можно уже решать последовательно.

В методе проектирования «сверху вниз» особое внимание должно уделяться тому, как осуществляется переход от одной задачи-составляющей к другой. Опыт показывает, что большинство ошибок возникает именно при написании таких переходов.

Пример 1

Задача: написать слово «порт».

Решение: сначала разделяем эту задачу на меньшие, более простые, задачи: написать каждую из букв (П, О, Р, Т).

Второе дробление на меньшие задачи: например:

- букву «П» мы уже знаем как записать;
- букву «О» можно составить из четырех отрезков (проектированием «снизу вверх»);
- букву «Р» можно составить из четырех отрезков (проектированием «снизу вверх»);
- букву «Т» можно составить из двух отрезков (опять-таки проектированием «снизу вверх»);

Как составлять отрезки, мы знаем. «Разбивку» основной задачи на меньшие закончено.

Теперь «сборка» алгоритма для основной задачи из меньших, «уже решенных»: нужно уметь переходить от одной (уже собранной) буквы к началу следующей. Эта задача, конечно, решается прыжками и поворотами.

Фильм 2.6.1. "Разработка алгоритма написания"

Пример 2

Роб смотрит вправо. Если слева от него край – написать «НЕТ», если нет края – написать «ЭТО».

Фильм 2.6.2. "Разработка алгоритма для двух вариантов"

Блок-схемы и расширенные блок-схемы

Для обеспечения наглядности проектирования программ удобно использовать графические методы. Иначе говоря, изображать алгоритмы или их схемы в виде картинок (схем). Это очень важно уже для предотвращения крупных ошибок: на картинках, схемах гораздо лучше, чем в тексте, можно изобразить схему связей между отдельными частями программы.

В процессе проектирования по способу «сверху вниз» как раз и получаются такого рода схемы связей отдельных частей алгоритма или программы.

Как видно, ближе всего такие схемы подходят для разработок «сверху вниз»: изображаются некоторые части решения, соответствующие меньшим задачам, на которые разбита основная задача и связи между этими более мелкими задачами. Затем, при необходимости, можно каждую такую «подзадачу» раздробить снова на меньшие и т.д. Где прекращается разбиение задачи – определяется разработчиком.

Пример 1

Если надо написать слово ПОСТ и разработчик программы знает, как вырисовывать отдельные буквы, то на схеме незачем расписывать этот процесс вырисовывания букв.

Фильм 2.7.1. "Расширенная блок-схема 1"

Пример 2

Если слева край – написать слово ПОСТ, если нет – слово ГОСТ.

Фильм 2.7.2. "Разработка алгоритма с ветвлением"

А что получится, если подобную графическую схему алгоритма довести до самых мелких частей? Самые маленькие «задачи», которые нельзя уже расчленить дальше, будут соответствовать отдельным командам языка, на котором записывается алгоритм. То есть каждая из «самых маленьких» задач, на которую оказалась разбита исходная задача, будет решается одной командой языка.

Пример 3

Рассмотрим построение блок-схемы для задачи RO-11a: необходимо прочертить прямую линию от одного края поля до другого в том направлении, в котором смотрит Роб.

Фильм 2.7.3. "Алгоритм к задаче 11"

Такая доведенная «до предела элементарности» схема называется блок-схемой алгоритма. Блок-схемы могут пригодиться для начинающих изучать языки программирования: по готовой блок-схеме легко записывать программы (то есть при уже готовом алгоритме в форме блок-схемы надо просто переписать ее командами из языка).

Пример 3

Рассмотрим построение блок-схемы для задачи RO-11a: необходимо прочертить прямую линию от одного края поля до другого в том направлении, в котором смотрит Роб.

Фильм 2.7.3. "Алгоритм к задаче 11"

Такая доведенная «до предела элементарности» схема называется блок-схемой алгоритма. Блок-схемы могут пригодиться для начинающих изучать языки программирования: по готовой блок-схеме легко записывать программы (то есть при уже готовом алгоритме в форме блок-схемы надо просто переписать ее командами из языка).

Для более сложных программ блок схемы получаются очень громоздкими и вовсе не наглядными. Поэтому после освоения языка их уже практически не используют.

Напротив, схемы программ, составленные из достаточно крупных блоков для задач-составляющих, всегда оказываются весьма полезными. Помимо удобства проектирования, они создают и удобство при выявлении ошибок. А именно, поиск и устранение ошибок (или, как говорят, отладка программы) превращается в

- отладку отдельных блоков программы – прямо по блокам со схемы,
- отладку связей между схемами.

А каждый блок представляет собой, по сути, меньшую и более простую программу, чем исходная. То есть для ошибочно работающей программы можно сначала выделить блок, который работает неверно, а затем уже искать ошибку в его реализации в виде программы.

Эта ценная черта расширенных блок-схем в настоящее время широко используется. В настоящий момент имеются даже графические языки, специально предназначенные для построения таких укрупненных схем программ, обеспечивающие все необходимые действия с ними (уточнения и дальнейшие разбиения, перевод в «обычные» языки программирования).

Тема 6. Процедуры в среде Роб

Вы уже знаете, что в математике широко применяются сокращения: например, "a умножить на b". Для натуральных чисел a и b — это сокращение выражения "сложить a + a + ... + a b раз".
Сокращения позволяют избежать ненужных повторений и описаний.

То же самое можно делать и в языке среды Роб, вводя (описывая) сокращения в программе. Такие сокращения называются процедурами. Сама процедура при этом - маленькая программа .

Описания всех процедур (в программе их может быть несколько) договоримся помещать обязательно
в конец программы. Если процедур несколько, то описания могут идти в произвольном порядке.

Само описание процедуры будет иметь вид:

процедура A

....

конец процедуры

Здесь точками обозначена та программа, сокращением которой является процедура (ее назовем телом процедуры). Буква A — это имя процедуры. Оно необходимо потому, что в программе может быть несколько процедур и разные процедуры надо отличать друг от друга Именем процедуры в среде Роб может быть любая латинская буква.

Используется процедура в любом месте программы как "обычная" команда. Для этого в программе
в нужном месте ставится вызов этой процедуры,
например:

сделай A

Выполняется при этом тело процедуры A. Приведенная выше команда “**Сделай A**” называется обращением к процедуре A.

сделай b

сделай b

процедура b

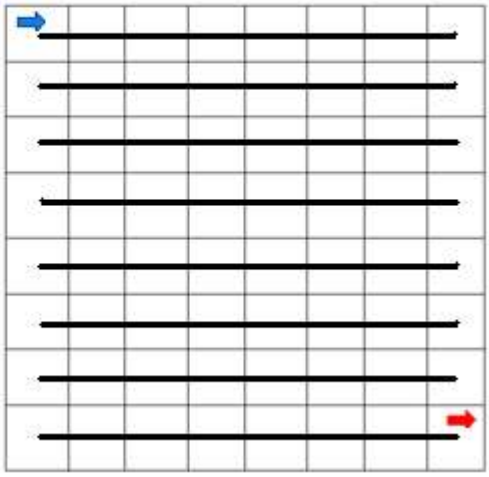
шаг

поворот

шаг

поворот

(повернуться трижды придется из-за того, что поворачиваем всегда в одну сторону).



Процедура будет выглядеть следующим образом:

процедура b

шаг

поворот

шаг

поворот

поворот

поворот

конец процедуры

Теперь наша программа для рисования четырех ступенек будет выглядеть так:

сделай b

сделай b

сделай b

сделай b

процедура b

шаг

поворот

шаг

поворот

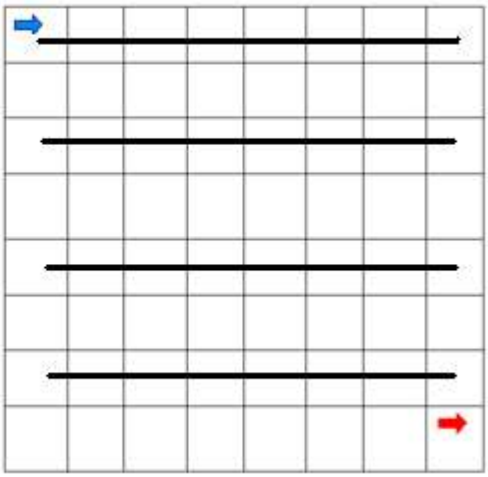
поворот

поворот

конец процедуры

Результат выполнения этой программы на рисунке

перед вами:



Эту же программу можно записать и без применения процедур, но тогда она станет почти в четыре раза длиннее. Кроме того, если придется разбираться в кем-то написанной программе, то это будет проще сделать именно в виде с процедурами.

Задачи к Теме 6

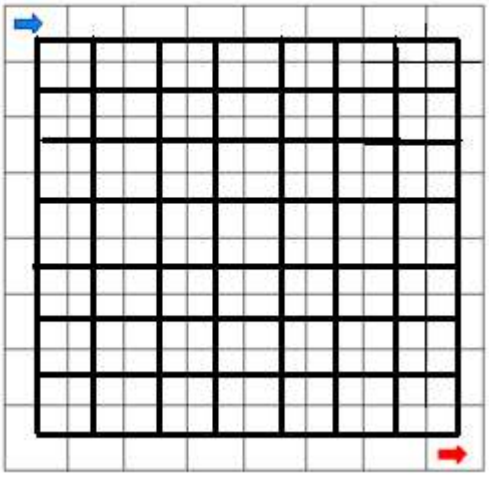
Задача RO24

а) Наберите эту программу и посмотрите ее выполнение в отладочном режиме.

б) Напишите эту же программу без применения процедур. Сравните получившуюся длину.

Задача RO25

Напишите программу, рисующую "лесенку вниз".



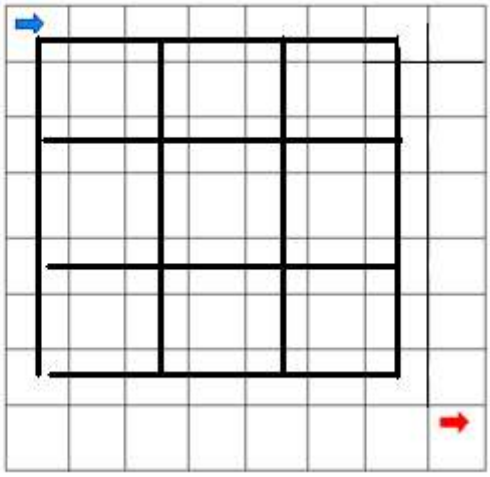
Задача RO26

а) Напишите программу, рисующую "лесенку вверх" до правого края поля. Ширина поля неизвестна, но не более 8. Роб в начале смотрит вправо (клетка с Робом также неизвестна).

б) Напишите программу, рисующую "лесенку вниз" до правого края поля. Ширина поля неизвестна, но не более 10. Роб в начале смотрит вправо (клетка с Робом также неизвестна).

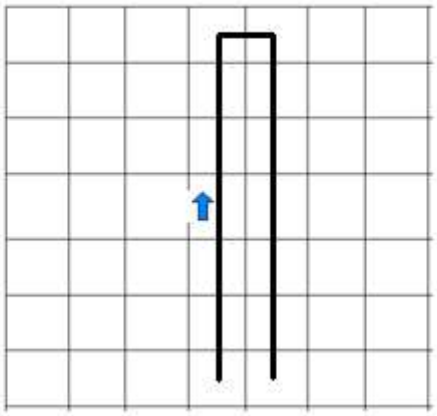
Задача RO27

Напишите программу, рисующую "ступеньки" лесенки вверх.



Задача RO28

Напишите программу, рисующую "ступеньки" лесенки вниз.



Задача RO29

а) Напишите программу, рисующую "ступеньки лесенки вверх" до правого края поля. Ширина поля неизвестна, но не более 8. Роб в начале смотрит вправо (клетка с Робом также неизвестна).

б) Напишите программу, рисующую "ступеньки лесенки вниз" до правого края поля. Ширина поля неизвестна, но не более 10. Роб в начале смотрит вправо (клетка с Робом также неизвестна).

Задача RO30

Напишите программу, рисующую "пьедестал" и использующую процедуры из предыдущих задач.

Тема 7. Оператор цикла в среде Роб

Еще одно очень важное сокращение в языке среды Роб — оператор цикла. (Напомним, оператор – это инструкция переменной длины.) Оператор цикла так же, как и процедура, позволяет избежать повторов в тексте программы и обычно существенно сокращает объем программы. Более того, многие программы без оператора цикла вообще написать нельзя!

Оператор цикла имеет вид:

пока (условие), повторять

...

конец цикла

Оператор цикла всегда начинается со слова “пока” и заканчивается словами “конец цикла”. Здесь условие точно такое же, как и в условном операторе (напомним, это "впереди край" или "впереди не край"), а точками снова обозначен некоторый набор команд, операторов и процедур, то есть какая-то программа (ее называют телом цикла).

Оператор цикла выполняется следующим образом:

1) сначала проверяется условие;

2) если оно выполнено — исполняется тело цикла, а затем Роб возвращается в начало этого же цикла (т.е. снова будет проверять выполнение условия);

3) если это условие оказалось невыполненным, то тело цикла уже не исполняется, и Роб переходит к выполнению следующего оператора.

Пример RS5

При выполнении следующей программы Роб нарисует прямую линию до того края, в направлении которого он смотрел в начальный момент выполнения программы.

пока впереди не край, повторять

шаг

конец цикла

Пример RS6

По программе

пока впереди не край, повторять

сделай b

конец цикла

процедура b

шаг

поворот

если впереди край, то

иначе

шаг

поворот

поворот

поворот

конец ветвления

конец процедуры

Роб нарисует "лесенку" от любой точки, где он находился в начальный момент выполнения программы, до края поля.

В примере RS6 использовалась процедура b, рисующая одну "ступеньку". Эта процедура многократно повторяется в цикле, пока Роб не дойдет до края; после чего выполнение программы закончится.

Посмотрите работу программ из примеров RS5 и RS6 в режиме отладки.

Задачи к Теме 7

В задачах RO31 — RO33 используйте алгоритмы, написанные для задач RO25 — RO30, но с использованием операторов цикла.

Задача RO31

Напишите программу, рисующую "ступеньки" лесенки вниз от точки, в которой находится Роб (смотрит вправо), до границы поля.

Задача RO32

Напишите программу, рисующую "ступеньки" лесенки вверх от точки, в которой находится Роб (смотрит вправо), до границы поля.

Задача RO33

Напишите программу, рисующую лесенку вверх от точки, в которой находится Роб (смотрит вправо), до границы поля.

Задача RO34

Обратите внимание, что в задачах RO31-RO33 нет ограничений на ширину поля, а в задачах RO26 и RO29 – есть. Объясните, почему без оператора цикла решить задачи RO31-RO33 не представляется возможным.

Задача RO35

Напишите программу, рисующую лесенку вверх от точки, в которой находится Роб (смотрит влево), до границы поля.

Задача RO36

Напишите программу, рисующую лесенку сначала вправо вверх от точки, в которой находится Роб (смотрит вправо), до границы поля, а от этой границы – лесенку влево вверх до границы поля. Для этого используйте программы из предыдущих задач.

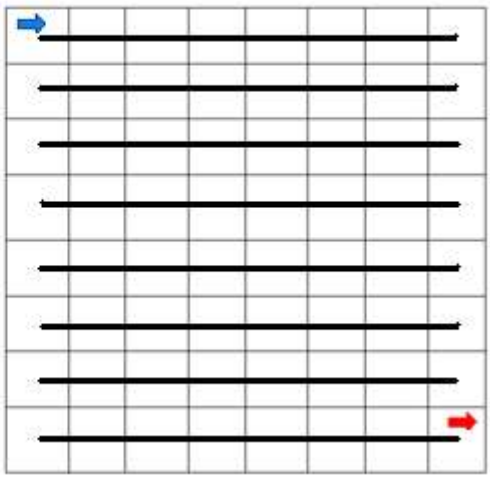
Тема 8. Вложенные циклы в среде Роб

В теле одного цикла могут находиться операторы других циклов — они называются вложенными. Как определить, какому началу цикла ("пока ...") соответствует какой конец ("конец цикла")?

Делается это так: находим такое начало цикла, за которым есть конец цикла, но между ними нет других начал цикла. Тем самым мы нашли начало и конец “самого внутреннего” цикла. Отметим его. Теперь будем повторять такое же действие, игнорируя отмеченный цикл. Далее найдем пару "начало цикла” — “конец цикла" от следующего внутреннего цикла и т.д.

Заметим теперь, что в точности так же само мы разбираемся со скобками в обычных выражениях в арифметике. Сначала ищем самую внутреннюю пару скобок (между которыми уже скобок нет), выполняем там действия и т.д. Т.е. здесь можно считать «пока» открывающей скобкой оператора цикла, а «конец цикла» - закрывающей скобкой.

Пример RS7



Напишем программу (алгоритм), по которой Роб нарисует горизонтальные прямые от края до края поля по каждому ряду точек поля как показано на рисунке.

Предложим следующий алгоритм решения:

Роб чертит прямую линию от левого края поля до правого, затем возвращается прыжками по той же линии, спускается на одну строку, снова чертит прямую линию и т.д. Конечно, при возврате и «спуске» нужна проверка, есть ли еще внизу пустые строки.

Этот алгоритм использует три цикла, два из которых вложенные

```
Цикл 3  пока впереди не край, повторять
          пока впереди не край, повторять
Цикл 1      шаг
          конец цикла
            поворот
            поворот
            поворот
          если впереди край, то
            поворот
          иначе
            поворот
            поворот
            поворот
          пока впереди не край, повторять
Цикл 2      прыжок
          конец цикла
            поворот
            прыжок
            поворот
          конец ветвления
        конец цикла
```

Вложенный цикл 1 обеспечивает рисование прямой линии, вложенный цикл 2 — возврат Роба от правого края к левому, а цикл 3 обеспечивает многократное повторение циклов 1,2 и смещение Роба вниз, пока он не дойдет до нижнего края. Циклы в этой программе размечены скобочками только для наглядности. Обратите внимание, что один из циклов находится внутри условного оператора, проверяющего, есть ли еще под Робом пустая строка.

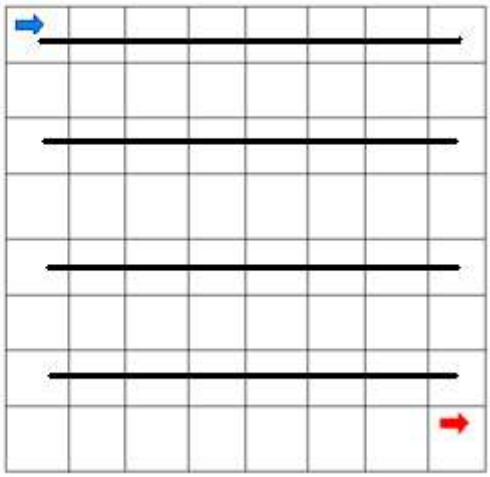
Задачи к Теме 8

Задача RO37

Эта программа будет работать правильно на поле произвольного размера. Объясните, почему.

Задача RO38

а) Разрисуйте поле горизонтальными прямыми, идущими через один ряд. Роб начинает в левом верхнем углу (смотрит вправо).



б). Эта программа будет работать правильно на поле произвольного размера. Объясните, почему.

Задача RO39

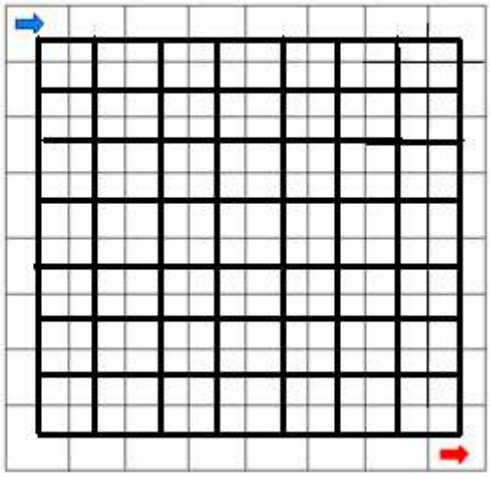
Разрисуйте поле вертикальными прямыми. Роб начинает в левом верхнем углу (смотрит вправо).

Задача RO40

Разрисуйте поле вертикальными прямыми, идущими через один ряд. Роб начинает в левом верхнем углу (смотрит вправо).

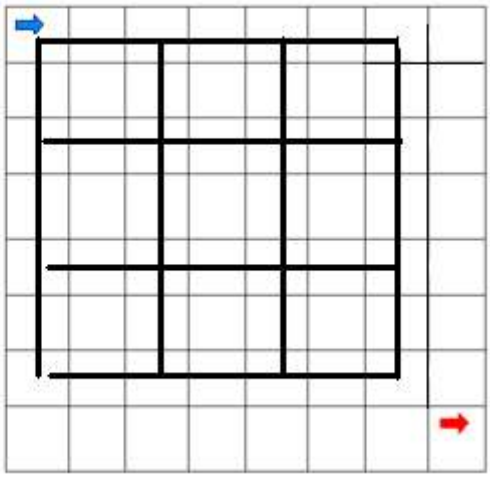
Задача RO41

Расчертите поле в клеточку со стороной 1. Роб начинает в левом верхнем углу (смотрит вправо).



Задача RO42

Расчертите поле в клеточку со стороной 2. Роб начинает в левом верхнем углу (смотрит вправо).

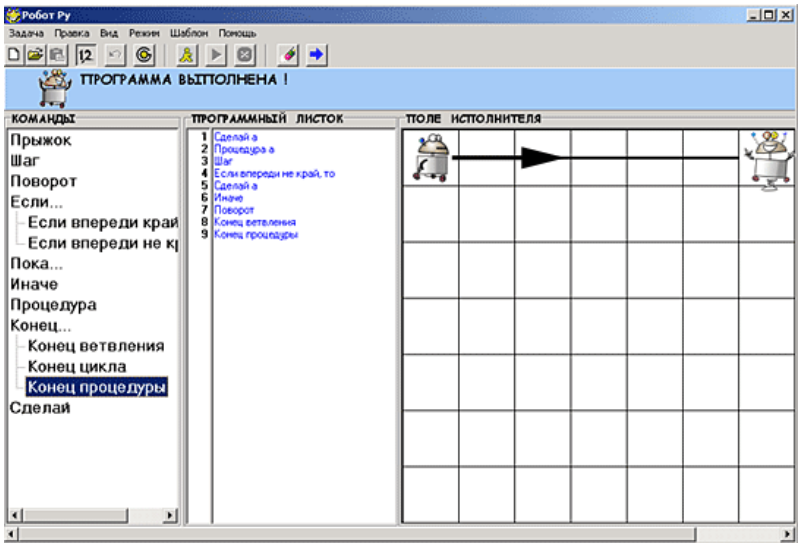


Тема 9. Рекурсия в среде Роб

Внутри тела процедуры разрешено употреблять
ее заголовок. Как в этом случае Роб будет выполнять процедуру? Конечно, как положено по правилам для процедур: дойдя до этого «внутреннего» заголовка, надо снова начинать выполнять тело этой процедуры с самого начала! Такое «обращение процедуры самой к себе» иногда (но не так часто для Роба) позволяет существенно сокращать запись программы и называется рекурсией. Применять рекурсию следует весьма аккуратно: здесь очень легко допустить ошибку!

Пример 1

Рекурсией можно заменять применение оператора цикла. По следующей программе Роб прочертит прямую линию от той точки, где он стоял, в том направлении, куда он смотрел в начальный
момент — до края поля:

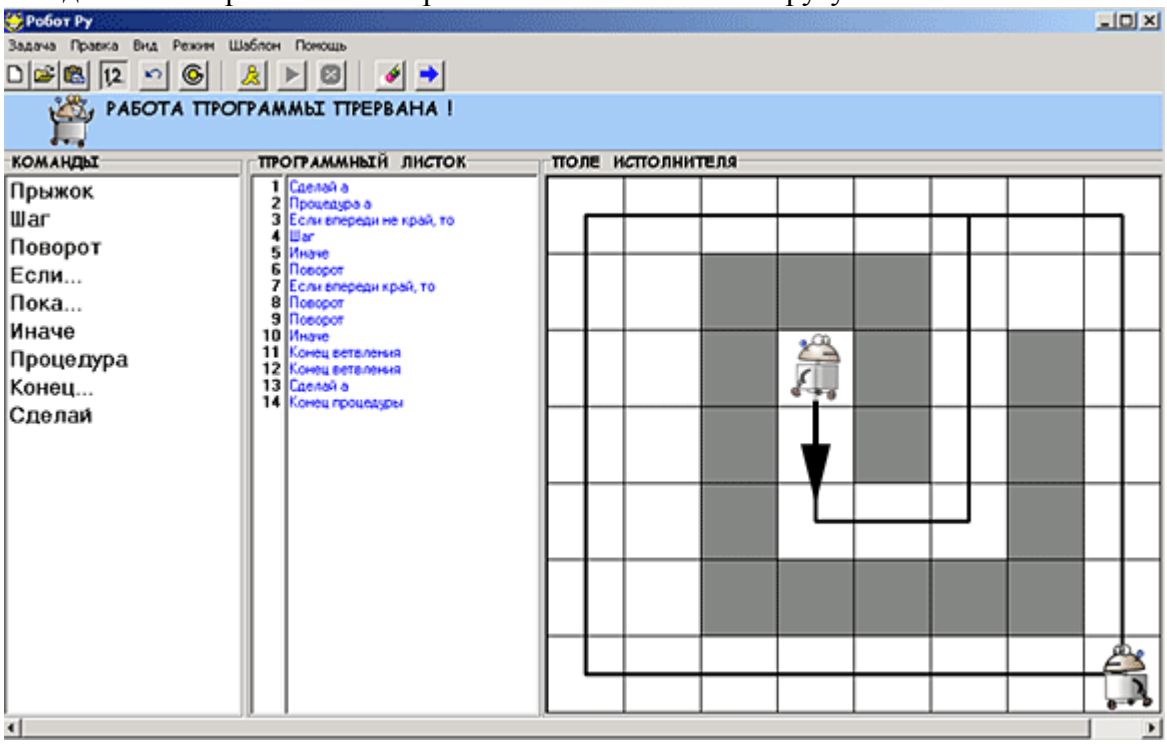


Текст программы:

1. **сделай а**
2. **процедура а**
3. **шаг**
4. **если впереди не край, то**
5. **сделай а**
6. **иначе**
7. **поворот**
8. **конец ветвления**
9. **конец процедуры**

Пример 2

Роб должен выбраться из лабиринта и начать бегать по кругу:



Текст программы:

1. Сделай а
2. Процедура а
3. Если впереди не край, то
4. Шаг
5. Иначе
6. Поворот
7. Если впереди край, то
8. Поворот
9. Поворот
10. Иначе
11. Конец ветвления
12. Конец ветвления
13. Сделай а
14. Конец процедуры