

АКАДЕМИЯ НАУК СССР
НАУЧНЫЙ ЦЕНТР БИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

Методические рекомендации

Н.Л.ЛУНИНА

**ОБУЧЕНИЕ ШКОЛЬНИКОВ
ПРОГРАММИРОВАНИЮ**

ПУЩИНО • 1986

АКАДЕМИЯ НАУК СССР
НАУЧНЫЙ ЦЕНТР БИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

Методические рекомендации

Н.Л.ЛУНИНА

**ОБУЧЕНИЕ ШКОЛЬНИКОВ
ПРОГРАММИРОВАНИЮ**

ПУЩИНО • 1986

В методических рекомендациях излагается курс программирования, который в течение нескольких лет преподается учащимся Пущинской экспериментальной средней школы в рамках производственного обучения. В основу его положен начальный курс программирования, преподаваемый на механико-математическом факультете МГУ. В первой части курса изучаются основные понятия программирования — исполнители и объекты, технология программирования сверху—вниз, управляющие конструкции — на примере учебного языка с использованием русской нотации. Издание содержит теоретический материал и задания, предназначенные для выполнения на ЭВМ; приведены образцы выполнения заданий. В приложении описано разработанное на мехмате МГУ программное обеспечение, которое может быть использовано при проведении занятий.

Методические рекомендации могут быть полезны и при обучении без ЭВМ, а также для самостоятельного обучения заинтересованных учащихся.

Наталья Леонидовна Лунина

ОБУЧЕНИЕ ШКОЛЬНИКОВ ПРОГРАММИРОВАНИЮ
Методические рекомендации

Отредактировано и подготовлено к печати в ОНТИ НЦБИ АН СССР

Редактор Л.И.Захватова

Технический редактор С.Г.Бабакулиева

Корректоры Т.К.Крейцер, Л.М.Орлова

Подписано в печать 4.09.86 г. Т17050. Уч.-изд.л. 3,5.

Усл. печ. л. 5,5. Формат 60x90/16. Тираж 500 экз.

Заказ 7438Р. Цена 40 к. Изд. № 444.

Набрано на фотонаборном автомате ФА-1000.

Отпечатано на ротапринте в Отделе научно-технической информации

Научного центра биологических исследований АН СССР в Пущине.

ПРЕДИСЛОВИЕ

В данной работе излагается опыт обучения программированию учащихся Пушкинской экспериментальной средней школы в рамках производственного обучения по специальности «Программирование», которое проводится в НИВЦ АН СССР с 1981/82 учебного года.

В основу излагаемых рекомендаций школьного курса положен начальный курс программирования, который читается студентам второго курса механико-математического факультета МГУ с 1980/81 уч. года доцентом А.Г.Кушниренко. Под его руководством коллективом сотрудников, аспирантов и студентов механико-математического факультета было разработано поддерживающее этот курс программное обеспечение.

Текст приложения к методическим рекомендациям написан аспирантом механико-математического факультета МГУ Д.В.Варсанофеевым.

В отличие от многих других курсов программирования, основное внимание в нашем курсе уделено не конкретной вычислительной машине или операционной системе и не конкретному языку программирования, а общим теоретическим понятиям современного программирования. Учащиеся вначале знакомятся с ними на примере учебного языка МИНИ. Запись алгоритма на этом языке, формальном по существу и близком к русскому языку по форме, является программой, выполнимой на ЭВМ. Язык МИНИ весьма прост, но содержит набор средств, позволяющих решать достаточно сложные учебные задачи.

Теоретические понятия, усвоенные при изучении учебного языка, закрепляются далее при изучении производственных языков программирования (Фортран, Паскаль).

Материал, содержащийся в настоящих рекомендациях (ч. I), рассчитан на первое полугодие. Занятия проводятся 1 день (5 уроков) в неделю. Во время обучения в основном используется ЭВМ класса СМ-4: все приведенные здесь задания ученики выполняют на машине. Программное обеспечение занятий, частично разработанное на механико-математическом фа-

культете МГУ, а частично созданное специально для школьников в НИВЦ АН СССР, работает под управлением операционной системы RSX-11M.

Материал рекомендаций может быть использован и при обучении без ЭВМ. Однако эффективность обучения при этом существенно снизится.

У учащихся, занимающихся по этой методике, не предлагаются никаких предварительных знаний о программировании. Как показал опыт, любознательность и желание заниматься позволяют новичку успешно преодолеть все трудности. В конце первого года обучения учащиеся оказываются способны разрабатывать программное обеспечение для занятий следующих групп, создавая программы на Фортране объемом до 500 операторов, содержащие до двух десятков подпрограмм.

Большую помощь в работе над текстом данных рекомендаций оказал В.В.Левитин. Э.Э.Шноль провел тщательное редактирование рукописи. Постоянное внимание к делу обучения школьников и к работе над разработками рекомендаций оказывал А.Г.Кушниренко. Всем названным товарищам автор выражает глубокую благодарность.

ТЕМА 1. ОСНОВНЫЕ ПОНЯТИЯ ПРОГРАММИРОВАНИЯ

Алгоритмы, исполнители, предписания

На наших занятиях мы будем писать программы и выполнять их на ЭВМ. Необходимость использования ЭВМ возникает тогда, когда у человека есть *задача*.

Рассмотрим следующую задачу: в лабиринт, план которого нам известен, засыпается человек или автоматический прибор (назовем его для определенности ПУТНИК). Нам нужно пройти его по лабиринту от входа до выхода (план лабиринта см. на с. 6).

На плане приведен лабиринт размером 51×28 .

Символами "==" отмечена стена,
"X" — препятствия;
остальные клетки свободны.

Мы принимаем решение подвести ПУТНИКА к восточной стене и спуститься вдоль нее. Если у этой стены нам встречаются препятствия, будем их обходить самым экономным образом (идти рядом с ними).

Нам понадобятся средства управления ПУТНИКОМ: придется его двигать и поворачивать. Будем считать, что ПУТНИК понимает следующие приказы:

сделай шаг
сделай 2 шага
сделай 3 шага
сделай 4 шага
сделай 8 шагов
сделай 16 шагов

повернись налево
повернись направо
повернись кругом

повернись на север
повернись на юг
повернись на запад
повернись на восток

вход

```
= ======  
= . =  
= =  
= = XXXXXXXXXXXXXXXXXXXXXXXX=  
= = XXXXXXXXXXXXXXXXXXXXXXXX=  
= =  
= =  
= = XXXXXXXXXXXXXXXXXXXXXXXX =  
= = XXXXX / XXXXXXX XXXXXXX XXXXX / XXXXXXX =  
= =  
= =  
= =  
= = XXXXXXXXXXXXXXXXXX XXXXXXX XXXXXXX XXXXXXXX =  
= = XXXXXXX XXXXXXX XXXXXXX XXXXXXXX =  
= =  
= =  
запад = = восток  
=XX XXXXXXXXXXXXXXXXXX XXXXXXXX =  
=XX XXXXXXXXXXXXXXXXXX XXXXXXXX =  
= =  
= =  
=XX XXXXXXXXXXXXXXXXXX XXXXXXXX =  
=XX XXXXXXXXXXXXXXXXXX XXXXXXXX =  
= =  
= =  
= = XXXXXXXXXXXXXXXXXX XXXXXXXX =  
= = XXXXXXXXXXXXXXXXXX XXXXXXXX =  
= =  
= =  
= =  
===== =  
выход
```

Действия, которые он производит по этим приказам, следующие: за 1 шаг он перемещается на одну клетку в том направлении, куда стоит «лицом»; налево и направо поворачивается относительно своего положения, а на север и т.д. — относительно плана лабиринта (север вверху).

Если мы выпишем друг за другом необходимые приказы

для передвижения ПУТНИКА в лабиринте, мы получим программу действий ПУТНИКА для решения нашей задачи.

Подведем некоторые итоги. Перед нами стояла задача — пройти по лабиринту. Мы предложили следующую схему действий, или, другими словами, *алгоритм* — дойти до восточной стены и спускаться вдоль нее, обходя препятствия. Выполнять эти действия будет *исполнитель ПУТНИК*. Он обладает следующей *системой предписаний*:

1. НАЧНИ РАБОТУ В КВАДРАНТЕ С КОДОМ XXXXXX
(где XXXXXX — 6-значное число из цифр от 0 до 9)
2. СДЕЛАЙ ШАГ
3. СДЕЛАЙ 2 ШАГА
4. СДЕЛАЙ 3 ШАГА
5. СДЕЛАЙ 4 ШАГА
6. СДЕЛАЙ 8 ШАГОВ
7. СДЕЛАЙ 16 ШАГОВ
8. ПОВЕРНИСЬ НАЛЕВО
9. ПОВЕРНИСЬ НАПРАВО
10. ПОВЕРНИСЬ КРУГОМ
11. ПОВЕРНИСЬ НА СЕВЕР
12. ПОВЕРНИСЬ НА ЮГ
13. ПОВЕРНИСЬ НА ЗАПАД
14. ПОВЕРНИСЬ НА ВОСТОК
15. КОНЧИ РАБОТУ

В этом списке появились предписания, о которых мы еще не говорили — НАЧНИ РАБОТУ... и КОНЧИ РАБОТУ. Никакое другое предписание не может быть выполнено до первого из них или после второго. Предписание НАЧНИ РАБОТУ... активизирует исполнитель и устанавливает его в некоторое исходное для задачи состояние (в нашем случае под клеткой «вход» лицом на юг).

Иногда в предписании НАЧНИ РАБОТУ... задается некоторая дополнительная информация, уточняющая задачу. Так, мы будем рассматривать разные лабиринты и для определенности указывать, в каком именно лабиринте работает наш ПУТНИК. Правило для построения лабиринта по его коду будет указано ниже, в задании для исполнителя ПУТНИК.

Теперь мы в состоянии написать *программу* для решения задачи. Она выглядит следующим образом:

НАЧНИ РАБОТУ В КВАДРАНТЕ С КОДОМ 234561
ПОВЕРНИСЬ НА ВОСТОК
СДЕЛАЙ 16 ШАГОВ
СДЕЛАЙ 16 ШАГОВ
СДЕЛАЙ 16 ШАГОВ

„ „ дошли до восточной стены
„ ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ ШАГ

„ „ дошли до первого препятствия у восточной стены
„ ПОВЕРНИСЬ НА ЗАПАД
СДЕЛАЙ 16 ШАГОВ
СДЕЛАЙ 8 ШАГОВ
СДЕЛАЙ 4 ШАГА
СДЕЛАЙ 2 ШАГА
ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ 3 ШАГА
ПОВЕРНИСЬ НА ВОСТОК
СДЕЛАЙ 16 ШАГОВ
СДЕЛАЙ 8 ШАГОВ
СДЕЛАЙ 4 ШАГА
СДЕЛАЙ 2 ШАГА

„ „ обошли первое препятствие у восточной стены
„ ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ 4 ШАГА
СДЕЛАЙ ШАГ

„ „ дошли до второго препятствия у восточной стены
„ ПОВЕРНИСЬ НА ЗАПАД
СДЕЛАЙ 16 ШАГОВ
СДЕЛАЙ 8 ШАГОВ
СДЕЛАЙ 3 ШАГА
ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ 3 ШАГА
ПОВЕРНИСЬ НА ВОСТОК
СДЕЛАЙ 16 ШАГОВ
СДЕЛАЙ 8 ШАГОВ
СДЕЛАЙ 3 ШАГА

„ „ обошли второе препятствие у восточной стены
„ ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ ШАГ

„ „ дошли до третьего препятствия у восточной стены
„ ПОВЕРНИСЬ НА ЗАПАД

СДЕЛАЙ 8 ШАГОВ
ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ 3 ШАГА
ПОВЕРНИСЬ НА ВОСТОК
СДЕЛАЙ 8 ШАГОВ

”
” обошли третье препятствие у восточной стены

”
ПОВЕРНИСЬ НА ЮГ
СДЕЛАЙ 8 ШАГОВ
СДЕЛАЙ 2 ШАГА
КОНЧИ РАБОТУ

Мы видим, что программа содержит не только предписания, но и некоторые строки, начинающиеся знаком ” — двойная кавычка. Эти строки не вызывают никаких действий ПУТНИКА, а служат только для повышения наглядности программы при чтении человеком. Такие строки называются комментариями.

Итак, программа — это последовательность предписаний или некоторая формальная запись такой последовательности (например, программа может содержать комментарии, не являющиеся предписаниями).

Возможность употреблять в программе нечто, отличное от предписаний, обеспечивается тем, что в момент выполнения программы между ней и исполнителем находится некоторый посредник, читающий программу и дающий на выполнение исполнителю его предписания. Посредник умеет распознавать комментарии. Понятие комментария является общим; форма его записи в разных случаях различна. Мы будем записывать комментарий по правилам учебного языка МИНИ. В этом языке комментарием является все, что располагается от знака ” — двойная кавычка — до конца строки.

При составлении программы мы можем ошибиться. Например, если мы напишем ”СДЕЛАЙ 5 ШАГОВ, ПУТНИК такую программу выполнить не сможет — он умеет понимать только свои собственные предписания (из приведенного списка). Если же мы допустим другую ошибку и заставим ПУТНИКА сделать лишний шаг, стоя у стены лицом к ней, произойдет ситуация отказ — исполнитель не сможет выполнить свое предписание и сообщит нам об этом.

Если мы не сделаем никаких ошибок, ПУТНИК будет проведен по лабиринту, и по предписанию КОНЧИ РАБОТУ распечатается план лабиринта и схема движения ПУТНИКА: (см. с. 10).

Сделаем несколько замечаний, показывающих, что алгоритм, исполнитель и программа тесно связаны друг с другом.

1. Мы могли бы предложить другой алгоритм решения

(спускаться вдоль западной стены; двигаться по ближайшему проходу и т.п.), тогда программа была бы, конечно, другой.

2. Мы могли бы придумать такой алгоритм, что возможностей нашего исполнителя не хватило бы, пришлось бы подыскивать другой исполнитель (например, умеющий находить ближайший проход).

3. Наш исполнитель мог бы обладать другими возможностями, например, уметь делать 10 шагов или поворачиваться только налево, тогда программа (даже при использовании одного и того же алгоритма) выглядела бы иначе.

Объекты

При выполнении программы для правильного выполнения очередного предписания исполнитель ПУТНИК обязан помнить историю своего движения по лабиринту.

Прежде всего он должен помнить, по какому лабиринту идет. Описать его мы можем табличкой размером 51×28 (по числу клеток), причем в каждой ячейке этой таблички должно быть указано, является ли клетка частью стены, частью препятствия или свободной.

ПУТНИК должен также помнить, где он находится в данный момент. Текущее положение ПУТНИКА можно описать двумя числами — номером ряда и номером клетки в ряду.

Наконец, ПУТНИК должен помнить, куда он повернут «лицом» в данный момент. Это направление можно описать одним из четырех слов — север, запад, юг, восток.

Теперь скажем, как называются наши новые понятия.

Возможность помнить свое состояние обеспечивают исполнителю его *объекты*. Одни объекты не меняются во время выполнения программы и называются *постоянными*. В нашем примере таким неизменным объектом является сам лабиринт (его схема). Другие объекты меняются и называются *переменными*.

Каждый объект имеет *имя, тип, состояние* (часто его называют также *значением*).

Именем объекта может быть любое слово (иногда даже несколько слов). Наряду с буквами в имя могут включаться цифры. Мы будем стараться выбирать имена объектов так, чтобы они соответствовали их содержанию.

Тип объекта определяется значениями, которые может объект принимать, и операциями, которые можно над ними выполнять. Наиболее простыми и распространенными являются объекты числовых типов, принимающие числовые значения, над которыми можно выполнять арифметические действия и операции сравнения. При этом в одном случае могут быть разрешены любые вещественные числа, а в другом — любые целые числа. Указание этого и задает тип такого числового объекта — *вещественный* или *целый*. В нашей задаче объекты НОМЕР РЯДА и НОМЕР КЛЕТКИ В РЯДУ являются объектами целого типа.

Но в нашей задаче мы столкнулись и с объектами более сложных типов. Приведем их сейчас не для запоминания, а для знакомства.

Объект НАПРАВЛЕНИЕ имеет тип *перечисление*. Это значит, что он может принимать одно из заранее перечисленных значений. Значениями могут служить слова, слова и т.д. В нашем случае разрешены 4 значения (4 слова)

НАПРАВЛЕНИЕ = (СЕВЕР, ЮГ, ЗАПАД, ВОСТОК)

Объект ТЕКУЩЕЕ ПОЛОЖЕНИЕ имеет тип *вектор* размерности 2:

ТЕКУЩЕЕ ПОЛОЖЕНИЕ = (НОМЕР РЯДА, НОМЕР КЛЕТКИ В РЯДУ)

Постоянный объект **ЛАБИРИНТ** имеет тип **матрица** (размером 51×28). В каждой ячейке этой матрицы (таблички) стоят символы: "—" , "X" или не стоит ничего (удобно говорить, что стоит особый символ пробел: " ").

Более общая точка зрения: **ЛАБИРИНТ** — это объект типа **матрица**, каждый элемент которой есть объект **КЛЕТКА** типа **перечисление** (в данном случае из трех значений)

КЛЕТКА = ("X", "—", " ")

В процессе выполнения программы объекты исполнителя (константы и переменные) имеют некоторое значение — только одно в каждый момент времени, и оно обязательно лежит в допустимом множестве значений. Например, когда при выполнении программы, приведенной в предыдущем пункте, **ПУТНИК** первый раз оказался у восточной стены,

объект **ПОЛОЖЕНИЕ** принял значение (2,50),

объект **НАПРАВЛЕНИЕ** принял значение **ВОСТОК**.

Заметим, что имена объектов и их типы фиксируются на этапе изготовления исполнителя. По предписанию **НАЧАТЬ РАБОТУ** объекты исполнителя обычно получают некоторые начальные значения (какие именно — предусматривается при создании исполнителя или указывается в предписании **НАЧАТЬ РАБОТУ**). В результате выполнения последующих предписаний значения некоторых объектов (переменных) будут меняться.

Так например, после выполнения предписания **ПОВЕРНИСЬ НА ЮГ** переменная **НАПРАВЛЕНИЕ** получит значение **ЮГ**, а переменная **ПОЛОЖЕНИЕ** не изменит своего значения.

Объекты бывают разные по своей значимости для исполнителя. Некоторые он должен помнить на протяжении всей работы. Все объекты, о которых мы говорили, являются именно такими. Но бывают и другие, которые важны лишь во время выполнения одного предписания. Так, можно предположить, что во время выполнения предписания **СДЕЛАЙ 16 ШАГОВ** где-то хранится число сделанных шагов и после каждого шага проверяется, не превысило ли оно 16. Как только предписание выполнилось, это число больше нас не интересует.

Объекты, необходимые исполнителю на протяжении всей работы программы, называются объектами исполнителя или **глобальными объектами**.

Объекты, представляющие интерес лишь во время выполнения отдельного предписания, называются объектами предписания или **локальными объектами**.

Параметры

Вспомним систему предписаний исполнителя ПУТНИК. Чтобы передвинуть его на 5 шагов, мы должны были выдать ему 2 предписания:

Можно, однако, подойти к этому вопросу иначе и объявить, что есть только одно предписание СДЕЛАЙ ... ШАГОВ, допускающее подстановку вместо ... любого натурального числа (для нашей задачи это число будет находиться в пределах от 1 до 51). Чтобы подчеркнуть, что мы при этом употребляем не 51 различное предписание, а всего одно, мы будем заключать в угловые скобки заданное значение числа шагов:

**СДЕЛАЙ 〈48〉 ШАГОВ
СДЕЛАЙ 〈30〉 ШАГОВ.**

Такое предписание, которое выполняет не одно действие, а целый ряд однотипных действий, называется предписанием с *параметрами*. Параметр — это переменная, которая может принимать различные значения. Для совершения одного конкретного действия из ряда однотипных мы должны указать значение этой переменной. Чтобы подчеркнуть, что значения параметров мы указываем сами, мы будем называть такие параметры *входными*. Бывают также предписания с выходными параметрами, но о них мы сейчас говорить не будем.

Выписывая такое предписание в системе предписаний, мы указываем, в каком месте предписания должны быть заданы параметры, и приводим имена переменных, являющихся параметрами. Чтобы подчеркнуть, что это входные параметры, мы будем указывать слово «вх:».

Вернемся к предписанию, позволяющему ПУТНИКУ делать любое число шагов. Эту переменную величину — число шагов — назовем РАССТОЯНИЕ. В нашей задаче РАССТОЯНИЕ будет объектом целого типа, а соответствующее предписание в системе предписаний будет записано следующим образом:

СДЕЛАЙ <ВХ:РАССТОЯНИЕ> ШАГОВ.

В системе предписаний исполнителя ПУТНИК есть еще одна серия однотипных предписаний:

ПОВЕРНИСЬ НА СЕВЕР
ПОВЕРНИСЬ НА ЗАПАД
ПОВЕРНИСЬ НА ЮГ
ПОВЕРНИСЬ НА ВОСТОК

Эти 4 предписания можно заменить одним предписанием с параметром:

ПОВЕРНИСЬ НА <ВХ:НАПРАВЛЕНИЕ>

Тогда в программе мы могли бы использовать его так:

**ПОВЕРНИСЬ НА <ЮГ>
ПОВЕРНИСЬ НА <ВОСТОК>**

Внимание, после таких обобщений мы будем иметь дело уже с другим исполнителем ПУТНИК-2, обладающим другой системой предписаний.

Система предписаний исполнителя ПУТНИК-2:

1. НАЧНИ РАБОТУ В КВАДРАНТЕ С КОДОМ XXXXXX
2. СДЕЛАЙ <ВХ:РАССТОЯНИЕ> ШАГОВ
3. ПОВЕРНИСЬ НА <ВХ:НАПРАВЛЕНИЕ>
4. КОНЧИ РАБОТУ

Заметим, что предписание может содержать несколько параметров. Так, мы могли бы объединить 2 приказа ПУТНИКу на выполнение разных действий в один приказ следующим образом:

**СДЕЛАЙ <ВХ:РАССТОЯНИЕ> ШАГОВ НА
<ВХ:НАПРАВЛЕНИЕ>,**

и в программе могли бы писать

**СДЕЛАЙ <48> ШАГОВ НА <ВОСТОК>
СДЕЛАЙ <30> ШАГОВ НА <ЗАПАД>**

Это была бы программа для исполнителя ПУТНИК-3, обладающего следующей системой предписаний:

1. НАЧНИ РАБОТУ В КВАДРАНТЕ С КОДОМ XXXXXX
2. СДЕЛАЙ <ВХ:РАССТОЯНИЕ> ШАГОВ НА
<ВХ:НАПРАВЛЕНИЕ>
3. КОНЧИ РАБОТУ

Задание для исполнителя ПУТНИК

Исполнитель ПУТНИК (система предписаний его не содержит параметров) двигается по лабиринту размером 51×28 . Со всех сторон лабиринт окружен барьером, в котором есть проходы слева вверху (вход) и справа внизу (выход). Внутри барьера содержится 7 свободных полос и 6 полос с препятствиями. Высота каждой полосы равна 2. Шестизначный код в предписании НАЧНИ РАБОТУ ... задает конфигурации шести полос препятствий так, что очередная цифра отвечает за очередную полосу (цифра 5 на третьем месте означает, что третья полоса имеет конфигурацию вида 5).

Виды конфигураций следующие:

- 1 — 26 клеток занятых, 23 свободных
- 2 — 19 клеток свободных, 30 занятых
- 3 — 4 клетки свободных, 43 занятых, 2 свободных

- 4 — 19 клеток занятых, 3 свободных, 27 занятых
 5 — 2 клетки занятых, 3 свободных, 27 занятых, 9 свободных,
 8 занятых
 6 — 2 клетки занятых, 5 свободных, 37 занятых, 5 свободных
 7 — 4 клетки свободных, 22 занятых, 11 свободных, 8 занятых,
 4 свободных
 8 — 6 клеток занятых, 3 свободных, 25 занятых, 3 свободных,
 12 занятых
 9 — 3 клетки свободных, 5 занятых, 6 свободных, 31 занятая,
 3 свободных
 0 — 49 клеток свободных

Пример. Лабиринт, приведенный в пункте «Алгоритмы, исполнители, предписания», имеет код 234561.

Задание — провести ПУТНИКА по лабиринту с указанным кодом.

- | | | |
|----------------|----------------|----------------|
| 1. код 765432 | 2. код 135798 | 3. код 687542 |
| 4. код 357981 | 5. код 246875 | 6. код 468752 |
| 7. код 684752 | 8. код 421386 | 9. код 123456 |
| 10. код 234567 | 11. код 456789 | 12. код 864213 |
| 13. код 642138 | 14. код 975324 | 15. код 951462 |

Указание. После выполнения программы, состоящей из двух предписаний:

**НАЧНИ РАБОТУ В КВАДРАНТЕ С КОДОМ 951462
КОНЧИ РАБОТУ**

мы получим напечатанную картинку соответствующего лабиринта.

Исполнитель СТЕКОВЫЙ КАЛЬКУЛЯТОР

Исполнитель СТЕКОВЫЙ КАЛЬКУЛЯТОР производит вычисления над целыми числами, которые хранятся в его памяти. Эта память организована в виде так называемого стека. В начале работы со стеком в нем нет никаких элементов, стек пуст. Мы можем положить в стек некоторый элемент, он окажется на вершине стека. Помещение в стек нескольких элементов происходит следующим образом:

=====	—v ==1==	(—v ==7==
=====	=1=	(—v ==3==
	=====	=7=
		=1=
		=====

НАЧАТЬ ПОЛОЖИТЬ ПОЛОЖИТЬ ПОЛОЖИТЬ
РАБОТУ 1 В СТЕК 7 В СТЕК 3 В СТЕК

Над числами, хранящимися в стеке, СТЕКОВЫЙ КАЛЬКУЛЯТОР может производить вычисления. В вычислениях всегда участвуют два верхних числа. В результате выполнения операции они из стека исчезают, а вместо них на вершину стека помещается результат операции. Таким образом, после выпол-

нения любой арифметической операции количество элементов в стеке уменьшается на единицу.

Когда над элементами стека производятся операции типа сложения и умножения, порядок слагаемых или сомножителей не играет роли. Для операций вычитания и деления это не так. Для правильного выполнения этих операций числа нужно помещать в стек в том порядке, в каком они встречаются в формуле:

Вычислить 3—7:

== 3 ==
=====

== 7 ==
= 3 =
=====

== -4 ==
=====

ПОЛОЖИТЬ 3 В СТЕК ПОЛОЖИТЬ 7 В СТЕК ВЫЧЕСТЬ

Вычислить 8/4:

== 8 ==
=====

== 4 ==
= 8 =
=====

== 2 ==
=====

ПОЛОЖИТЬ 8 В СТЕК ПОЛОЖИТЬ 4 В СТЕК РАЗДЕЛИТЬ

Замечание. Если деление нацело невозможно, то в качестве результата берется целая часть частного. Например,
результат $8/3$ равен 2,
результат $-8/3$ равен -2 .

Приказ ПОЛОЖИТЬ 8 В СТЕК записывается слишком длинно, и мы будем пользоваться более короткой записью «8—>С», имея в виду, что два символа «минус» и «больше» должны восприниматься как один символ «стрелка».

Для арифметических действий мы будем пользоваться либо словесным указанием (СЛОЖИТЬ, ВЫЧЕСТЬ, ...), либо знаком арифметической операции (+, —, ...). Таким образом, возникают пары предписаний, вызывающих одинаковые действия.

Выпишем систему предписаний исполнителя СТЕКОВЫЙ КАЛЬКУЛЯТОР:

предписание	действие
1. НАЧАТЬ ВЫЧИСЛЕНИЕ	включить исполнитель, очистить стек
2. ПОКАЗАТЬ РЕЗУЛЬТАТ	распечатать вершину стека
3. +	
4. СЛОЖИТЬ	{ сложить два верхних числа, заменить их на результат
5. —	
6. ВЫЧЕСТЬ	{ вычесть верхнее число из второго сверху, заменить их на результат
7. *	
8. УМНОЖИТЬ	{ перемножить два верхних числа, заменить их на результат

9. /	
10. РАЗДЕЛИТЬ	$\left. \begin{array}{l} \text{разделить второе сверху число на верхнее, заменить их} \\ \text{на результат} \end{array} \right\}$
11. 0->C	
12. 1->C	
13. 2->C	
14. 3->C	
15. 4->C	записать
16. 5->C	в
17. 6->C	стек
18. 7->C	указанное
19. 8->C	число
20. 9->C	
21. 10->C	

22. КОНЧИТЬ ВЫЧИСЛЕНИЕ выключить исполнитель

Отказы исполнителя СТЕКОВЫЙ КАЛЬКУЛЯТОР:

1. Попытка посмотреть результат при пустом стеке.
2. Попытка выполнить арифметическую операцию, когда в стеке меньше двух чисел.
3. Попытка выполнить операцию деления, когда делитель равен нулю.

Замечание. При реализации стека в ЭВМ он имеет какую-то максимально возможную, ограниченную емкость, например, вмещает не более 100 элементов. При попытке поместить в такой стек 101-й элемент также произойдет отказ.

Сделаем несколько общих замечаний. Программа не будет выполнена при наличии в ней несуществующих предписаний. Кроме того, она не будет выполнена при отсутствии предписания НАЧАТЬ РАБОТУ. Ответ на вопрос о том, будет ли выполнена программа при отсутствии в ней предписания КОНЧИТЬ РАБОТУ, зависит от конкретного устройства посредника между нашей программой и исполнителем — в одних случаях будет, в других нет.

Задание для исполнителя СТЕКОВЫЙ КАЛЬКУЛЯТОР

Напишите программу для исполнителя СТЕКОВЫЙ КАЛЬКУЛЯТОР, выполняющую вычисления по приведенной формуле и печатающую результат:

1. $6 + (5 + 4) * 3 - 2 * 1 = 30$
2. $1 + (2 + 3 + 4) * 5 - 6 = 38$
3. $(5 + 4) / 3 - (12 - 7) * 2 + 10 =$
4. $i + 2 * (3 + 4) - 18 / 3 = 5$
5. $6 / 2 + 3 * (4 + 1) = 13$
6. $1 - 8 / 4 + (2 + 3) * 4 = 13$
7. $(5 - 3) * 2 + (11 + 4) / 5 =$

8. $3 + 2 * (12 - 9) + 2 * 3 - 7$
9. $(1 + 2) * (3 + 4) - 24 / 2$
10. $4 + (2 + 1) * (4 - 2) / 3 + 16 / 4$
11. $2 + 3 + (11 - 9) * (1 + 2)$
12. $12 / 4 - (1 + 2) * 4 + 7 * 3$
13. $1 + 18 / (4 + 2) - 3 * 3 + 18$
14. $15 / (2 + 3) - (6 - 2) / 2 + 13$
15. $(7 + 8) / (6 - 1) - 2 * 4 + 20$

Обратите внимание, что в системе предписаний нашего СТЕКОВОГО КАЛЬКУЛЯТОРА нет предписаний, помещающих в стек двузначные числа (кроме числа 10).

Указание. Подсчитать каждое из этих выражений можно не единственным способом. Мы примем следующий алгоритм подсчета: просматриваем выражение слева направо. Увидев число, кладем его в стек. Увидев знак операции, запоминаем его и, возможно, выполняем ранее запомненные операции с учетом правила старшинства операций.

Образец выполнения задания для исполнителя СТЕКОВЫЙ КАЛЬКУЛЯТОР

$$15. (7 + 8) / (6 - 1) - 2 * 4 + 20$$

НАЧАТЬ ВЫЧИСЛЕНИЕ

„
.. стек пуст

„
7—>C

8—>C

СЛОЖИТЬ

„
.. на вершине — значение первой скобки

„
6—>C

1—>C

ВЫЧЕСТЬ

„
.. на вершине — значение второй скобки

„
РАЗДЕЛИТЬ

„
.. на вершине — результат деления первой скобки на вторую

„
2—>C

4—>C

УМНОЖИТЬ

„
.. на вершине — результат умножения

„

ВЫЧЕСТЬ

„ „ на вершине — результат вычитания

„ „

2—>С

10—>С

УМНОЖИТЬ

„ „

„ „ на вершине — число 20

„ „

СЛОЖИТЬ

„ „

„ „ на вершине — окончательный результат

„ „

ПОКАЗАТЬ РЕЗУЛЬТАТ

КОНЧИТЬ ВЫЧИСЛЕНИЕ

В результате выполнения этой программы будет напечатано
число 15.

Исполнитель РЕДАКТОР СЛОВА

Система предписаний:

1. НАЧАТЬ РАБОТУ СО СЛОВОМ <ВХ:СЛОВО>
2. КУРСОР В ПРАВО НА <ВХ:ЧИСЛО> БУКВ
3. КУРСОР В ЛЕВО НА <ВХ:ЧИСЛО> БУКВ
4. УДАЛИТЬ <ВХ:ЧИСЛО> БУКВ
5. ПЕРЕСТАВИТЬ БУКВЫ
6. КОНЧИТЬ РАБОТУ

РЕДАКТОР работает со «словом». У него есть курсор, указывающий на отмеченную позицию в слове. По предписанию НАЧАТЬ РАБОТУ ... курсор устанавливается над первой позицией. РЕДАКТОР СЛОВА умеет:

- перемещать курсор;
- удалять буквы, начиная с отмеченной, после чего оставшаяся часть слова сдвигается налево;
- переставлять отмеченную букву с ее правой соседкой.

При удалении букв позиция курсора обычно меняется. Единственное исключение составляет случай, когда удаляется самая правая буква. Тогда курсор перемещается на одну позицию влево, т.е. устанавливается над последней оставшейся буквой.

Числа в предписаниях 2—4 — целые однозначные (от 1 до 9).

Исполнитель РЕДАКТОР СЛОВА производит действия над глобальным объектом сложной структуры (слово + номер позиции, над которой стоит курсор). В следующей таблице на примере показано состояние глобального объекта после выполнения каждого предписания.

4. ДАНО: РОБОТ находится рядом с западным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
5. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на восток
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
6. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на восток
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
7. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на юг
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
8. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на юг
ПОЛУЧИТЬ: РОБОТ обошел препятствия; площадь подсчитана
9. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
10. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
11. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
12. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
13. ДАНО: РОБОТ находится рядом с восточным верхним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
14. ДАНО: РОБОТ находится рядом с восточным верхним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
15. ДАНО: РОБОТ находится рядом с восточным верхним краем препятствия лицом на юг
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан

Образец выполнения задания. Вариант 15

Обходить препятствие будем по часовой стрелке. Для исключения неопределенности при описании положения РОБОТА, находящегося около угла препятствия, договоримся, что слово «рядом» будет означать «в свободной клетке справа или слева, на той же горизонтали», а «под» и «над» — «на клетку южнее» или «на клетку севернее».

ПРОГРАММА ОБХОДА ПРЕПЯТСТВИЯ С ПОДСЧЕТОМ ПЕРИМЕТРА
ДАНО: ..РОБОТ НАХОДИТСЯ РЯДОМ С ВОСТОЧНЫМ ВЕРХНИМ
..КРАЕМ ПРЕПЯТСТВИЯ ЛИЦОМ НА ЮГ

- УДАЛИТЬ <7> БУКВ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
2. НАЧАТЬ РАБОТУ СО СЛОВОМ <МАСТОДОНТ>
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <4> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
3. НАЧАТЬ РАБОТУ СО СЛОВОМ <ОБЕЗЬЯНА>
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВЛЕВО НА <2> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
4. НАЧАТЬ РАБОТУ СО СЛОВОМ <КОЛИБРИ>
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <3> БУКВЫ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
5. НАЧАТЬ РАБОТУ СО СЛОВОМ <АНТИЛОПА>
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВПРАВО НА <5> БУКВ

УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ

6. НАЧАТЬ РАБОТУ СО СЛОВОМ <ЕДИНОРОГ>
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <4> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <2> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
7. НАЧАТЬ РАБОТУ СО СЛОВОМ <БРОНЕНОСЕЦ>
КУРСОР ВПРАВО НА <3> БУКВЫ
УДАЛИТЬ <3> БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ

8. НАЧАТЬ РАБОТУ СО СЛОВОМ <ПАНГОЛИН>
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
9. НАЧАТЬ РАБОТУ СО СЛОВОМ <БУРЕВЕСТНИК>
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <4> БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
10. НАЧАТЬ РАБОТУ СО СЛОВОМ <ПТЕРОДАКТИЛЬ>
УДАЛИТЬ <3> БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ

КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ

11. НАЧАТЬ РАБОТУ СО СЛОВОМ <СТЕГОЗАВР>
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <4> БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
12. НАЧАТЬ РАБОТУ СО СЛОВОМ <БРОНТОЗАВР>
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <3> БУКВЫ
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <3> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
13. НАЧАТЬ РАБОТУ СО СЛОВОМ <ОРАНГУТАНГ>
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <3> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <4> БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ
14. НАЧАТЬ РАБОТУ СО СЛОВОМ <КРОКОДИЛ>
КУРСОР ВПРАВО НА <2> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВПРАВО НА <1> БУКВУ

УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ

15. НАЧАТЬ РАБОТУ СО СЛОВОМ <КРЕВЕТКА>
УДАЛИТЬ <1> БУКВУ
КУРСОР ВПРАВО НА <1> БУКВУ
УДАЛИТЬ <2> БУКВЫ
КУРСОР ВПРАВО НА <3> БУКВЫ
УДАЛИТЬ <1> БУКВУ
КУРСОР ВЛЕВО НА <2> БУКВЫ
ПЕРЕСТАВИТЬ БУКВЫ
КУРСОР ВЛЕВО НА <1> БУКВУ
ПЕРЕСТАВИТЬ БУКВЫ
КОНЧИТЬ РАБОТУ

Задание № 2 для исполнителя РЕДАКТОР СЛОВА

Напишите программу для исполнителя РЕДАКТОР СЛОВА, преобразующую слово следующим образом:

- | | | |
|-----------------|----|--------|
| 1. шерстокрыл | —> | крот |
| 2. трубкозуб | —> | зубр |
| 3. жаворонок | —> | норка |
| 4. сизоворонка | —> | сова |
| 5. каравайка | —> | рак |
| 6. ласточка | —> | косач |
| 7. спаниэль | —> | сип |
| 8. шилоклювка | —> | вол |
| 9. альбатрос | —> | барс |
| 10. игуанодон | —> | динго |
| 11. трицератопс | —> | птица |
| 12. стрекоза | —> | треска |
| 13. опоссум | —> | сом |
| 14. каракатица | —> | кит |
| 15. осьминог | —> | сиг |

Задание № 3 для исполнителя РЕДАКТОР СЛОВА

С помощью исполнителя РЕДАКТОР СЛОВА исправьте приведенные слова таким образом, чтобы они составляли правильные термины, используемые в программировании:

1. SURBOUTINEE
2. PARAMMETRE
3. PROCCEDRUE
4. COMMUTATITION
5. LOGIYCALE
6. INETGGER

7. FUNNCTOIN
8. CALCULLATORE
9. MAGGNEYTIC
10. DISCPLAYE
11. OPREATORE
12. LOCCATIONCE
13. TRANLSATER
14. VARYIABELE
15. IDETNIFICATIONE

Указание. Правильные термины найдите по словарю, имея в виду, что первые две буквы в каждом слове — верные.

Задание № 4 для исполнителя РЕДАКТОР СЛОВА

Измените исполнитель РЕДАКТОР СЛОВА (составьте новую систему предписаний) так, чтобы с его помощью можно было из слова «компьютер» сделать слова «компот» и «тремок», но так, чтобы нельзя было сделать вообще любое слово.

Исполнитель ЧЕРТЕЖНИК

Система предписаний:

1. НАЧАТЬ РАБОТУ С ПОЛЕМ <ВХ:ДЛИНА, ШИРИНА>
2. ДВИГАТЬСЯ НА <ВХ:РАССТОЯНИЕ>
НА <ВХ:НАПРАВЛЕНИЕ>
3. РИСОВАТЬ НА <ВХ:РАССТОЯНИЕ>
НА <ВХ:НАПРАВЛЕНИЕ>
4. КОНЧИТЬ РАБОТУ

Исполнитель ЧЕРТЕЖНИК работает с прямоугольным полем, разбитым на клетки. Размер поля в клетках задается в предписании НАЧАТЬ РАБОТУ ... двумя двузначными числами. Он не может превышать 75 по горизонтали и 20 по вертикали. По предписанию НАЧАТЬ РАБОТУ ... ЧЕРТЕЖНИК располагается над клеткой с координатами (1,1) (мы будем говорить также, что координаты пера становятся (1,1)).

ЧЕРТЕЖНИК умеет рисовать и двигаться на целое число клеток в одном из восьми направлений (восток, северо-восток, север и т.д.). Обратите внимание, что НАПРАВЛЕНИЕ ЧЕРТЕЖНИКА отличается от НАПРАВЛЕНИЯ ПУТНИКА, которое имеет только 4 значения. По предписанию ДВИГАТЬСЯ ... перо переносится в клетку, отстоящую от предыдущей на указанное число клеток в нужном направлении. По предписанию РИСОВАТЬ ... ЧЕРТЕЖНИК рисует пунктирную линию (из звездочек) в указанном направлении, причем первая звездочка рисуется в клетке, от которой начинается движение. Первый параметр предписания РИСОВАТЬ ... задает количество звездочек в линии. Заметим, что при такой реализации

действий «двигаться» и «рисовать» координаты пера после выполнения соответствующих предписаний с одинаковыми параметрами будут разными.

В следующей таблице на примере показано значение координат текущих клеток после выполнения каждого предписания некоторой программы.

Предписание	Координаты пера после выполнения предписания
НАЧАТЬ РАБОТУ С ПОЛЕМ <10,08>	не существует
ДВИГАТЬСЯ НА <6> НА <СЕВЕР>	(1,1)
ДВИГАТЬСЯ НА <2> НА <ВОСТОК>,	(1,7)
РИСОВАТЬ НА <5> НА <ЮГ>	(3,7)
РИСОВАТЬ НА <5> НА <СЕВЕРО-ВОСТОК>	(3,3)
РИСОВАТЬ НА <5> НА <ЮГ>	(7,7)
КОНЧИТЬ РАБОТУ	(7,3)
	не существует

В результате выполнения этой программы будет нарисована буква «И»:

```

===== ===== =====
= * * *
= * * * *
5= * * * *
= ** * *
= * * *
= =
===== ===== =====

```

5 10

Отказы исполнителя ЧЕРТЕЖНИК:

- Сделана попытка вывести перо за границы поля.
- Задано недопустимое значение параметров ДЛИНА, ШИРИНА или РАССТОЯНИЕ (не являющееся натуральным числом или выходящее за допустимые границы, например ДЛИНА больше 75).
- Задано недопустимое значение параметра НАПРАВЛЕНИЕ (не совпадающее ни с одним из 8 географических направлений).

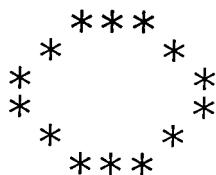
Кроме того, программа не будет работать, если в ней: содержатся недопустимые предписания; нет предписания НАЧАТЬ РАБОТУ.

Задание для исполнителя ЧЕРТЕЖНИК

Напишите программу, в результате выполнения которой на листе 16×16 будет нарисовано слово

- | | | | | |
|--------|--------|--------|---------|---------|
| 1. нос | 4. сок | 7. лот | 10. кот | 13. тис |
| 2. сон | 5. тик | 8. вол | 11. тон | 14. тир |
| 3. ток | 6. кит | 9. лов | 12. рис | 15. рог |

Указание. Сначала нужно нарисовать такой рисунок на листе бумаги в клетку и подсчитать все расстояния. При этом придется криволинейные элементы букв заменить ломаными линиями. Пример — возможное изображение буквы «О»:



Образец выполнения задания

Вариант 15 — нарисовать слово «рог».

Будем рисовать буквы размером 4×6 , оставляя по 5 клеток сверху и снизу, и по одной клетке между буквами.

НАЧАТЬ РАБОТУ С ПОЛЕМ $\langle 16,16 \rangle$

ДВИГАТЬСЯ НА $\langle 1 \rangle$ НА $\langle \text{ВОСТОК} \rangle$

ДВИГАТЬСЯ НА $\langle 5 \rangle$ НА $\langle \text{СЕВЕР} \rangle$

„

„ подошли к нижнему краю буквы «р»

РИСОВАТЬ НА $\langle 6 \rangle$ НА $\langle \text{СЕВЕР} \rangle$

РИСОВАТЬ НА $\langle 4 \rangle$ НА $\langle \text{ВОСТОК} \rangle$

РИСОВАТЬ НА $\langle 4 \rangle$ НА $\langle \text{ЮГ} \rangle$

РИСОВАТЬ НА $\langle 4 \rangle$ НА $\langle \text{ЗАПАД} \rangle$

„

„ нарисовали букву «р»

„

ДВИГАТЬСЯ НА $\langle 5 \rangle$ НА ВОСТОК

ДВИГАТЬСЯ НА $\langle 2 \rangle$ НА $\langle \text{ЮГ} \rangle$

„

„ подошли к левому нижнему углу буквы «о»

„

РИСОВАТЬ НА $\langle 6 \rangle$ НА $\langle \text{СЕВЕР} \rangle$

РИСОВАТЬ НА $\langle 4 \rangle$ НА $\langle \text{ВОСТОК} \rangle$

РИСОВАТЬ НА <6> НА <ЮГ>
РИСОВАТЬ НА <4> НА <ЗАПАД>

„ нарисовали букву «о» (в форме прямоугольника)

„ ДВИГАТЬСЯ НА <5> НА <ВОСТОК>

„

„ подошли к нижнему краю буквы «г»

РИСОВАТЬ НА <6> НА <СЕВЕР>
РИСОВАТЬ НА <4> НА <ВОСТОК>

„

„ нарисовали букву «г»

„ КОНЧИТЬ РАБОТУ

ТЕМА 2. ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ СВЕРХУ—ВНИЗ

Теоретический материал

При изучении предыдущей темы мы познакомились с некоторыми исполнителями. Мы решали с их помощью задачи, алгоритм решения которых был не слишком сложным, а программы получались короткие (полтора—два десятка строк). Но даже при составлении таких небольших программ нередко делаются ошибки.

Задачи, встречающиеся в жизни, намного сложнее.

Решение сложных задач при помощи простых исполнителей требует изощренных алгоритмов и приводит к длинным программам (длиной в сотни, тысячи и даже сотни тысяч строк).

Если такую программу писать без всяких правил, то

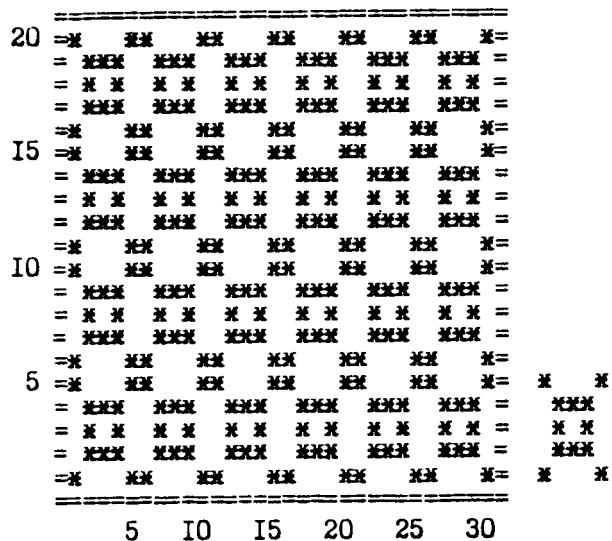
- сделанные в ней ошибки будет очень трудно найти;
- окажется, что никто, кроме автора, ее не поймет;
- в такую программу будет очень трудно вносить изменения.

Тем не менее большие программы пишутся и успешно работают. В процессе работы над сложными программами выработался специальный подход, который называется «исходящее проектирование» или «технология программирования сверху—вниз». Мы познакомимся с этим подходом на примере.

Рассмотрим задачу: нарисовать изображенный ниже орнамент на поле 30×20 (см. с. 31).

Заметим, что этот орнамент состоит из повторяющихся клеток размером 5×5 . Рисунок внутри каждой клетки представляет собой квадрат, дополненный четырьмя отрезками, расположенным по диагоналям (одна из таких клеток нарисована рядом отдельно).

На некоторых печатающих устройствах или дисплеях наша фигура 3×3 будет не квадратом, а прямоугольником, но мы для простоты все равно будем называть ее квадратом.



Эту задачу можно решить, нарисовав 24 раза такую клетку. Поскольку клетки по-разному расположены друг относительно друга, придется при этом внимательно следить за перемещением пера между рисованием клеток. Однако можно разбить чертеж на более крупные части, например на 6 одинаковых вертикальных полос. Задача «изобразить полосу» не зависит от того, какая это полоса, если условиться начинать рисование с определенной точки на полосе (назовем ее «начало»). При таком разбиении задачи на части последовательность наших действий становится более простой.

ПРОГРАММА РИСОВАНИЯ ОРНАМЕНТА
 НАЧАТЬ РИСОВАНИЕ
 ИЗОБРАЗИТЬ ПОЛОСУ
 ВСТАТЬ В НАЧАЛО ВТОРОЙ ПОЛОСЫ
 ИЗОБРАЗИТЬ ПОЛОСУ
 ВСТАТЬ В НАЧАЛО ТРЕТЬЕЙ ПОЛОСЫ
 ИЗОБРАЗИТЬ ПОЛОСУ
 ВСТАТЬ В НАЧАЛО ЧЕТВЕРТОЙ ПОЛОСЫ
 ИЗОБРАЗИТЬ ПОЛОСУ
 ВСТАТЬ В НАЧАЛО ПЯТОЙ ПОЛОСЫ
 ИЗОБРАЗИТЬ ПОЛОСУ
 ВСТАТЬ В НАЧАЛО ШЕСТОЙ ПОЛОСЫ
 ИЗОБРАЗИТЬ ПОЛОСУ
 КОНЧИТЬ РИСОВАНИЕ
 КОНЕЦ ПРОГРАММЫ

(Здесь подразумевается, что по предписанию НАЧАТЬ РИСОВАНИЕ перо установится в начало первой полосы).

Для того, чтобы получить право так писать, нам необходим исполнитель, имеющий все перечисленные предписания в своей системе предписаний. Назовем его ИЗОБРАЗИТЕЛЬ ПОЛОС-1 и выпишем его систему предписаний (мы будем употреблять сокращение СП:)

СП:

1. НАЧАТЬ РИСОВАНИЕ
2. ИЗОБРАЗИТЬ ПОЛОСУ
3. ВСТАТЬ В НАЧАЛО ВТОРОЙ ПОЛОСЫ
4. ВСТАТЬ В НАЧАЛО ТРЕТЬЕЙ ПОЛОСЫ
5. ВСТАТЬ В НАЧАЛО ЧЕТВЕРТОЙ ПОЛОСЫ
6. ВСТАТЬ В НАЧАЛО ПЯТОЙ ПОЛОСЫ
7. ВСТАТЬ В НАЧАЛО ШЕСТОЙ ПОЛОСЫ
8. КОНЧИТЬ РИСОВАНИЕ

Заметим, однако, что поскольку мы будем рисовать полосу всегда одинаково, то и перемещение пера между полосами тоже должно быть одинаковым. Введя понятия «начало полосы» и «конец полосы», можно сказать, что при изображении каждой полосы перо перемещается из ее «начала» в ее «конец», а между рисованием перо перемещается из «конца» предыдущей в «начало» следующей одинаковым образом.

Итак, нам понадобится несколько другой исполнитель ИЗОБРАЗИТЕЛЬ ПОЛОС со следующей системой предписаний:

СП:

1. НАЧАТЬ РИСОВАНИЕ
2. ИЗОБРАЗИТЬ ПОЛОСУ
3. ВСТАТЬ В НАЧАЛО ПОЛОСЫ
4. КОНЧИТЬ РИСОВАНИЕ

Одновременно упростится и программа:

ПРОГРАММА РИСОВАНИЯ ОРНАМЕНТА

```
НАЧАТЬ РИСОВАНИЕ
ИЗОБРАЗИТЬ ПОЛОСУ
ВСТАТЬ В НАЧАЛО ПОЛОСЫ
ИЗОБРАЗИТЬ ПОЛОСУ
КОНЧИТЬ РИСОВАНИЕ
КОНЕЦ ПРОГРАММЫ
```

Отметим важный момент. Сейчас мы не продумываем, как именно полосы будут рисоваться. То, что мы действительно сделали — это:

1 — разбили задачу на более простые части;

2 — продумали последовательность выполнения частей;
3 — составили систему предписаний нужного нам исполнителя;
4 — продумали, что должно быть сделано в результате выполнения каждого предписания, в том числе (что очень важно), где должно находиться перо до и после его выполнения.

Проделать пункты 3 и 4 — это значит составить *внешнее описание* исполнителя, нужного для решения задачи на данном этапе. Такой этап в решении задачи (п. 1—4) называется *шагом декомпозиции* задачи (слово «декомпозиция» означает «разбиение» и в данном случае понимается как разбиение процесса решения задачи на более простые этапы).

Возвратимся к нашему примеру. Выполняя п. 1 декомпозиции, мы свели задачу рисования орнамента к рисованию отдельных полос. Выполняя п. 2, мы составили программу для исполнителя ИЗОБРАЗИТЕЛЬ ПОЛОС. Затем (как требуется в п. 3) составили систему предписаний нового исполнителя. Покажем теперь, какова стандартная запись п. 4 шага декомпозиции на языке МИНИ.

ПРЕДПИСАНИЕ НАЧАТЬ РИСОВАНИЕ

ДАНО: —————
ПОЛУЧИТЬ: ЧЕРТЕЖНИК ВКЛЮЧЕН; ПЕРО
В НАЧАЛЕ ПЕРВОЙ ПОЛОСЫ

КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПОЛОСУ
ДАНО: ПЕРО В НАЧАЛЕ ПОЛОСЫ
ПОЛУЧИТЬ: ПОЛОСА НАРИСОВАНА;
ПЕРО В КОНЦЕ ПОЛОСЫ

КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО ПОЛОСЫ
ДАНО: ПЕРО В КОНЦЕ НАРИСОВАННОЙ
ПОЛОСЫ
ПОЛУЧИТЬ: ПЕРО В НАЧАЛЕ СЛЕДУЮЩЕЙ ПОЛОСЫ

КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ КОНЧИТЬ РИСОВАНИЕ
ДАНО: ОРНАМЕНТ НАРИСОВАН
ПОЛУЧИТЬ: ЧЕРТЕЖНИК ВЫКЛЮЧЕН

КОНЕЦ ПРЕДПИСАНИЯ

Повторим еще раз — здесь написано, *что* надо делать, но не написано *как*.

Проделанный шаг декомпозиции приблизил нас к окончательному решению задачи: рисовать полосу проще, чем рисовать весь орнамент целиком. Мы поняли, какой исполнитель

нам для этого нужен и как им нужно пользоваться, чтобы получить весь орнамент. Трудность состоит в том, что такого исполнителя — ИЗОБРАЗИТЕЛЯ ПОЛОС — у нас нет. Есть другой исполнитель, ЧЕРТЕЖНИК, обладающий меньшими возможностями. Для решения задачи мы должны *реализовать* новый, сложный исполнитель *на базе* того, который существует (*базового исполнителя*), т.е. представить все действия нового исполнителя через последовательность действий ЧЕРТЕЖНИКА.

Предписание ВСТАТЬ В НАЧАЛО ПОЛОСЫ легко реализовать с помощью исполнителя ЧЕРТЕЖНИК, помня о том, что перо при этом должно перемещаться из «конца» предыдущей полосы в «начало» следующей. Мы будем считать «началом» полосы ее левый нижний угол, а «концом» — левый верхний. При таком соглашении получим:

ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО ПОЛОСЫ
ДВИГАТЬСЯ НА <5> НА <ВОСТОК>
ДВИГАТЬСЯ НА <19> НА <ЮГ>
КОНЕЦ ПРЕДПИСАНИЯ

Задача рисования полосы остается все еще сложной для ЧЕРТЕЖНИКА. Чтобы упростить решение задачи, заметим, что полоса состоит из 4 расположенных друг над другом одинаковых клеток. Назовем «началом клетки» ее левый нижний угол, «концом клетки» — левый верхний. Изображая последовательно 4 раза клетку таким образом, чтобы перо перемещалось из «начала» клетки в ее «конец», мы получим полосу, причем между рисованием клеток придется переходить от «конца» предыдущей клетки к «началу» следующей. Заметим, что при таком рисовании «конец» последней клетки будет и «концом» полосы.

Итак, реализация предписания ИЗОБРАЗИТЬ ПОЛОСУ на базе исполнителя ИЗОБРАЗИТЕЛЬ КЛЕТОК будет выглядеть так:

ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПОЛОСУ
ДАНО: ПЕРО В НАЧАЛЕ ПОЛОСЫ
ПОЛУЧИТЬ: ПОЛОСА НАРИСОВАНА;
ПЕРО В КОНЦЕ ПОЛОСЫ
ИЗОБРАЗИТЬ КЛЕТКУ
ВСТАТЬ В НАЧАЛО КЛЕТКИ
ИЗОБРАЗИТЬ КЛЕТКУ
ВСТАТЬ В НАЧАЛО КЛЕТКИ
ИЗОБРАЗИТЬ КЛЕТКУ
ВСТАТЬ В НАЧАЛО КЛЕТКИ
ИЗОБРАЗИТЬ КЛЕТКУ
КОНЕЦ ПРЕДПИСАНИЯ

Для того, чтобы такая запись получила право на существование, необходимо, чтобы у нас был исполнитель ИЗОБРАЗИТЕЛЬ КЛЕТОК. В его системе предписаний должны содер-

жаться предписания ИЗОБРАЗИТЬ КЛЕТКУ, при выполнении которого перо перемещается из «начала» клетки в ее «конец», и ВСТАТЬ В НАЧАЛО КЛЕТКИ, при выполнении которого перо перемещается из «конца» предыдущей клетки в «начало» следующей.

Таким образом, мы провели второй шаг декомпозиции задачи — реализовали исполнитель ИЗОБРАЗИТЕЛЬ ПОЛОС на базе исполнителя ИЗОБРАЗИТЕЛЬ КЛЕТОК и исполнителя ЧЕРТЕЖНИК.

Наконец, изобразить клетку и встать в начало клетки с помощью ЧЕРТЕЖНИКА можно достаточно просто, и здесь нам не понадобятся другие (подчиненные) исполнители.

А теперь покажем, как мы будем записывать программы для сложных задач (с тем, чтобы они выполнялись на ЭВМ).

Сделаем одно замечание. Поскольку во время выполнения программы у нас работают разные исполнители, мы сейчас каждый раз будем указывать, к какому именно исполнителю относится данное предписание. Такая запись, конечно, увеличивает сложность оформления программы, и впоследствии мы от нее откажемся.

ПРОГРАММА РИСОВАНИЯ ОРНАМЕНТА

ИСПОЛНИТЕЛЬ ИЗОБРАЗИТЕЛЬ ПОЛОС

FIGURE
CPI

- 1. НАЧАТЬ РИСОВАНИЕ
 - 2. ИЗОБРАЗИТЬ ПОЛОСУ
 - 3. ВСТАТЬ В НАЧАЛО ПОЛОСЫ
 - 4. КОНЧИТЬ РИСОВАНИЕ

.ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ:
ИЗОБРАЗИТЕЛЬ КЛЕТОК, ЧЕРТЕЖНИК
КОНЕЦ ОПИСАНИЙ

K

. ПРЕДПИСАНИЕ НАЧАТЬ РИСОВАНИЕ

ДАНО: — — — —

. . . ПОЛУЧИТЬ: ЧЕРТЕЖНИК ВКЛЮЧЕН;
ПЕРО В НАЧАЛЕ ПЕРВОЙ ПОЛОСЫ
. . . ЧЕРТЕЖНИК. НАЧАТЬ РАБОТУ С ПОЛЕМ <50,20>
. КОНЕЦ ПРЕДПИСАНИЯ

. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПОЛОСУ
. . ДАНО: ПЕРО В НАЧАЛЕ ПОЛОСЫ
. . . ПОЛУЧИТЬ: ПОЛОСА НАРИСОВАНА; ПЕРО В КОНЦЕ ПОЛОСЫ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ВСТАТЬ В НАЧАЛО КЛЕТКИ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ВСТАТЬ В НАЧАЛО КЛЕТКИ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ВСТАТЬ В НАЧАЛО КЛЕТКИ
. . . ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
. КОНЕЦ ПРЕДПИСАНИЯ

. ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО ПОЛОСЫ
. . ДАНО: ПЕРО В КОНЦЕ НАРИСОВАННОЙ ПОЛОСЫ
. . . ПОЛУЧИТЬ: ПЕРО В НАЧАЛЕ СЛЕДУЮЩЕЙ ПОЛОСЫ
. . . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <5> НА <ВОСТОК>
. . . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <19> НА <ЮГ>
. КОНЕЦ ПРЕДПИСАНИЯ

. ПРЕДПИСАНИЕ КОНЧИТЬ РИСОВАНИЕ
. . ДАНО: ОРНАМЕНТ НАРИСОВАН
. . . ПОЛУЧИТЬ: ЧЕРТЕЖНИК ВЫКЛЮЧЕН
. . . ЧЕРТЕЖНИК. КОНЧИТЬ РАБОТУ
. КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ

ИСПОЛНИТЕЛЬ ИЗОБРАЗИТЕЛЬ КЛЕТОК
.СП:
. . 1. ИЗОБРАЗИТЬ КЛЕТКУ
. . 2. ВСТАТЬ В НАЧАЛО КЛЕТКИ
.ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ: ЧЕРТЕЖНИК
КОНЕЦ ОПИСАНИЙ

. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ КЛЕТКУ
. . ДАНО: ПЕРО В НАЧАЛЕ КЛЕТКИ
. . . ПОЛУЧИТЬ: КЛЕТКА НАРИСОВАНА; ПЕРО В КОНЦЕ КЛЕТКИ
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <2> НА <СЕВЕРО-ВОСТОК>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <СЕВЕР>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ВОСТОК>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ЮГ>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ЗАПАД>
. . .
. . . " нарисовали один отрезок и квадрат
. . .
. . . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <2> НА <ВОСТОК>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <2> НА <ЮГО-ВОСТОК>
. . . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <4> НА <СЕВЕР>

```
. . . ЧЕРТЕЖНИК РИСОВАТЬ НА <2> НА <ЮГО-ЗАПАД>
. . . ЧЕРТЕЖНИК ДВИГАТЬСЯ НА <2> НА <ЗАПАД>
. . . ЧЕРТЕЖНИК РИСОВАТЬ НА <2> НА <СЕВЕРО-ЗАПАД>
. . .
. . . " нарисовали последний отрезок и встали в конец клетки
. КОНЕЦ ПРЕДПИСАНИЯ

. ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО КЛЕТКИ
. . ДАНО: ПЕРО В КОНЦЕ КЛЕТКИ
. . ПОЛУЧИТЬ: ПЕРО В НАЧАЛЕ СЛЕДУЮЩЕЙ КЛЕТКИ
. . ЧЕРТЕЖНИК ДВИГАТЬСЯ НА <1> НА <СЕВЕР>
. КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ
```

Прокомментируем написанное.

1. Этот текст имеет определенную структуру: сначала идет программа, взятая в своеобразные «скобки» — слова ПРОГРАММА — КОНЕЦ ПРОГРАММЫ; затем следуют описания всех сконструированных нами исполнителей, также заключенные в скобки ИСПОЛНИТЕЛЬ — КОНЕЦ ИСПОЛНИТЕЛЯ.

2. Внутри описания каждого исполнителя есть собственно описательная часть, заканчивающаяся словами КОНЕЦ ОПИСАНИЙ, а далее следует реализация всех предписаний с помощью исполнителей более низкого уровня.

3. Реализация каждого предписания также заключена в скобки ПРЕДПИСАНИЕ — КОНЕЦ ПРЕДПИСАНИЯ.

Реализация содержит строки ДАНО и ПОЛУЧИТЬ, которые не вызывают никаких действий. Фактически заполнение этих строк происходит раньше, до реализации — на соответствующем шаге декомпозиции.

4. Правила записи программ: расположение «открывающих» и «закрывающих» скобок друг под другом, вертикальная разметка, сдвиг внутренних строк вправо — помогают выделить в записи программы ее структурные части.

5. Сама программа и реализация всех предписаний получились достаточно короткими и понятными. Это произошло потому, что мы каждый раз конструировали исполнитель с нужными нам возможностями.

6. В конечном счете, все предписания придуманных нами исполнителей реализованы с помощью ЧЕРТЕЖНИКА. В этой задаче он является единственным базовым исполнителем. В более сложных случаях базовых исполнителей может быть несколько.

7. Слова ИСПОЛНИТЕЛЬ—КОНЕЦ ИСПОЛНИТЕЛЯ, ПРЕДПИСАНИЕ—КОНЕЦ ПРЕДПИСАНИЯ являются словами учебного языка МИНИ. Однако действия, необходимые при нисходящем проектировании: разбиение задачи на части, продумывание связей между частями, создание средств для решения

каждой задачи — являются общими, не зависящими от языка и лишь иначе записываются в других языках программирования.

Задание. Упростите программу рисования, изменив предписание ИЗОБРАЗИТЬ ПОЛОСУ так, чтобы в конце рисования полосы мы попадали в начало следующей.

Программа рисования слова

Рассмотрим задачу рисования слова «ток» на поле размером 16×16 . В качестве базового будем использовать исполнитель ЧЕРТЕЖНИК. Заметим, что выполнение задачи состоит из рисования букв и перемещения пера между буквами.

ПРОГРАММА РИСОВАНИЯ СЛОВА «ТОК»

- ЧЕРТЕЖНИК. НАЧАТЬ РАБОТУ С ПОЛЕМ $\langle 16,16 \rangle$
- РАЗМЕСТИТЕЛЬ. ВСТАТЬ В НАЧАЛО СЛОВА
- ИЗОБРАЗИТЕЛЬ БУКВ. ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ
- РАЗМЕСТИТЕЛЬ. ВСТАТЬ В НАЧАЛО БУКВЫ
- ИЗОБРАЗИТЕЛЬ БУКВ. ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ
- РАЗМЕСТИТЕЛЬ. ВСТАТЬ В НАЧАЛО БУКВЫ
- ИЗОБРАЗИТЕЛЬ БУКВ. ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
- ЧЕРТЕЖНИК. КОНЧИТЬ РАБОТУ

КОНЕЦ ПРОГРАММЫ

=====

ИСПОЛНИТЕЛЬ ИЗОБРАЗИТЕЛЬ БУКВ

- СП:
 - 1. ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ
 - 2. ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ
 - 3. ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
- ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ: ЧЕРТЕЖНИК

КОНЕЦ ОПИСАНИЙ

=====

ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ

- ДАНО: ПЕРО НАХОДИТСЯ В ЛЕВОМ НИЖНЕМ УГЛУ ПРЯМОУГОЛЬНИКА, ОТВЕДЕНОГО ДЛЯ БУКВЫ
- ПОЛУЧИТЬ: БУКВА НАРИСОВАНА, ПЕРО В ПРАВОМ НИЖНЕМ УГЛУ ПРЯМОУГОЛЬНИКА
- ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА $\langle 2 \rangle$ НА \langle ВОСТОК \rangle
- ЧЕРТЕЖНИК. РИСОВАТЬ НА $\langle 6 \rangle$ НА \langle СЕВЕР \rangle
- ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА $\langle 1 \rangle$ НА \langle ЗАПАД \rangle
- ЧЕРТЕЖНИК. РИСОВАТЬ НА $\langle 3 \rangle$ НА \langle ВОСТОК \rangle
- ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА $\langle 5 \rangle$ НА \langle ЮГ \rangle

КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ

- ДАНО: ПЕРО НАХОДИТСЯ В ЛЕВОМ НИЖНЕМ УГЛУ ПРЯМОУГОЛЬНИКА, ОТВЕДЕНОГО ДЛЯ БУКВЫ
- ПОЛУЧИТЬ: БУКВА НАРИСОВАНА, ПЕРО В ПРАВОМ НИЖНЕМ УГЛУ ПРЯМОУГОЛЬНИКА
- ЧЕРТЕЖНИК. РИСОВАТЬ НА $\langle 6 \rangle$ НА \langle СЕВЕР \rangle

. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <4> НА <ВОСТОК>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <6> НА <ЮГ>
. . . ЧЕРТЕЖНИК. РИСОВАТЬ НА <4> НА <ЗАПАД>
. . . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <3> НА <ВОСТОК>
. . КОНЕЦ ПРЕДПИСАНИЯ

. . ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
. . ДАНО: ПЕРО НАХОДИТСЯ В ЛЕВОМ НИЖНЕМ УГЛУ
ПРЯМОУГОЛЬНИКА, ОТВЕДЕННОГО ДЛЯ БУКВЫ
. . ПОЛУЧИТЬ: БУКВА НАРИСОВАНА, ПЕРО В ПРАВОМ НИЖНЕМ
УГЛУ ПРЯМОУГОЛЬНИКА
. . ЧЕРТЕЖНИК. РИСОВАТЬ НА <6> НА <СЕВЕР>
. . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <3> НА <ВОСТОК>
. . ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ЮГО-ЗАПАД>
. . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <1> НА <ЮГ>
. . ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ЮГО-ВОСТОК>
. . КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ

"=====

ИСПОЛНИТЕЛЬ РАЗМЕСТИТЕЛЬ

. СП:
. . 1. ВСТАТЬ В НАЧАЛО СЛОВА
. . 2. ВСТАТЬ В НАЧАЛО БУКВЫ
. . ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ: ЧЕРТЕЖНИК
КОНЕЦ ОПИСАНИЙ

"-----

. . ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО СЛОВА
. . ДАНО: РАБОТА НАЧАТА, ПЕРО В ТОЧКЕ (1,1)
. . ПОЛУЧИТЬ: ПЕРО В ЛЕВОМ НИЖНЕМ УГЛУ ПРЯМОУГОЛЬНИКА,
ОТВЕДЕННОГО ДЛЯ ПЕРВОЙ БУКВЫ
. . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <5> НА <СЕВЕР>
. . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <1> НА <ВОСТОК>
. . КОНЕЦ ПРЕДПИСАНИЯ

. . ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО БУКВЫ
. . ДАНО: ПЕРО В ПРАВОМ НИЖНЕМ УГЛУ
ПРЯМОУГОЛЬНИКА, ОТВЕДЕННОГО
ДЛЯ НАРИСОВАННОЙ БУКВЫ
. . ПОЛУЧИТЬ: ПЕРО В ЛЕВОМ НИЖНЕМ УГЛУ
ПРЯМОУГОЛЬНИКА, ОТВЕДЕННОГО ДЛЯ БУКВЫ
. . ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <2> НА <ВОСТОК>
. . КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ

"=====

Задание.

1. Ответьте на следующие вопросы:
— в какой точке расположено «начало слова» (укажите координаты);
— какой прямоугольник отведен для каждой буквы;
— где находится «начало буквы».

2. Изобразите на листе бумаги в клетку результат работы программы.

Задание по модификации программы рисования слова

В приведенной выше программе рисования слова изменено *одно* предписание (разное для разных вариантов). Так, новая программа варианта 1 отличается от приведенной предписанием ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ; новая программа варианта 5 отличается от приведенной предписанием ВСТАТЬ В НАЧАЛО БУКВЫ. Требуется на листе бумаги 16×16 изобразить результат работы измененной программы соответствующего варианта.

1. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ
 - . ДВИГАТЬСЯ НА <5> НА <СЕВЕР>
 - . ДВИГАТЬСЯ НА <3> НА <ВОСТОК>
 - . РИСОВАТЬ НА <4> НА <ЗАПАД>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>КОНЕЦ ПРЕДПИСАНИЯ
2. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>
 - . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . ДВИГАТЬСЯ НА <3> НА <ЗАПАД>
 - . РИСОВАТЬ НА <4> НА <ЮГ>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>
 - . ДВИГАТЬСЯ НА <2> НА <ЮГ>КОНЕЦ ПРЕДПИСАНИЯ
3. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
 - . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . ДВИГАТЬСЯ НА <3> НА <ЮГ>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>
 - . ДВИГАТЬСЯ НА <3> НА <СЕВЕР>
 - . РИСОВАТЬ НА <6> НА <ЮГ>КОНЕЦ ПРЕДПИСАНИЯ
4. ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО СЛОВА
 - . ДВИГАТЬСЯ НА <3> НА <СЕВЕР>
 - . ДВИГАТЬСЯ НА <2> НА <ВОСТОК>КОНЕЦ ПРЕДПИСАНИЯ
5. ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО БУКВЫ
 - . ДВИГАТЬСЯ НА <1> НА <ВОСТОК>
 - . ДВИГАТЬСЯ НА <2> НА <СЕВЕР>КОНЕЦ ПРЕДПИСАНИЯ
6. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ
 - . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>

- . РИСОВАТЬ НА <4> НА <ЮГ>
 - . РИСОВАТЬ НА <4> НА <ЗАПАД>
 - . ДВИГАТЬСЯ НА <3> НА <ЮГО-ВОСТОК>
- КОНЕЦ ПРЕДПИСАНИЯ
7. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ
- . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . ДВИГАТЬСЯ НА <3> НА <ЮГ>
 - . РИСОВАТЬ НА <2> НА <ВОСТОК>
 - . ДВИГАТЬСЯ НА <2> НА <ЮГ>
 - . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . РИСОВАТЬ НА <3> НА <ВОСТОК>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
 - . РИСОВАТЬ НА <3> НА <ЗАПАД>
 - . ДВИГАТЬСЯ НА <2> НА <ВОСТОК>
- КОНЕЦ ПРЕДПИСАНИЯ
8. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
- . РИСОВАТЬ НА <5> НА <СЕВЕР>
 - . РИСОВАТЬ НА <2> НА <СЕВЕРО-ВОСТОК>
 - . РИСОВАТЬ НА <3> НА <ВОСТОК>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
- КОНЕЦ ПРЕДПИСАНИЯ
9. ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО БУКВЫ
- . ДВИГАТЬСЯ НА <1> НА <ЮГО-ВОСТОК>
- КОНЕЦ ПРЕДПИСАНИЯ
10. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ
- . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>
 - . ДВИГАТЬСЯ НА <2> НА <ЮГО-ЗАПАД>
 - . РИСОВАТЬ НА <3> НА <ВОСТОК>
 - . РИСОВАТЬ НА <4> НА <ЮГ>
 - . РИСОВАТЬ НА <4> НА <ЗАПАД>
 - . ДВИГАТЬСЯ НА <3> НА <ВОСТОК>
- КОНЕЦ ПРЕДПИСАНИЯ
11. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ
- . ДВИГАТЬСЯ НА <5> НА <СЕВЕР>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
 - . ДВИГАТЬСЯ НА <1> НА <СЕВЕР>
 - . РИСОВАТЬ НА <4> НА <СЕВЕРО-ВОСТОК>
 - . ДВИГАТЬСЯ НА <1> НА <СЕВЕР>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
- КОНЕЦ ПРЕДПИСАНИЯ
12. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
- . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>
 - . РИСОВАТЬ НА <4> НА <ЮГ>

- . РИСОВАТЬ НА <4> НА <ЗАПАД>
КОНЕЦ ПРЕДПИСАНИЯ
 - 13. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПЕРВУЮ БУКВУ
 - . ДВИГАТЬСЯ НА <5> НА <СЕВЕР>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
 - . РИСОВАТЬ НА <5> НА <ВОСТОК>
 - . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . ДВИГАТЬСЯ НА <2> НА <ЮГО-ЗАПАД>
 - . РИСОВАТЬ НА <4> НА <ЮГ>
 - . ДВИГАТЬСЯ НА <2> НА <ЮГО-ВОСТОК>КОНЕЦ ПРЕДПИСАНИЯ
 - 14. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ВТОРУЮ БУКВУ
 - . РИСОВАТЬ НА <5> НА <СЕВЕР>
 - . РИСОВАТЬ НА <2> НА <СЕВЕРО-ВОСТОК>
 - . РИСОВАТЬ НА <3> НА <ВОСТОК>
 - . РИСОВАТЬ НА <6> НА <ЮГ>
 - . ДВИГАТЬСЯ НА <2> НА <СЕВЕР>
 - . РИСОВАТЬ НА <4> НА <ЗАПАД>
 - . ДВИГАТЬСЯ НА <4> НА <ЮГО-ВОСТОК>КОНЕЦ ПРЕДПИСАНИЯ
 - 15. ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ТРЕТЬЮ БУКВУ
 - . РИСОВАТЬ НА <6> НА <СЕВЕР>
 - . РИСОВАТЬ НА <4> НА <ВОСТОК>
 - . РИСОВАТЬ НА <6> НА <ЮГ>КОНЕЦ ПРЕДПИСАНИЯ

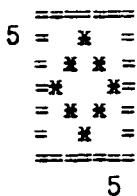
Образец выполнения. Вариант 15

Задание по рисованию простого орнамента

Напишите программу для рисования орнамента из элементов, каждый из которых помещается в клетке 5×5 . Размеры

поля для каждого варианта указаны. Элементы для всех вариантов одинаковы.

Рисунок одного элемента:



Размеры поля:

- | | | |
|-------------------|--------------------|--------------------|
| 1. 15×10 | 6. 30×20 | 11. 35×15 |
| 2. 45×15 | 7. 25×10 | 12. 40×20 |
| 3. 25×20 | 8. 30×15 | 13. 35×10 |
| 4. 20×10 | 9. 35×20 | 14. 40×20 |
| 5. 25×15 | 10. 30×10 | 15. 20×15 |

Образец выполнения. Вариант 15.

Алгоритм: разобъем поле размером 20×15 на $20:5=4$ вертикальные полосы. В каждой полосе будет $15:5=3$ клетки.

ПРОГРАММА РИСОВАНИЯ ОРНАМЕНТА

- . ИЗОБРАЗИТЕЛЬ ПОЛОС. НАЧАТЬ РИСОВАНИЕ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ИЗОБРАЗИТЬ ПОЛОСУ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ВСТАТЬ В НАЧАЛО ПОЛОСЫ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ИЗОБРАЗИТЬ ПОЛОСУ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ВСТАТЬ В НАЧАЛО ПОЛОСЫ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ИЗОБРАЗИТЬ ПОЛОСУ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ВСТАТЬ В НАЧАЛО ПОЛОСЫ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. ИЗОБРАЗИТЬ ПОЛОСУ
- . ИЗОБРАЗИТЕЛЬ ПОЛОС. КОНЧИТЬ РИСОВАНИЕ

КОНЕЦ ПРОГРАММЫ

" =====

ИСПОЛНИТЕЛЬ ИЗОБРАЗИТЕЛЬ ПОЛОС

- . СП:
 - . . 1. НАЧАТЬ РИСОВАНИЕ
 - . . 2. ИЗОБРАЗИТЬ ПОЛОСУ
 - . . 3. ВСТАТЬ В НАЧАЛО ПОЛОСЫ
 - . . 4. КОНЧИТЬ РИСОВАНИЕ
- . ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ: ИЗОБРАЗИТЕЛЬ КЛЕТОК,
- . ЧЕРТЕЖНИК
- КОНЕЦ ОПИСАНИЙ

" -----

. ПРЕДПИСАНИЕ НАЧАТЬ РИСОВАНИЕ

. . ДАНО: -----

. . ПОЛУЧИТЬ: ЧЕРТЕЖНИК ВКЛЮЧЕН;

ПЕРО В НАЧАЛЕ ПЕРВОЙ ПОЛОСЫ
ЧЕРТЕЖНИК. НАЧАТЬ РАБОТУ С ПОЛЕМ <45,15>
КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ ПОЛОСУ
ДАНО: ПЕРО В НАЧАЛЕ ПОЛОСЫ
ПОЛУЧИТЬ: ПОЛОСА НАРИСОВАНА, ПЕРО В КОНЦЕ ПОЛОСЫ
ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
ИЗОБРАЗИТЕЛЬ КЛЕТОК. ВСТАТЬ В НАЧАЛО КЛЕТКИ
ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
ИЗОБРАЗИТЕЛЬ КЛЕТОК. ВСТАТЬ В НАЧАЛО КЛЕТКИ
ИЗОБРАЗИТЕЛЬ КЛЕТОК. ИЗОБРАЗИТЬ КЛЕТКУ
КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО ПОЛОСЫ
ДАНО: ПЕРО В КОНЦЕ НАРИСОВАННОЙ ПОЛОСЫ
ПОЛУЧИТЬ: ПЕРО В НАЧАЛЕ СЛЕДУЮЩЕЙ ПОЛОСЫ
ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <5> НА <ВОСТОК>
ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <14> НА <ЮГ>
КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ КОНЧИТЬ РИСОВАНИЕ
ДАНО: ОРНАМЕНТ НАРИСОВАН
ПОЛУЧИТЬ: ЧЕРТЕЖНИК ВЫКЛЮЧЕН
ЧЕРТЕЖНИК. КОНЧИТЬ РАБОТУ
КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ

ИСПОЛНИТЕЛЬ ИЗОБРАЗИТЕЛЬ КЛЕТОК

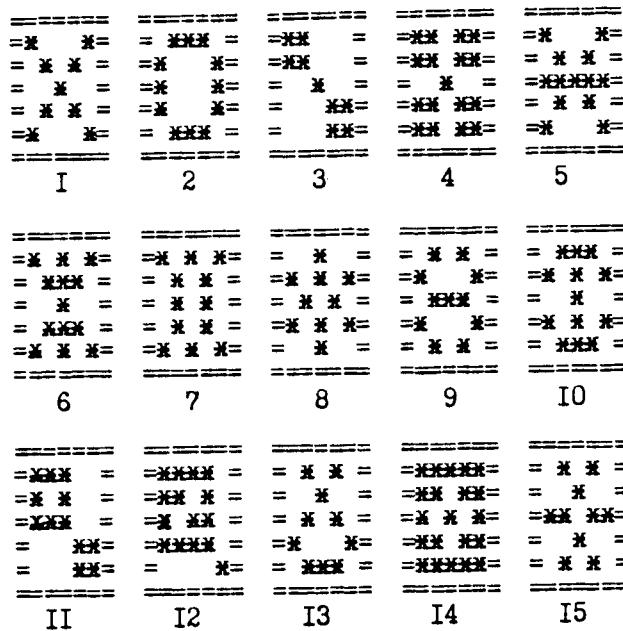
СП:
1. ИЗОБРАЗИТЬ КЛЕТКУ
2. ВСТАТЬ В НАЧАЛО КЛЕТКИ
ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ: ЧЕРТЕЖНИК
КОНЕЦ ОПИСАНИЙ

ПРЕДПИСАНИЕ ИЗОБРАЗИТЬ КЛЕТКУ
ДАНО: ПЕРО В НАЧАЛЕ КЛЕТКИ
ПОЛУЧИТЬ: КЛЕТКА НАРИСОВАНА, ПЕРО В КОНЦЕ КЛЕТКИ
ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <2> НА <СЕВЕР>
ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <СЕВЕРО-ВОСТОК>
ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ЮГО-ВОСТОК>
ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <ЮГО-ЗАПАД>
ЧЕРТЕЖНИК. РИСОВАТЬ НА <3> НА <СЕВЕРО-ЗАПАД>
ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <2> НА <СЕВЕР>
КОНЕЦ ПРЕДПИСАНИЯ

ПРЕДПИСАНИЕ ВСТАТЬ В НАЧАЛО КЛЕТКИ
ДАНО: ПЕРО В КОНЦЕ НАРИСОВАННОЙ КЛЕТКИ
ПОЛУЧИТЬ: ПЕРО В НАЧАЛЕ СЛЕДУЮЩЕЙ КЛЕТКИ
ЧЕРТЕЖНИК. ДВИГАТЬСЯ НА <1> НА <СЕВЕР>
КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ

Задание по рисованию сложного орнамента

Напишите программу, в результате выполнения которой на поле 30×15 будет нарисован орнамент, каждый элемент которого помещается в клетке 5×5 . Ниже приведены рисунки элементов для каждого варианта.



Дополнительное задание

Напишите программу для рисования того же орнамента на поле 33×18 . Обратите внимание, что при этом получаются «полные» и «неполные» клетки.

ТЕМА 3. УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Постановка вопроса

В задачах, которые мы решали до сих пор, последовательность необходимых действий была ясна заранее, до выполнения программы. При выполнении таких программ предписания выполнялись последовательно, одно за другим. Такие программы называются **линейными**.

Понятно, что круг существующих задач гораздо шире. Довольно часто необходимая последовательность действий выясняется только во время выполнения программы, т.е. зависит от условий, заранее не известных. Кроме того, необходимость выполнения предписания последовательно, друг за другом, приносит следующее неудобство: если одно и то же действие нужно выполнить несколько раз подряд, мы должны много раз написать соответствующее предписание. Нам были бы весьма полезны средства, позволяющие управлять последовательностью выполнения предписаний во время работы программы. Такие средства существуют и называются **управляющими конструкциями**.

Повторение заданное число раз

Простейшим видом управляющей конструкции является цикл с указанным числом повторений (**ЦИКЛ N РАЗ ВЫПОЛНИТЬ**). Эта конструкция обеспечивает возможность многократного повторения некоторой части программы и служит для сокращения записи программы. При помощи такой конструкции предписания, записанные между словами **ВЫПОЛНИТЬ** и **КОНЕЦ ЦИКЛА**, повторяются указанное число раз, после чего выполняется предписание, записанное после слов **КОНЕЦ ЦИКЛА**. Так, одинаково выполняются следующие фрагменты программ:

ПРИМЕР 1

СДЕЛАТЬ ШАГ
СДЕЛАТЬ ШАГ

ЦИКЛ 5 РАЗ ВЫПОЛНИТЬ
. СДЕЛАТЬ ШАГ

СДЕЛАТЬ ШАГ СДЕЛАТЬ ШАГ СДЕЛАТЬ ШАГ ПОВЕРНУТЬСЯ НАЛЕВО	КОНЕЦ ЦИКЛА ПОВЕРНУТЬСЯ НАЛЕВО .
.	.

ПРИМЕР 2

НАРИСОВАТЬ ПОЛОСУ ВСТАТЬ В НАЧАЛО ПОЛОСЫ НАРИСОВАТЬ ПОЛОСУ ВСТАТЬ В НАЧАЛО ПОЛОСЫ НАРИСОВАТЬ ПОЛОСУ ВСТАТЬ В НАЧАЛО ПОЛОСЫ НАРИСОВАТЬ ПОЛОСУ ВСТАТЬ В НАЧАЛО ПОЛОСЫ НАРИСОВАТЬ ПОЛОСУ	ЦИКЛ 4 РАЗА ВЫПОЛНИТЬ . НАРИСОВАТЬ ПОЛОСУ . ВСТАТЬ В НАЧАЛО ПОЛОСЫ КОНЕЦ ЦИКЛА НАРИСОВАТЬ ПОЛОСУ .
.	.

Повторение по условию

Часто возникает необходимость выполнить несколько раз одинаковые действия, причем количество повторений неизвестно заранее. Вместо этого в таких случаях задаются условия, при которых выполнение повторения необходимо закончить.

Рассмотрим такую задачу: ПУТНИК попал в лабиринт (схема лабиринта ему неизвестна). Дело происходит ночью или в тумане, и он не видит форму и размеры далеких препятствий. Дадим ему карманный фонарик, чтобы с его помощью он мог ответить на вопрос: свободен ли прямо перед ним маленький участок лабиринта по размерам равный одному шагу. Назовем такой участок клеткой (чтобы нам легче было следить за его передвижением по плану). Будем считать, что ПУТНИК умеет отвечать на вопросы:

КЛЕТКА СВОБОДНА?
КЛЕТКА ЗАНЯТА?

ПУТНИК отвечает на первый вопрос «да» (и, соответственно, на второй вопрос «нет»), когда пространство перед ним свободно. Если же перед ним стена или препятствие, он отвечает «нет» на первый вопрос (и, следовательно, «да» на второй). Как именно происходит эта проверка нас не интересует; нам важно лишь, что ни положение, ни ориентация при этом не изменяются.

Возможность такой проверки должна быть указана в системе предписаний исполнителя. Там она записывается так:

КЛЕТКА СВОБОДНА: ДА/НЕТ
КЛЕТКА ЗАНЯТА: ДА/НЕТ

Подобные предписания, которые не вызывают никаких действий исполнителя, а только отвечают на поставленный вопрос, называются *предписания, вырабатывающие значения*. В отличие от них все те предписания, которые мы рассматривали раньше, называются *предписания типа действие*.

Чтобы довести ПУТНИКА до ближайшего препятствия, находящегося впереди него, мы можем скомандовать:

```
ЦИКЛ ПОКА КЛЕТКА СВОБОДНА
  . ВЫПОЛНЯТЬ
  . СДЕЛАТЬ ШАГ
КОНЕЦ ЦИКЛА
```

Этот фрагмент программы будет выполняться так:

1. ПУТНИК проверит, свободна ли клетка перед ним.
2. Получит ответ «да».
3. Сделает шаг.

4. Проверит, свободна ли клетка перед ним, и т.д.

В некоторый момент он получит ответ «нет». Тогда выполнение действий, указанных внутри цикла, прекратится, и будет выполняться предписание, указанное после слов КОНЕЦ ЦИКЛА. При выполнении такого фрагмента в разных случаях ПУТНИК сделает разное число шагов:

=	=
=	=
В	В
XXXXXX=	XXXXXX=
XXXXXX=	XXXXXX=

На этих и следующих рисунках начальное положение ПУТНИКА отмечено буквой, а сама буква задает направление движения («В» — НА ВОСТОК). Препятствия обозначены символами «Х», стена — символами «==». В первом случае ПУТНИК сделает 9 шагов, во втором — 15.

Приведем еще один пример употребления конструкции ЦИКЛ ПОКА. Пусть ПУТНИК находится перед препятствием неизвестной формы, а нам нужно повернуть его так, чтобы он мог двигаться дальше. Тогда мы командуем ему

```
ЦИКЛ ПОКА КЛЕТКА ЗАНЯТА
  . ВЫПОЛНЯТЬ
  . ПОВЕРНУТЬС' НАЛЕВО
КОНЕЦ ЦИКЛА
```

Рассмотрим 3 случая расположения ПУТНИКА:

XXXXXX	XXXXXX	XXXXXX
XXXXXX	XXXXXX	XXXXXX
BXXXXX	XXXXBXXXX	XXXXB
XXXXX	XXXX XXXX	XXXX

Во всех трех случаях ПУТНИК стоит лицом на восток. Выполняя написанную конструкцию цикла, он
в первом случае повернется 2 раза,
во втором — 3 раза,
в третьем не повернется ни разу.

Вопрос. Сколько раз повернется ПУТНИК при выполнении такой конструкции в следующей ситуации:

XXXXXX
XXXXXX
XXXXBXXXX
XXXXXX
XXXXXX

Мы видим, что рассмотренные конструкции цикла, несмотря на важные отличия, обладают общим свойством: они вызывают многократное повторение некоторого фрагмента программы.

Выбор из двух вариантов

Довольно часто при выполнении программы возникает необходимость выполнить либо один, либо другие действия в зависимости от некоторого условия. Для возможности проверки такого условия в системе предписаний исполнителя должно содержаться соответствующее предписание, вырабатывающее значение.

Допустим, мы хотим, чтобы ПУТНИК двигался или поворачивался в зависимости от того, свободна ли клетка перед ним. Такой приказ ПУТНИКу можно было бы записать так:

ЕСЛИ КЛЕТКА СВОБОДНА,
ТО СДЕЛАТЬ ШАГ
ИНАЧЕ ПОВЕРНУТЬСЯ НАПРАВО

При выполнении такого приказа ПУТНИК:

- 1 — проверит, свободна ли клетка;
- 2 — получив ответ «да», сделает шаг;
- 3 — получив ответ «нет», повернется налево.

Вернемся к уже знакомой нам картинке (ПУТНИК стоит лицом на восток):

XXXXXX
XXXXXX
XXXXX
XXXXX

XXXXXX
XXXXXX
XXXXBXXXX
XXXX

XXXXXX
XXXXXX
XXXXB
XXXX

Выполняя написанную здесь конструкцию ЕСЛИ-ТО-ИНАЧЕ, ПУТНИК

- в первом случае повернется 1 раз;
во втором случае повернется 1 раз;
в третьем сделает 1 шаг.

Усложним задачу. Пусть при первом ответе «да» ПУТНИК движется, а при ответе «нет» — поворачивается на 270 градусов. Это можно было бы записать так:

```
ЕСЛИ КЛЕТКА СВОБОДНА
    ТО СДЕЛАТЬ ШАГ
ИНАЧЕ ПОВЕРНУТЬСЯ НАЛЕВО
    ПОВЕРНУТЬСЯ КРУГОМ
```

Пусть эти 4 строки входят в некоторую большую программу, например:

```
СДЕЛАТЬ ШАГ
ЕСЛИ КЛЕТКА СВОБОДНА
    ТО СДЕЛАТЬ ШАГ
ИНАЧЕ ПОВЕРНУТЬСЯ НАЛЕВО
    ПОВЕРНУТЬСЯ КРУГОМ
СДЕЛАТЬ 3 ШАГА
ПОВЕРНУТЬСЯ НАПРАВО
    . . .
```

Посмотрим, как будет ПУТНИК выполнять этот фрагмент программы. Если клетка занята, будет выполняться следующая последовательность предписаний:

```
СДЕЛАТЬ ШАГ
ПОВЕРНУТЬСЯ НАЛЕВО
ПОВЕРНУТЬСЯ КРУГОМ
СДЕЛАТЬ 3 ШАГА
ПОВЕРНУТЬСЯ НАПРАВО
```

Постараемся выписать аналогичную последовательность для случая, когда выполняется условие КЛЕТКА СВОБОДНА. Получим:

```
СДЕЛАТЬ ШАГ
СДЕЛАТЬ ШАГ
    . . . ???
```

Что должен делать ПУТНИК дальше, выполняя такую программу? Понятно, что налево поворачиваться не должен — это предписание стоит после слова ИНАЧЕ, значит, выполняется только при противоположном условии. Должен ли он поворачиваться кругом? Должен ли делать 3 шага? Как определить, какая часть программы должна выполняться только при нарушении условия, указанного после ЕСЛИ, а какая в любом случае?

Нам нужны указания, что выбор закончился и дальнейшие действия выполняются всегда. По аналогии с комбинацией ЦИКЛ—КОНЕЦ ЦИКЛА введем в качестве такой «закрывающей скобки» слова КОНЕЦ ЕСЛИ, которые будем понимать

так: «конец управляющей конструкции ЕСЛИ—ТО—ИНАЧЕ». Тогда приведенный приказ перепишется так:

```
ЕСЛИ КЛЕТКА СВОБОДНА
. ТО СДЕЛАТЬ ШАГ
. ИНАЧЕ ПОВЕРНУТЬСЯ НАЛЕВО
. ПОВЕРНУТЬСЯ КРУГОМ
КОНЕЦ ЕСЛИ
```

Наличие «скобок» в управляющих конструкциях, а также другие правила записи — вертикальная разметка, сдвиг внутренних предписаний вправо — приводят к тому, что начало и конец конструкций в программе становятся сразу заметны, а это значительно облегчает чтение программы:

```
СДЕЛАТЬ ШАГ
ЕСЛИ КЛЕТКА СВОБОДНА
. ТО СДЕЛАТЬ ШАГ
. ИНАЧЕ ПОВЕРНУТЬСЯ НАЛЕВО
. ПОВЕРНУТЬСЯ КРУГОМ
КОНЕЦ ЕСЛИ
СДЕЛАТЬ 3 ШАГА
ПОВЕРНУТЬСЯ НАПРАВО
. . . .
```

Теперь нам нетрудно выписать последовательность предложений, выполняемых ПУТНИКом при условии КЛЕТКА СВОБОДНА:

```
СДЕЛАТЬ ШАГ
СДЕЛАТЬ ШАГ
СДЕЛАТЬ 3 ШАГА
ПОВЕРНУТЬСЯ НАПРАВО
```

Итак, при выполнении конструкции ЕСЛИ—ТО—ИНАЧЕ

- Происходит проверка условия, написанного после слова ЕСЛИ.

- Когда при проверке вырабатывается ответ «да», выполняются все предписания типа действие, написанные между словами ТО и ИНАЧЕ, затем — первое предписание, указанное после слов КОНЕЦ ЕСЛИ.

- Когда при проверке вырабатывается ответ «нет», выполняются все предписания типа действие, написанные между словами ИНАЧЕ и КОНЕЦ ЕСЛИ, затем — первое предписание, указанное после слов КОНЕЦ ЕСЛИ.

Замечание. Довольно частым вариантом выбора одной возможности из двух является такой случай, когда при нарушении условия не нужно делать ничего. Возможность такого выбора записывается так:

```
ЕСЛИ КЛЕТКА СВОБОДНА
. ТО СДЕЛАТЬ ШАГ
КОНЕЦ ЕСЛИ
```

При выполнении этой конструкции ПУТНИК в двух первых (изображенных на рисунке выше) случаях не сделает ничего, а в третьем случае сделает один шаг.

Выбор из нескольких вариантов

При выполнении программы может возникнуть необходимость более сложного выбора — не из одного, а из нескольких возможных вариантов. В таких случаях используется конструкция ВЫБОРА.

Приведем пример:

ВЫБОР

- . ПУТНИК СТОИТ ЛИЦОМ НА ВОСТОК => ПОВЕРНУТЬСЯ НАЛЕВО
- . ПУТНИК СТОИТ ЛИЦОМ НА СЕВЕР => СДЕЛАТЬ ШАГ
- . ПУТНИК СТОИТ ЛИЦОМ НА ЗАПАД => ПОВЕРНУТЬСЯ НАПРАВО
- . ПУТНИК СТОИТ ЛИЦОМ НА ЮГ => ПОВЕРНУТЬСЯ КРУГОМ

КОНЕЦ ВЫБОРА

Здесь в каждой строке до стрелки стоит предписание, вырабатывающее значение, а после стрелки — предписание типа действие. При выполнении такой конструкции происходит проверка условий, начиная с первого. Как только встретится верное условие, будет выполнено предписание типа действие, указанное за стрелкой, и выполнение конструкции выбора прекращается — дальнейшая проверка условий не производится.

Так, в нашем примере ПУТНИК:

- 1 — проверит, стоит ли он лицом на восток;
 - 2 — получив ответ «да», повернется налево и будет выполнять первое предписание, указанное после слов КОНЕЦ ВЫБОРА;
 - 3 — получив ответ «нет», проверит, стоит ли он лицом на север;
 - 4 — получив ответ «да», сделает шаг и будет выполнять предписание, указанное после слов КОНЕЦ ВЫБОРА и т.д.
- Заметим, что если перед выполнением такой конструкции ПУТНИК стоит лицом на восток, то первое же условие выполняется и ПУТНИК поворачивается налево, т.е. оказывается лицом на север. Но делать шаг после этого ПУТНИК не будет, так как второе условие хотя и становится истинным, не проверяется.

Рассмотрим еще один пример с аналогичной ситуацией:

ВЫБОР

- . ЧИСЛО ЧЕТНОЕ => УВЕЛИЧИТЬ ЧИСЛО НА 1
- . ЧИСЛО НЕЧЕТНОЕ => УВЕЛИЧИТЬ ЧИСЛО НА 10

КОНЕЦ ВЫБОРА

Пусть у нас есть число 2. Тогда первое условие является истинным, и будет выполняться действие, стоящее после первой стрелки. На этом выполнение конструкции выбора закончится. Несмотря на то, что число станет нечетным, а значит, станет истинным второе условие, проверяться оно не будет.

В конструкции ВЫБОРА допустимо использование условия ИНАЧЕ. Оно считается истинным, если не выполнилось ни одно из условий, указанных в конструкции ранее. Например, конструкцию

ВЫБОР

- . ПУТНИК СТОИТ ЛИЦОМ НА ВОСТОК => ПОВЕРНУТЬСЯ НАЛЕВО
- . ПУТНИК СТОИТ ЛИЦОМ НА СЕВЕР => СДЕЛАТЬ ШАГ
- . ПУТНИК СТОИТ ЛИЦОМ НА ЗАПАД => ПОВЕРНУТЬСЯ НАПРАВО
- . ПУТНИК СТОИТ ЛИЦОМ НА ЮГ => ПОВЕРНУТЬСЯ НАПРАВО

КОНЕЦ ВЫБОРА

можно записать следующим образом:

ВЫБОР

- . ПУТНИК СТОИТ ЛИЦОМ НА ВОСТОК => ПОВЕРНУТЬСЯ НАЛЕВО
- . ПУТНИК СТОИТ ЛИЦОМ НА СЕВЕР => СДЕЛАТЬ ШАГ
- . ИНАЧЕ => ПОВЕРНУТЬСЯ НАПРАВО

КОНЕЦ ВЫБОРА

При выполнении такой конструкции при всех условиях, кроме перечисленных (в нашем примере — когда ПУТНИК стоит лицом на запад или на юг), будет выполняться предписание типа действие, указанное после слова ИНАЧЕ, а именно ПУТНИК будет поворачиваться направо.

Замечание. В общем случае после каждой стрелки может находиться не одно предписание типа действие, а несколько. В таком случае при выполнении соответствующего условия выполняется вся группа предписаний, указанных после стрелки, после чего выполнение конструкции ВЫБОРА заканчивается.

Сложные условия

В задачах часто возникает необходимость предпринять те или иные действия в зависимости от выполнения не одного, а нескольких условий сразу. Например, УЧЕНИКу нужно идти в школу, когда выполняются 2 условия:

- день рабочий (не воскресенье);
- температура выше -30° .

Фрагмент программы для поведения УЧЕНИКА записывается так:

ЕСЛИ ДЕНЬ РАБОЧИЙ И ТЕМПЕРАТУРА ВЫШЕ -30°
. ТО ИДТИ В ШКОЛУ
. ИНАЧЕ ОСТАВАТЬСЯ ДОМА
КОНЕЦ ЕСЛИ

Два условия соединены здесь союзом «И». Он не является частью управляющей конструкции ЕСЛИ—ТО—ИНАЧЕ и не является предписанием, вырабатывающим значение. Союз «И» соединяет два таких предписания, выполняя роль логической операции: показывает, что действие, указанное после ТО, нужно выполнять только при выполнении обоих условий.

Рассмотрим 4 случая:

1. Воскресенье, -35°

2. Воскресенье, -15°
3. Вторник, -32°
4. Четверг, $+4^{\circ}$

В первом случае не выполняется ни одно из условий, и в школу идти не нужно.

Во втором случае не выполняется условие ДЕНЬ РАБОЧИЙ — в школу идти не нужно.

В третьем случае не выполняется условие ТЕМПЕРАТУРА ВЫШЕ -30° ; в школу опять-таки идти не нужно.

В четвертом случае выполнены оба условия, и вступает в силу предписание типа действие, указанное после слова ТО — УЧЕНИК должен идти в школу.

Наши рассуждения о том, когда выполняется сложное условие, состоящее из двух простых, соединенных союзом «И», можно оформить в виде такой таблички:

НЕТ И НЕТ	= НЕТ
НЕТ И ДА	= НЕТ
ДА И НЕТ	= НЕТ
ДА И ДА	= ДА

Иногда нужно выполнить некоторые действия при выполнении одного из двух условий. Например, нужно надеть куртку, когда либо на улице идет дождь, либо холоднее 12° . Приказ соответствующему исполнителю на выполнение этих действий мы запишем так:

ЕСЛИ ИДЕТ ДОЖДЬ ИЛИ ХОЛОДНЕЕ 12°
. ТО НАДЕТЬ КУРТКУ
КОНЕЦ ЕСЛИ

Снова рассмотрим 4 случая:

1. На улице сухо, температура 20° . Оба условия нарушены, и куртку надевать не надо.
2. На улице дождь, температура 15° . Нужно надеть куртку для защиты от дождя, хотя и тепло.
3. На улице сухо, температура 10° . Холодно — придется надеть куртку.
4. На улице дождь, температура 8° . Надеваем куртку для защиты как от дождя, так и от холода.

Наши рассуждения оформим табличкой:

НЕТ ИЛИ НЕТ	= НЕТ
ДА ИЛИ НЕТ	= ДА
НЕТ ИЛИ ДА	= ДА
ДА ИЛИ ДА	= ДА

Иногда нам придется иметь дело со сложным условием, состоящим более чем из двух простых. Мы будем вычислять его значение, последовательно применяя правила из табличек к каждым двум условиям. Если все условия соединены одинак-

ковыми логическими операциями (все операции «И» или все операции «ИЛИ»), у нас не возникнет сложностей. А как быть, если встретятся разные логические операции?

Заметим, что операция «И» очень похожа на операцию умножения, а «ИЛИ» — на операцию сложения, если считать, что значению «ДА» соответствует какое-либо положительное число, а значению «НЕТ» — 0.

НЕТ И НЕТ	= НЕТ	$O \times O = O$
НЕТ И ДА	= НЕТ	$O \times X = O$
ДА И НЕТ	= НЕТ	$X \times O = O$
ДА И ДА	= ДА	$X \times X = X^2$
НЕТ ИЛИ НЕТ	= НЕТ	$O + O = O$
ДА ИЛИ НЕТ	= ДА	$X + O = X$
НЕТ ИЛИ ДА	= ДА	$O + X = X$
ДА ИЛИ ДА	= ДА	$X + X = 2X$

По аналогии с арифметическими операциями введем понятие старшинства логических операций. Будем считать, что операция «И» (соответствующая умножению) старше, чем операция «ИЛИ» (соответствующая сложению). Тогда сложные условия, состоящие из трех простых, следует понимать так:

ХОЛОДНО И МОКРО ИЛИ СОЛНЕЧНО = (ХОЛОДНО И МОКРО) ИЛИ СОЛНЕЧНО
СОЛНЕЧНО ИЛИ ХОЛОДНО И МОКРО = СОЛНЕЧНО ИЛИ (ХОЛОДНО И МОКРО)

Построение отрицаний к сложным условиям

Вспомним нашу программу для поведения УЧЕНИКА:

ЕСЛИ ДЕНЬ РАБОЧИЙ И ТЕМПЕРАТУРА ВЫШЕ -30°
. ТО ИДТИ В ШКОЛУ

. ИНАЧЕ ОСТАВАТЬСЯ ДОМА
КОНЕЦ ЕСЛИ

После слова ЕСЛИ здесь указано то условие, при котором необходимо идти в школу. Мы могли бы поступить иначе — указать условие, когда нужно оставаться дома, и тогда наш приказ выглядел бы так:

ЕСЛИ . . .
. ТО ОСТАВАТЬСЯ ДОМА
. ИНАЧЕ ИДТИ В ШКОЛУ
КОНЕЦ ЕСЛИ

Понятно, что после слова ЕСЛИ здесь должно быть указано условие, противоположное условию ДЕНЬ РАБОЧИЙ И ТЕМПЕРАТУРА ВЫШЕ -30° . Противоположное условие условию ДЕНЬ РАБОЧИЙ будет ДЕНЬ ВЫХОДНОЙ, к ТЕМ-

ПЕРАТУРА ВЫШЕ -30° — ТЕМПЕРАТУРА НИЖЕ -30°
(строго говоря, ниже или равна, но для краткости мы будем писать так).

Какой логической операцией они должны быть соединены?
Когда ученик должен оставаться дома? Либо в выходной день, либо в сильный мороз. Итак, следующие две записи являются эквивалентными:

ЕСЛИ ДЕНЬ РАБОЧИЙ И
. ТЕМПЕРАТУРА ВЫШЕ -30°
. ТО ИДТИ В ШКОЛУ
. ИНАЧЕ ОСТАВАТЬСЯ ДОМА
КОНЕЦ ЕСЛИ

ЕСЛИ ДЕНЬ ВЫХОДНОЙ ИЛИ
. ТЕМПЕРАТУРА НИЖЕ -30°
. ТО ОСТАВАТЬСЯ ДОМА
. ИНАЧЕ ИДТИ В ШКОЛУ
КОНЕЦ ЕСЛИ

а такие два условия являются противоположными:

ДЕНЬ РАБОЧИЙ И
ТЕМПЕРАТУРА ВЫШЕ -30°

ДЕНЬ ВЫХОДНОЙ ИЛИ
ТЕМПЕРАТУРА НИЖЕ -30°

Запишем иначе приказ о том, когда следует надевать куртку:

ЕСЛИ ИДЕТ ДОЖДЬ ИЛИ ХОЛОДНЕЕ 12°
ТО НАДЕТЬ КУРТКУ
КОНЕЦ ЕСЛИ

Получим:

ЕСЛИ НЕТ ДОЖДЯ И ТЕПЛЕЕ 12°
. ТО НИЧЕГО НЕ ДЕЛАТЬ
. ИНАЧЕ НАДЕТЬ КУРТКУ
КОНЕЦ ЕСЛИ

Можно проверить, что во всех 4 случаях различных погодных условий эти конструкции будут выполняться одинаково. Итак, условия

ИДЕТ ДОЖДЬ ИЛИ ХОЛОДНЕЕ 12°

и
НЕТ ДОЖДЯ И ТЕПЛЕЕ 12°
являются противоположными.

Запишем эти сведения в табличку:

НЕ (A И B) = (НЕ A) ИЛИ (НЕ B)
НЕ (A ИЛИ B) = (НЕ A) и (НЕ B)

Вложенные конструкции

Вспомним общий вид изученных нами управляющих конструкций:

ЕСЛИ <УСЛОВИЕ>
. ТО <ПРЕДПИСАНИЕ 1 ТИПА ДЕЙСТВИЕ>
. <ПРЕДПИСАНИЕ 2 ТИПА ДЕЙСТВИЕ>
.

- . ИНАЧЕ
- . <ПРЕДПИСАНИЕ К ТИПА ДЕЙСТВИЕ>
- КОНЕЦ ЕСЛИ
- ВЫБОР
- . <УСЛОВИЕ 1> => <ПРЕДПИСАНИЕ 1 ТИПА ДЕЙСТВИЕ>
- . <УСЛОВИЕ 2> => <ПРЕДПИСАНИЕ 2 ТИПА ДЕЙСТВИЕ>
- .
- . <УСЛОВИЕ K> => <ПРЕДПИСАНИЕ K ТИПА ДЕЙСТВИЕ>
- . ИНАЧЕ=> <ПРЕДПИСАНИЕ K+1 ТИПА ДЕЙСТВИЕ>
- КОНЕЦ ВЫБОРА
- ЦИКЛ N РАЗ ВЫПОЛНИТЬ
- . <ПРЕДПИСАНИЕ 1 ТИПА ДЕЙСТВИЕ>
- . <ПРЕДПИСАНИЕ 2 ТИПА ДЕЙСТВИЕ>
- .
- . <ПРЕДПИСАНИЕ K ТИПА ДЕЙСТВИЕ>
- КОНЕЦ ЦИКЛА
- ЦИКЛ ПОКА <УСЛОВИЕ>
- . ВЫПОЛНЯТЬ
- . <ПРЕДПИСАНИЕ 1 ТИПА ДЕЙСТВИЕ>
- . <ПРЕДПИСАНИЕ 2 ТИПА ДЕЙСТВИЕ>
- .
- . <ПРЕДПИСАНИЕ K ТИПА ДЕЙСТВИЕ>
- КОНЕЦ ЦИКЛА

Мы уже знаем, что там, где стоит слово УСЛОВИЕ, может находиться и сложное условие, состоящее из двух или более простых. Сделаем еще одно обобщение.

Везде, где стоят слова ПРЕДПИСАНИЕ ТИПА ДЕЙСТВИЕ, может находиться любая управляющая конструкция.

Разберем на нескольких примерах, как происходит выполнение программ, содержащих такие *вложенные* конструкции. Для наглядности рядом с фрагментом программы будем приводить его схему, где цикл будем обозначать знаками



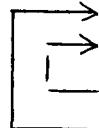
а ЕСЛИ-ТО-ИНАЧЕ — развилкой:



Пример 1.

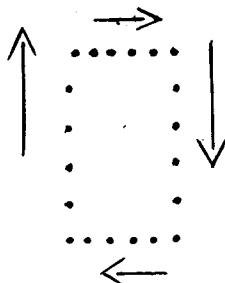
- ЦИКЛ 4 РАЗА ВЫПОЛНИТЬ
- . ПОВЕРНУТЬСЯ НАПРАВО
 - . ЦИКЛ 5 РАЗ ВЫПОЛНИТЬ
 - .. СДЕЛАТЬ ШАГ

Схема



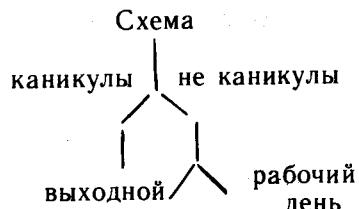
КОНЕЦ ЦИКЛА КОНЕЦ ЦИКЛА

В результате выполнения такой конструкции ПУТНИК сделает 20 шагов по 5 в каждом из четырех направлений и пройдет такой путь:



Пример 2.

```
ЕСЛИ КАНИКУЛЫ
. . ТО ПОЙТИ В ПОХОД
. . ИНАЧЕ
. . ЕСЛИ ДЕНЬ ВЫХОДНОЙ
. . . ТО ПОЙТИ В ЛЕС
. . . ИНАЧЕ ПОЙТИ В ШКОЛУ
. . КОНЕЦ ЕСЛИ
КОНЕЦ ЕСЛИ
```



Заметим, что здесь выполняются различные действия в трех различных случаях:

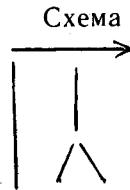
- каникулы;
- не каникулы, выходной;
- не каникулы, рабочий день.

Можно было бы записать эквивалентный фрагмент программы с помощью конструкции ВЫБОРА:

```
ВЫБОР
. КАНИКУЛЫ => ПОЙТИ В ПОХОД
. НЕ КАНИКУЛЫ И ВЫХОДНОЙ => ПОЙТИ В ЛЕС
. НЕ КАНИКУЛЫ И РАБОЧИЙ ДЕНЬ => ПОЙТИ В ШКОЛУ
КОНЕЦ ВЫБОРА
```

Пример 3.

```
ПОВЕРНУТЬСЯ НА ВОСТОК
УСТАНОВИТЬ В НОЛЬ СЧЕТЧИК ДЛИНЫ
ЦИКЛ ПОКА МОЖНО СДЕЛАТЬ ШАГ
. ВЫПОЛНЯТЬ
. СДЕЛАТЬ ШАГ
. ПОВЕРНУТЬСЯ НА ЮГ
. ЕСЛИ НЕЛЬЗЯ СДЕЛАТЬ ШАГ
. . ТО УВЕЛИЧИТЬ НА ЕДИНИЦУ СЧЕТЧИК ДЛИНЫ
. КОНЕЦ ЕСЛИ
. ПОВЕРНУТЬСЯ НА ВОСТОК
КОНЕЦ ЦИКЛА
```



При выполнении такого фрагмента программы ПУТНИК будет двигаться в восточном направлении до стены или препятствия и при этом подсчитывать суммарную длину всех препятствий, находящихся к югу от него (рядом с которыми он проходит).

На рисунках положение ПУТНИКА отмечено одним из символов «В» или «Ю». Символ является первой буквой стороны света, куда ПУТНИК стоит лицом.

B XX X= XX X= длина=0 №1.	B XX X= XX X= длина=0 №2.	Ю XX X= XX X= длина=0 №3.	Ю XX X= XX X= длина=1 №4.
B XX X= XX X= длина=1 №5.	B XX X= XX X= длина=1 №6.	Ю XX X= XX X= длина=1 №7.	Ю XX X= XX X= длина=2 №8.
B XX X= XX X= длина=2 №9.	B XX X= XX X= длина=2 №10.	Ю XX X= XX X= длина=2 №11.	B XX X= XX X= длина=2 №12.
B XX X= XX X= длина=2 №13.	Ю XX X= XX X= длина=2 №14.	Ю XX X= XX X= длина=2 №15.	B XX X= XX X= длина=2 №16.
Ю XX X= XX X= длина=2 №17.	Ю XX X= XX X= длина=3 №18.	Ю XX X= XX X= длина=3 №19.	

В заключение этого раздела приведем (без подробного разбора) следующий, более трудный пример.

Пример 4.

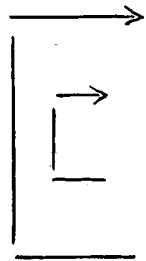
ПРОГРАММА ВЫХОДА ИЗ ЛАБИРИНТА ПО ПРАВИЛУ
ПРАВОЙ РУКИ
НАЧАТЬ РАБОТУ В КВАДРАНТЕ Х Х Х Х Х

```

. ЦИКЛ ПОКА НЕ ВЫШЕЛ
.. ВЫПОЛНЯТЬ
.. ЦИКЛ ПОКА НЕЛЬЗЯ СДЕЛАТЬ ШАГ
.. . ВЫПОЛНЯТЬ
.. . ПОВЕРНУТЬСЯ НАЛЕВО
.. . КОНЕЦ ЦИКЛА
.. СДЕЛАТЬ ШАГ
.. ПОВЕРНУТЬСЯ НАПРАВО
.. КОНЕЦ ЦИКЛА
.. КОНЧИТЬ РАБОТУ
КОНЕЦ ПРОГРАММЫ

```

Схема



Действуя по такой программе, ПУТНИК выйдет из любого (незнакомого ему) лабиринта «по правилу правой руки», т.е. таким образом, чтобы справа от него все время было препятствие. На каждом из приведенных рисунков показаны положение и ориентация ПУТНИКА на небольшом участке пути.

=10	=10	=3	=10
=	=	=	=
=XXX	=XXX	=XXX	=XXX
=XXX	=XXX	=XXX	=XXX
#1.	#2.	#3.	#4.
=	=	=	=
=	=	=	=
=10	=3	=10	=B
=XXX	=XXX	=XXX	=XXX
=XXX	=XXX	=XXX	=XXX
#5.	#6.	#7.	#8.
=	=	=	=
=	=	=	=
= B	= 10	= B	= BB
=XXX	=XXX	=XXX	=XXX
=XXX	=XXX	=XXX	=XXX
#9.	#10.	#11.	#12.
=	=	=	=
=	=	=	=
= B	= 10	=	=
=XXX	=XXX	=XXX	=XXX3
=XXX	=XXX	=XXX	=XXX
#13.	#14.	#15.	#16.

Задание. Напишите аналогичную программу для выхода из лабиринта по «правилу левой руки».

Задание № 1 по теме УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Напишите с помощью конструкции ВЫБОРа кусочно-линейную функцию, заданную графически (см. график на с. 62).

Образец выполнения. Вариант 15

ВЫБОР

- . X < -1 => Y = 1
- . X < 0 => Y = 2X + 3
- . X < 1 => Y = -X + 3
- . ИНАЧЕ => Y = 2

КОНЕЦ ВЫБОРА

Задание № 2 по теме УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

С помощью вложенной конструкции ЦИКЛ Н РАЗ ВЫПОЛНИТЬ напишите программу для вычисления

1. Суммы первых 10 четных чисел.
2. Суммы первых 9 нечетных чисел.
3. Суммы первых 8 чисел, кратных 3.
4. Суммы первых 7 чисел, дающих при делении на 3 остаток 1.
5. Суммы первых 6 чисел, дающих при делении на 3 остаток 2.
6. Суммы первых 5 чисел, кратных 4.
7. Суммы первых 4 чисел, дающих при делении на 4 остаток 1.
8. Суммы первых 3 чисел, кратных 5.
9. Суммы первых 4 чисел, дающих при делении на 5 остаток 1.
10. Суммы первых 6 чисел, дающих при делении на 5 остаток 2.
11. Суммы первых 5 чисел, кратных 6.
12. Суммы первых 7 чисел, дающих при делении на 6 остаток 1.
13. Суммы первых 8 чисел, кратных 7.
14. Суммы первых 9 чисел, дающих при делении на 7 остаток 1.
15. Суммы первых 10 чисел, дающих при делении на 7 остаток 2.

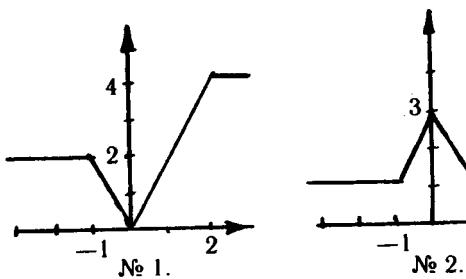
Исполнитель СУММАТОР

- СП:
1. НАЧАТЬ ВЫЧИСЛЕНИЕ
 2. СУММА. УСТАНОВИТЬ В НОЛЬ
 3. СЛАГАЕМОЕ. УСТАНОВИТЬ В НОЛЬ
 4. СЛАГАЕМОЕ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 5. СУММА. УВЕЛИЧИТЬ НА СЛАГАЕМОЕ
 6. СЛАГАЕМОЕ. ПОКАЗАТЬ РЕЗУЛЬТАТ
 7. СУММА. ПОКАЗАТЬ РЕЗУЛЬТАТ
 8. КОНЧИТЬ ВЫЧИСЛЕНИЕ

Образец выполнения. Вариант 15

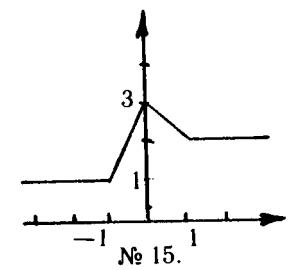
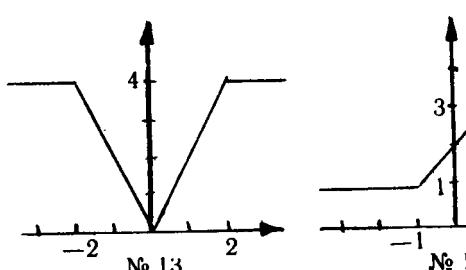
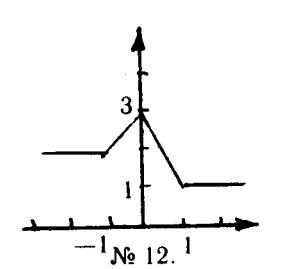
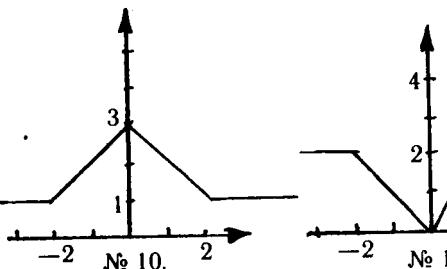
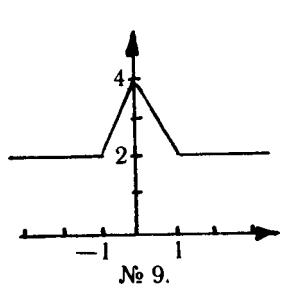
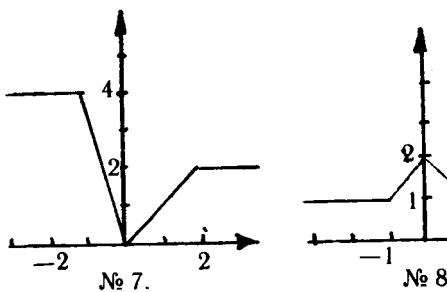
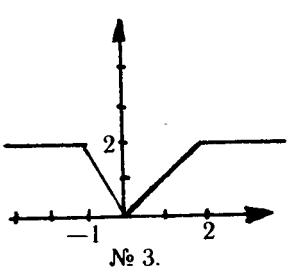
ПРОГРАММА ВЫЧИСЛЕНИЯ СУММЫ

- . НАЧАТЬ ВЫЧИСЛЕНИЕ
- . СУММА. УСТАНОВИТЬ В НОЛЬ



The figure shows two separate graphs labeled № 4 and № 5.

- Graph № 4:** A V-shaped function opening upwards. The vertex is at the origin (0, 0). The function is constant at y = 2 for x < -1 and x > 1, and decreases linearly from y = 2 to y = 0 at x = -1 and x = 1.
- Graph № 5:** A V-shaped function opening upwards. The vertex is at (-1, 1). The function is constant at y = 1 for x < -1 and x > 1, and decreases linearly from y = 1 to y = 0 at x = -1 and x = 1.



```

. СЛАГАЕМОЕ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. СЛАГАЕМОЕ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. СЛАГАЕМОЕ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. ЦИКЛ 10 РАЗ ВЫПОЛНИТЬ
.. ЦИКЛ 7 РАЗ ВЫПОЛНИТЬ
... СЛАГАЕМОЕ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
.. КОНЕЦ ЦИКЛА
.. СЛАГАЕМОЕ. ПОКАЗАТЬ РЕЗУЛЬТАТ
.. СУММА. УВЕЛИЧИТЬ НА СЛАГАЕМОЕ
. КОНЕЦ ЦИКЛА
. СУММА. ПОКАЗАТЬ РЕЗУЛЬТАТ
. КОНЧИТЬ ВЫЧИСЛЕНИЕ
КОНЕЦ ПРОГРАММЫ

```

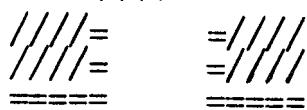
Вопрос. Какие числа будут напечатаны при выполнении предписаний ПОКАЗАТЬ РЕЗУЛЬТАТ?

Исполнитель ПЕШЕХОД

- СП:
1. НАЧАТЬ ПЕРЕХОД УЛИЦЫ С КОДОМ XXX
 2. СДЕЛАТЬ ШАГ
 3. СТОЯТЬ НА МЕСТЕ
 4. ПОВЕРНУТЬСЯ НАЛЕВО
 5. ПОВЕРНУТЬСЯ НАПРАВО
 6. ПЕРЕШЕЛ ДОРОГУ : ДА/НЕТ
 7. ДОШЕЛ ДО СЕРЕДИНЫ : ДА/НЕТ
 8. СВЕТОФОР РАБОТАЕТ : ДА/НЕТ
 9. ЦВЕТ КРАСНЫЙ : ДА/НЕТ
 10. ЦВЕТ ЖЕЛТЫЙ : ДА/НЕТ
 11. ЦВЕТ ЗЕЛЕНЫЙ : ДА/НЕТ
 12. МАШИНА СЛЕВА : ДА/НЕТ
 13. МАШИНА СПРАВА : ДА/НЕТ
 14. СЧЕТЧИК ШАГОВ. УСТАНОВИТЬ В НОЛЬ
 15. СЧЕТЧИК ШАГОВ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 16. СЧЕТЧИК ШАГОВ. ПОКАЗАТЬ ЗНАЧЕНИЕ
 17. КОНЧИТЬ ПЕРЕХОД

Необходимые пояснения к набору действий:

ПЕШЕХОД начинает переход перекрестка с одного из углов, стоя лицом на север (положение ПЕШЕХОДА обозначено символом «С»)



Ширина улиц ПЕШЕХОДУ неизвестна. Чтобы пересечь перекресток, ему нужно пересечь одну улицу, повернуться и пересечь другую. При переходе светофор может работать, а может оказаться неисправным. При работающем светофоре ПЕШЕХОД должен подчиняться сигналам светофора: дождаться зеленого света, дойти до середины улицы, снова дождаться зеленого света, перейти улицу до конца. В противном случае его ожидают разные неприятности (он может быть сбит машиной или оштрафован). При неработающем светофоре ПЕШЕХОД должен следить за машинами: дождаться такой ситуации, когда машины слева нет, дойти до середины улицы, дождаться, когда не будет машины справа, и перейти улицу до конца. В начале перехода ПЕШЕХОД должен убедиться в том, работает ли светофор (если только эта информация не задана в условии задачи).

Вопросы.

1. Как будет переходить улицу фаталист (полагающийся только на судьбу и не заботящийся о своей безопасности?)
2. Как будет вести себя самоубийца, решивший броситься под машину?

Напишите для этих двух случаев программы с использованием конструкции ЦИКЛ ПОКА.

Задание № 3 по теме УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Напишите программу для исполнителя ПЕШЕХОД, при помощи которой требуется:

- 1 — перейти улицу без подсчета ширины (неизвестно, работает ли светофор).
- 2 — перейти улицу при работающем светофоре с подсчетом ширины.
- 3 — перейти улицу при неработающем светофоре с подсчетом ширины.
- 4 — перейти перекресток без подсчета ширины улиц (неизвестно, работает ли светофор).
- 5 — перейти перекресток с подсчетом ширины 1-й улицы.
Светофор работает.
- 6 — перейти перекресток с подсчетом ширины 1-й улицы.
Светофор не работает.
- 7 — перейти перекресток с подсчетом ширины 2-й улицы.
Светофор работает.
- 8 — перейти перекресток с подсчетом ширины 2-й улицы.
Светофор не работает.

Образец выполнения. Вариант 8.

ПРОГРАММА ПЕРЕХОДА ПЕРЕКРЕСТКА
. НАЧАТЬ ПЕРЕХОД УЛИЦЫ С КОДОМ XXX
. ЦИКЛ ПОКА МАШИНА СЛЕВА

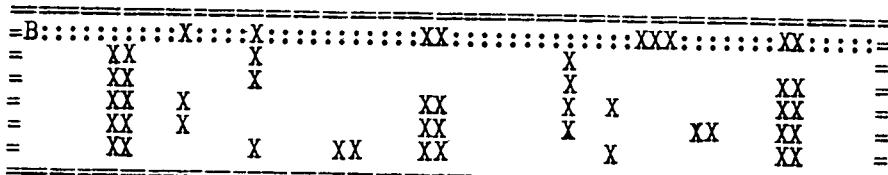
```
. . ВЫПОЛНЯТЬ
. . СТОЯТЬ НА МЕСТЕ
. КОНЕЦ ЦИКЛА

. "
. ЦИКЛ ПОКА НЕ ДОШЕЛ ДО СЕРЕДИНЫ
. . ВЫПОЛНЯТЬ
. . СДЕЛАТЬ ШАГ
. КОНЕЦ ЦИКЛА
. .. дошли до середины первой улицы
. . ЦИКЛ ПОКА МАШИНА СПРАВА
. . ВЫПОЛНЯТЬ
. . СТОЯТЬ НА МЕСТЕ
. КОНЕЦ ЦИКЛА
. "
. ЦИКЛ ПОКА НЕ ПЕРЕШЕЛ ДОРОГОУ
. . ВЫПОЛНЯТЬ
. . СДЕЛАТЬ ШАГ
. КОНЕЦ ЦИКЛА
. .. перешли первую улицу
. ПОВЕРНУТЬСЯ НАПРАВО
. СЧЕТЧИК ШАГОВ. УСТАНОВИТЬ В НОЛЬ

. "
. ЦИКЛ ПОКА МАШИНА СЛЕВА
. . ВЫПОЛНЯТЬ
. . СТОЯТЬ НА МЕСТЕ
. КОНЕЦ ЦИКЛА
. "
. ЦИКЛ ПОКА НЕ ДОШЕЛ ДО СЕРЕДИНЫ
. . ВЫПОЛНЯТЬ
. . СДЕЛАТЬ ШАГ
. . СЧЕТЧИК ШАГОВ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. КОНЕЦ ЦИКЛА
. .. дошли до середины второй улицы
. ЦИКЛ ПОКА МАШИНА СПРАВА
. . ВЫПОЛНЯТЬ
. . СТОЯТЬ НА МЕСТЕ
. КОНЕЦ ЦИКЛА
. "
. ЦИКЛ ПОКА НЕ ПЕРЕШЕЛ ДОРОГОУ
. . ВЫПОЛНЯТЬ
. . СДЕЛАТЬ ШАГ
. . СЧЕТЧИК ШАГОВ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. КОНЕЦ ЦИКЛА
. "
. СЧЕТЧИК ШАГОВ. ПОКАЗАТЬ ЗНАЧЕНИЕ
. КОНЧИТЬ РАБОТУ
КОНЕЦ ПРОГРАММЫ
```

Исполнитель РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ

Следующая наша задача будет посвящена движению РОБОТА в неизвестном квадранте. Квадрант обнесен замкнутой стеной и содержит внутри вертикальные ряды прямоугольных препятствий; все препятствия одного ряда имеют одинаковую ширину. Пример:



Нам нужно будет провести РОБОТА по всему квадранту или по его части и получить некоторую информацию о квадранте (количество препятствий, их площадь и т.п.). Задача эта сложная, и для ее решения нам понадобятся исполнители с богатыми возможностями. Сейчас мы выпишем систему предписаний базовых исполнителей и обсудим некоторые их действия. Заметим сразу, что при выполнении первых двух заданий мы будем пользоваться лишь небольшой частью возможностей базовых исполнителей.

Группу базовых исполнителей мы будем условно называть РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ, хотя, кроме них, в нашем распоряжении будут еще 5 пар исполнителей-счетчиков: ЧИСЛО — ОБЩЕЕ_ЧИСЛО, ШИРИНА — ОБЩАЯ_ШИРИНА и т.п.

Система предписаний базовых исполнителей:

1. РОБОТ. НАЧАТЬ РАБОТУ В КВАДРАНТЕ
 2. РОБОТ. ВПЕРЕДИ СВОБОДНО : ДА/НЕТ
 3. РОБОТ. СПРАВА СВОБОДНО : ДА/НЕТ
 4. РОБОТ. СЛЕВА СВОБОДНО : ДА/НЕТ
 5. РОБОТ. ВПЕРЕДИ ЗАНЯТО : ДА/НЕТ
 6. РОБОТ. СПРАВА ЗАНЯТО : ДА/НЕТ
 7. РОБОТ. СЛЕВА ЗАНЯТО : ДА/НЕТ
 8. РОБОТ. СДЕЛАТЬ ШАГ
 9. РОБОТ. ШАГАТЬ ДО УПОРА
 10. РОБОТ. ПОВЕРНУТЬСЯ НАПРАВО
 11. РОБОТ. ПОВЕРНУТЬСЯ НАЛЕВО
 12. РОБОТ. ПОВЕРНУТЬСЯ НА СЕВЕР
 13. РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
 14. РОБОТ. ПОВЕРНУТЬСЯ НА ЗАПАД
 15. РОБОТ. ПОВЕРНУТЬСЯ НА ВОСТОК
 16. РОБОТ. КОНЧИТЬ РАБОТУ
-
1. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. УСТАНОВИТЬ В НОЛЬ
 2. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ШИРИНА МЕНЬШЕ ВЫСОТЫ
ДА/НЕТ

3. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ШИРИНА РАВНА ВЫСОТЕ : ДА/НЕТ
 4. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ШИРИНА БОЛЬШЕ ВЫСОТЫ : ДА/НЕТ
 5. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО КРАЙ
 6. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО РЯД СВОБОДЕН
 7. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО НАЙДЕН УГОЛ
 8. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. КРАЙ : ДА/НЕТ
 9. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. НЕ КРАЙ : ДА/НЕТ
 10. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. РЯД СВОБОДЕН : ДА/НЕТ
 11. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. НАЙДЕН УГОЛ : ДА/НЕТ
 12. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. НЕ НАЙДЕН УГОЛ : ДА/НЕТ
 13. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО ФЛАГ ПОДНЯТ
 14. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО ФЛАГ ОПУЩЕН
 15. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ФЛАГ ПОДНЯТ : ДА/НЕТ
 16. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ФЛАГ ОПУЩЕН : ДА/НЕТ
1. ЧИСЛО. УСТАНОВИТЬ В НОЛЬ
 2. ЧИСЛО. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ЧИСЛО. ПОКАЗАТЬ ЗНАЧЕНИЕ
1. ОБЩ_ЧИСЛО. УСТАНОВИТЬ В НОЛЬ
 2. ОБЩ_ЧИСЛО. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ОБЩ_ЧИСЛО. ПОКАЗАТЬ ЗНАЧЕНИЕ
 4. ОБЩ_ЧИСЛО. УВЕЛИЧИТЬ НА ЧИСЛО
1. ШИРИНА. УСТАНОВИТЬ В НОЛЬ
 2. ШИРИНА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ШИРИНА. ПОКАЗАТЬ ЗНАЧЕНИЕ
1. ОБЩ_ШИРИНА. УСТАНОВИТЬ В НОЛЬ
 2. ОБЩ_ШИРИНА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ОБЩ_ШИРИНА. ПОКАЗАТЬ ЗНАЧЕНИЕ
 4. ОБЩ_ШИРИНА. УВЕЛИЧИТЬ НА ШИРИНА
1. ВЫСОТА. УСТАНОВИТЬ В НОЛЬ
 2. ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ВЫСОТА. ПОКАЗАТЬ ЗНАЧЕНИЕ
1. ОБЩ_ВЫСОТА. УСТАНОВИТЬ В НОЛЬ
 2. ОБЩ_ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ОБЩ_ВЫСОТА. ПОКАЗАТЬ ЗНАЧЕНИЕ
 4. ОБЩ_ВЫСОТА. УВЕЛИЧИТЬ НА ВЫСОТА
1. ПЛОЩАДЬ. УСТАНОВИТЬ В НОЛЬ
 2. ПЛОЩАДЬ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ПЛОЩАДЬ. ПОКАЗАТЬ ЗНАЧЕНИЕ
 4. ПЛОЩАДЬ. УСТАНОВИТЬ В ШИРИНА ВЫСОТА
1. ОБЩ_ПЛОЩАДЬ. УСТАНОВИТЬ В НОЛЬ
 2. ОБЩ_ПЛОЩАДЬ. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 3. ОБЩ_ПЛОЩАДЬ. ПОКАЗАТЬ ЗНАЧЕНИЕ
 4. ОБЩ_ПЛОЩАДЬ. УВЕЛИЧИТЬ НА ПЛОЩАДЬ

1. ПЕРИМЕТР. УСТАНОВИТЬ В НОЛЬ
2. ПЕРИМЕТР. УВЕЛИЧИТЬ НА ЕДИНИЦУ
3. ПЕРИМЕТР. ПОКАЗАТЬ ЗНАЧЕНИЕ
4. ПЕРИМЕТР. УВЕЛИЧИТЬ НА ШИРИНА
5. ПЕРИМЕТР. УВЕЛИЧИТЬ НА ВЫСОТУ

1. ОБЩ_ПЕРИМЕТР. УСТАНОВИТЬ В НОЛЬ
2. ОБЩ_ПЕРИМЕТР. УВЕЛИЧИТЬ НА ЕДИНИЦУ
3. ОБЩ_ПЕРИМЕТР. ПОКАЗАТЬ ЗНАЧЕНИЕ
4. ОБЩ_ПЕРИМЕТР. УВЕЛИЧИТЬ НА ПЕРИМЕТР

Предписания РОБОТа типа действие вызывают очевидные действия, управляющие движением РОБОТа. Предписания РОБОТа, вырабатывающие значение, отвечают на вопрос: свободно ли впереди (слева, справа) от РОБОТа на расстоянии 1 шага в зависимости от его положения в квадранте.

Предписание ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. УСТАНОВИТЬ В НОЛЬ устанавливает в ноль значения всех 10 счетчиков. Ответы ВСТРОЕНОГО ВЫЧИСЛИТЕЛЯ на вопросы типа КРАЙ вырабатываются не в соответствии с истинным расположением РОБОТа в квадранте, а зависят лишь от выполненных ранее предписаний ЗАПОМНИТЬ ЧТО ... При этом 3 признака: РЯД СВОБОДЕН, НАЙДЕН УГОЛ, КРАЙ — являются взаимосвязанными — всегда выполняется лишь один из них. Предписание, устанавливающее истинность одного из них, одновременно устанавливает ложность двух других. По предписанию БОРТОВОЙ ВЫЧИСЛИТЕЛЬ. УСТАНОВИТЬ В НОЛЬ устанавливается признак РЯД СВОБОДЕН (а значит, НЕ КРАЙ и НЕ НАЙДЕН УГОЛ).

Задание № 1 для исполнителя РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ

Напишите программу, для которой указаны ДАНО и ПОЛУЧИТЬ. Во всех вариантах имеется в виду препятствие, не прилегающее к стене. При выполнении на машине поставьте РОБОТа в необходимое начальное положение средствами редактирования лабиринта. Обратите внимание, что в программе должно отсутствовать предписание РОБОТ. НАЧАТЬ РАБОТУ..., так как оно устанавливает РОБОТа в левый верхний угол квадранта лицом на восток.

1. ДАНО: РОБОТ находится рядом с западным нижним краем препятствия лицом на восток
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
2. ДАНО: РОБОТ находится рядом с западным нижним краем препятствия лицом на восток
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
3. ДАНО: РОБОТ находится рядом с западным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан

4. ДАНО: РОБОТ находится рядом с западным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
5. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на восток
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
6. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на восток
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
7. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на юг
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
8. ДАНО: РОБОТ находится рядом с западным верхним краем препятствия лицом на юг
ПОЛУЧИТЬ: РОБОТ обошел вокруг препятствия; площадь подсчитана
9. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
10. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
11. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
12. ДАНО: РОБОТ находится рядом с восточным нижним краем препятствия лицом на север
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
13. ДАНО: РОБОТ находится рядом с восточным верхним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан
14. ДАНО: РОБОТ находится рядом с восточным верхним краем препятствия лицом на запад
ПОЛУЧИТЬ: РОБОТ обошел препятствие; площадь подсчитана
15. ДАНО: РОБОТ находится рядом с восточным верхним краем препятствия лицом на юг
ПОЛУЧИТЬ: РОБОТ обошел препятствие; периметр подсчитан

Образец выполнения задания. Вариант 15

Обходить препятствие будем по часовой стрелке. Для исключения неопределенности при описании положения РОБОТА, находящегося около угла препятствия, договоримся, что слово «рядом» будет означать «в свободной клетке справа или слева, на той же горизонтали», а «под» и «над» — «на клетку южнее» или «на клетку севернее».

ПРОГРАММА ОБХОДА ПРЕПЯТСТВИЯ С ПОДСЧЕТОМ ПЕРИМЕТРА
ДАНО: ..РОБОТ НАХОДИТСЯ РЯДОМ С ВОСТОЧНЫМ ВЕРХНИМ
 ..КРАЕМ ПРЕПЯТСТВИЯ ЛИЦОМ НА ЮГ

.. „ПРЕПЯТСТВИЕ НЕ ПРИМЫКАЕТ К СТЕНЕ
 ПОЛУЧИТЬ: „РОБОТ ОБОШЕЛ ПРЕПЯТСТВИЕ; ПЕРИМЕТР ПОДСЧИТАН
 ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. УСТАНОВИТЬ В НОЛЬ
 ОБХОДЧИК. СПУСТИТЬСЯ ВДОЛЬ ПРЕПЯТСТВИЯ С ПОДСЧЕТОМ
 ВЫСОТЫ
 ОБХОДЧИК. ПРОЙТИ ПОД ПРЕПЯТСТВИЕМ С ПОДСЧЕТОМ ШИРИНЫ
 ОБХОДЧИК. ПОДНЯТЬСЯ ВДОЛЬ ПРЕПЯТСТВИЯ
 ОБХОДЧИК. ПРОЙТИ НАД ПРЕПЯТСТВИЕМ
 „
 ПЕРИМЕТР. УВЕЛИЧИТЬ НА ВЫСОТУ
 ПЕРИМЕТР. УВЕЛИЧИТЬ НА ВЫСОТУ
 ПЕРИМЕТР. УВЕЛИЧИТЬ НА ШИРИНА
 ПЕРИМЕТР. УВЕЛИЧИТЬ НА ШИРИНА
 ПЕРИМЕТР. ПОКАЗАТЬ ЗНАЧЕНИЕ
 „
 РОБОТ. КОНЧИТЬ РАБОТУ
 КОНЕЦ ПРОГРАММЫ
 =====

ИСПОЛНИТЕЛЬ ОБХОДЧИК

. . СП:
 СПУСТИТЬСЯ ВДОЛЬ ПРЕПЯТСТВИЯ С ПОДСЧЕТОМ ВЫСОТЫ
 ПРОЙТИ ПОД ПРЕПЯТСТВИЕМ С ПОДСЧЕТОМ ШИРИНЫ
 ПОДНЯТЬСЯ ВДОЛЬ ПРЕПЯТСТВИЯ
 ПРОЙТИ НАД ПРЕПЯТСТВИЕМ
 КОНЕЦ ОПИСАНИЙ
 =====
 ПРЕДПИСАНИЕ СПУСТИТЬСЯ ВДОЛЬ ПРЕПЯТСТВИЯ
 С ПОДСЧЕТОМ ВЫСОТЫ
 ДАНО: „РОБОТ НАХОДИТСЯ РЯДОМ С ВОСТОЧНЫМ ВЕРХНИМ
 „КРАЕМ ПРЕПЯТСТВИЯ ЛИЦОМ НА ЮГ
 ПОЛУЧИТЬ: „РОБОТ НАХОДИТСЯ ПОД ВОСТОЧНЫМ НИЖНИМ
 „КРАЕМ ПРЕПЯТСТВИЯ ЛИЦОМ НА ЗАПАД. ВЫСОТА
 „ПРЕПЯТСТВИЯ ПОДСЧИТАНА
 =====
 ЦИКЛ ПОКА РОБОТ. СПРАВА ЗАНЯТО
 ВЫПОЛНЯТЬ
 РОБОТ. СДЕЛАТЬ ШАГ
 ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 КОНЕЦ ЦИКЛА
 „СПРАВА СВОБОДНО, Т.Е. СПУСТИЛИСЬ ВДОЛЬ ПРЕПЯТСТВИЯ
 РОБОТ. ПОВЕРНУТЬСЯ НА ЗАПАД
 РОБОТ. СДЕЛАТЬ ШАГ
 КОНЕЦ ПРЕДПИСАНИЯ
 =====
 ПРЕДПИСАНИЕ ПРОЙТИ ПОД ПРЕПЯТСТВИЕМ С ПОДСЧЕТОМ ШИРИНЫ
 ДАНО: „РОБОТ ПОД ВОСТОЧНЫМ НИЖНИМ КРАЕМ
 „ПРЕПЯТСТВИЯ ЛИЦОМ НА ЗАПАД
 ПОЛУЧИТЬ: „РОБОТ РЯДОМ С ЗАПАДНЫМ НИЖНИМ КРАЕМ
 „ПРЕПЯТСТВИЯ ЛИЦОМ НА СЕВЕР;
 „ШИРИНА ПРЕПЯТСТВИЯ ПОДСЧИТАНА
 =====
 ЦИКЛ ПОКА РОБОТ. СПРАВА ЗАНЯТО
 ВЫПОЛНЯТЬ
 РОБОТ. СДЕЛАТЬ ШАГ
 ШИРИНА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
 КОНЕЦ ЦИКЛА
 „СПРАВА СВОБОДНО, Т.Е. ПРОШЛИ ПОД ПРЕПЯТСТВИЕМ
 РОБОТ. ПОВЕРНУТЬСЯ НА ЗАПАД

```

. . РОБОТ. СДЕЛАТЬ ШАГ
. . КОНЕЦ ПРЕДПИСАНИЯ

. ПРЕДПИСАНИЕ ПОДНЯТЬСЯ ВДОЛЬ ПРЕПЯТСТВИЯ
. ДАНО: .. РОБОТ РЯДОМ С ЗАПАДНЫМ НИЖНИМ КРАЕМ
. . . ПРЕПЯТСТВИЯ ЛИЦОМ НА ЗАПАД
. . . ПОЛУЧИТЬ: .. РОБОТ НАД ЗАПАДНЫМ ВЕРХНИМ КРАЕМ
. . . . . ПРЕПЯТСТВИЯ ЛИЦОМ НА ВОСТОК

. . . ЦИКЛ ПОКА РОБОТ. СПРАВА СВОБОДНО
. . . . . ВЫПОЛНЯТЬ
. . . . . РОБОТ. СДЕЛАТЬ ШАГ
. . . . . КОНЕЦ ЦИКЛА
. . . . . . СПРАВА СВОБОДНО, Т.Е. ПОДНЯЛИСЬ ВДОЛЬ ПРЕПЯТСТВИЯ
. . . . . РОБОТ. ПОВЕРНУТЬСЯ НАПРАВО
. . . . . РОБОТ. СДЕЛАТЬ ШАГ
. . . . . КОНЕЦ ПРЕДПИСАНИЯ

. ПРЕДПИСАНИЕ ПРОЙТИ НАД ПРЕПЯТСТВИЕМ
. . ДАНО: .. РОБОТ НАД ЗАПАДНЫМ ВЕРХНИМ КРАЕМ
. . . . . ПРЕПЯТСТВИЯ ЛИЦОМ НА ВОСТОК
. . . ПОЛУЧИТЬ: .. РОБОТ РЯДОМ С ВОСТОЧНЫМ ВЕРХНИМ КРАЕМ
. . . . . ПРЕПЯТСТВИЯ ЛИЦОМ НА ЮГ

. . . ЦИКЛ ПОКА РОБОТ. СПРАВА ЗАНЯТО
. . . . . ВЫПОЛНЯТЬ
. . . . . РОБОТ. СДЕЛАТЬ ШАГ
. . . . . КОНЕЦ ЦИКЛА
. . . . . . СПРАВА СВОБОДНО, Т.Е. ПРОШЛИ НАД ПРЕПЯТСТВИЕМ
. . . . . РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
. . . . . РОБОТ. СДЕЛАТЬ ШАГ
. . . . . КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ

```

Вопросы.

Можно ли упростить эту программу, используя конструкцию ЦИКЛ 4 РАЗА ВЫПОЛНИТЬ? Можно ли таким же образом упростить аналогичную программу для подсчета площади?

Задание № 2 для исполнителя РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ

Напишите программу прохождения одного занятого ряда. Конструкция ДАНО общая для всех вариантов. Конструкция ПОЛУЧИТЬ содержит различную для каждого варианта часть.

ДАНО: РОБОТ находится рядом с западным верхним краем занятого ряда.

ПОЛУЧИТЬ: РОБОТ прошел ряд слева от занятого. В занятом ряду подсчитано...

- 1 — количество всех препятствий;
- 2 — количество всех проходов;
- 3 — общая высота всех препятствий;
- 4 — общая высота всех проходов;
- 5 — количество препятствий, кроме тех, что у верхнего края;
- 6 — количество проходов, кроме тех, что у верхнего края;

- 7 — общая высота препятствий, кроме тех, что у верхнего края;
- 8 — общая высота проходов, кроме тех, что у верхнего края;
- 9 — количество препятствий, кроме тех, что у нижнего края;
- 10 — количество проходов, кроме тех, что у нижнего края;
- 11 — общая высота препятствий, кроме тех, что у нижнего края;
- 12 — общая высота проходов, кроме тех, что у нижнего края;
- 13 — количество препятствий, кроме тех, что у краев;
- 14 — количество проходов, кроме тех, что у краев;
- 15 — общая высота препятствий, кроме тех, что у краев.

Образец выполнения. Вариант 15

Мы будем спускаться вдоль ряда, проходя поочередно то рядом с проходом, то рядом с препятствием. Эти 2 действия будем повторять циклически до тех пор, пока можно двигаться в южном направлении (впереди свободно). Ряд может начинаться не с прохода, а с препятствия; такую возможность рассмотрим отдельно, до цикла. Ряд может закончиться как препятствием, так и проходом, поэтому каждый раз после прохождения прохода нужно убедиться, что рядом препятствие (слева занято). Кроме того, высоту последнего препятствия, если оно у стены, не нужно учитывать, поэтому высоту каждого препятствия мы будем подсчитывать отдельно и прибавлять к общей высоте лишь в тех случаях, когда препятствие не у стены (после прохождения препятствия слева свободно).

ПРОГРАММА ПРОХОЖДЕНИЯ ЗАНЯТОГО РЯДА
 . ДАНО: ..РОБОТ НАХОДИСЯ РЯДОМ С ЗАПАДНЫМ ВЕРХНИМ
 ..КРАЕМ ЗАНЯТОГО РЯДА
 . ПОЛУЧИТЬ: ..РОБОТ ПРОШЕЛ РЯД СЛЕВА ОТ ЗАНЯТОГО. В ЗАНЯТОМ
 ..РЯДУ ПОДСЧИТАНА ОБЩАЯ ВЫСОТА ПРЕПЯТСТВИЙ,
 . ..КРОМЕ ТЕХ, ЧТО У КРАЕВ

```
: РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
: ОБЩ_ ВЫСОТА. УСТАНОВИТЬ В НОЛЬ
: ЕСЛИ РОБОТ. СЛЕВА ЗАНЯТО
: . ТО ПОМОЩНИК. ПРОЙТИ РЯДОМ С ПРЕПЯТСТВИЕМ
: . ИНАЧЕ
: . КОНЕЦ ЕСЛИ
: "
: ЦИКЛ ПОКА РОБОТ. ВПЕРЕДИ СВОБОДНО
: . ВЫПОЛНЯТЬ
: . . ПОМОЩНИК. ПРОЙТИ РЯДОМ С ПРОХОДОМ
: . . ЕСЛИ РОБОТ. СЛЕВА ЗАНЯТО
: . . . ТО ПОМОЩНИК. ПРОЙТИ РЯДОМ С ПРЕПЯТСТВИЕМ
: . . . ЕСЛИ РОБОТ. СЛЕВА СВОБОДНО
: . . . . ТО ОБЩ_ ВЫСОТА. УВЕЛИЧИТЬ НА ВЫСОТА
: . . . . ИНАЧЕ
: . . . . КОНЕЦ ЕСЛИ
: . . . ИНАЧЕ
: . . . КОНЕЦ ЕСЛИ
: . КОНЕЦ ЦИКЛА
: "
: ОБЩ_ ВЫСОТА. ПОКАЗАТЬ ЗНАЧЕНИЕ
: РОБОТ. КОНЧИТЬ РАБОТУ
```

КОНЕЦ ПРОГРАММЫ

ИСПОЛНИТЕЛЬ ПОМОЩНИК

. СП:

. . ПРОЙТИ РЯДОМ С ПРЕПЯТСТВИЕМ
. . ПРОЙТИ РЯДОМ С ПРОХОДОМ

КОНЕЦ ОПИСАНИЙ

. . ПРЕДПИСАНИЕ ПРОЙТИ РЯДОМ С ПРЕПЯТСТВИЕМ
. . ДАНО: ..РОБОТ НАХОДИТСЯ РЯДОМ С ЗАПАДНЫМ ВЕРХНИМ
. . ..КРАЕМ ПРЕПЯТСТВИЯ
. . ПОЛУЧИТЬ:..РОБОТ НАХОДИТСЯ РЯДОМ С ЗАПАДНЫМ ВЕРХНИМ
. . ..КРАЕМ ПРОХОДА ИЛИ У НИЖНЕЙ СТЕНЫ

. . . ВЫСОТА. УСТАНОВИТЬ В НОЛЬ
. . . ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. . . ЦИКЛ ПОКА РОБОТ. СЛЕВА ЗАНЯТО И РОБОТ. ВПЕРЕДИ СВОБОДНО
. . . ВЫПОЛНЯТЬ
. . . . РОБОТ. СДЕЛАТЬ ШАГ
. . . . ЕСЛИ РОБОТ. СЛЕВА ЗАНЯТО
. ТО ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. ИНАЧЕ
. . . . КОНЕЦ ЕСЛИ
. . . . КОНЕЦ ЦИКЛА
. . . . КОНЕЦ ПРЕДПИСАНИЯ

. . ПРЕДПИСАНИЕ ПРОЙТИ РЯДОМ С ПРОХОДОМ
. . ДАНО: ..РОБОТ НАХОДИТСЯ РЯДОМ С ЗАПАДНЫМ ВЕРХНИМ
. . ..КРАЕМ ПРОХОДА
. . ПОЛУЧИТЬ:..РОБОТ НАХОДИТСЯ РЯДОМ С ЗАПАДНЫМ ВЕРХНИМ
. . ..КРАЕМ ПРОХОДА ИЛИ У НИЖНЕЙ СТЕНЫ

. . . ЦИКЛ ПОКА РОБОТ. СЛЕВА СВОБОДНО И РОБОТ. ВПЕРЕДИ СВОБОДНО
. . . ВЫПОЛНЯТЬ
. . . . РОБОТ. СДЕЛАТЬ ШАГ
. . . . КОНЕЦ ЦИКЛА
. . . . КОНЕЦ ПРЕДПИСАНИЯ

КОНЕЦ ИСПОЛНИТЕЛЯ

**Задание № 3 для исполнителя РОБОТ
с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ**

Напишите программу для прохождения всего лабиринта с подсчетом указанных характеристик:

ДАНО: имя квадранта равно Б28/4-БИС

ПОЛУЧИТЬ: РОБОТ прошел весь лабиринт и находится в северо-восточном углу. Подсчитано ...

- 1 — количество полос с препятствиями;
- 2 — количество свободных полос;
- 3 — количество рядов с препятствиями;
- 4 — количество свободных рядов;
- 5 — количество всех препятствий;
- 6 — количество всех проходов;
- 7 — количество всех квадратных препятствий;
- 8 — количество всех препятствий, кроме квадратных;
- 9 — количество всех препятствий, у которых ширина меньше высоты;

- 10 — общий периметр всех препятствий;
- 11 — общий периметр всех квадратных препятствий;
- 12 — общий периметр препятствий, у которых ширина меньше высоты;
- 13 — общая площадь всех квадратных препятствий;
- 14 — общая площадь всех препятствий, кроме квадратных;
- 15 — общая площадь всех препятствий.

Образец выполнения. Вариант 15

Пройдем свободную полосу, а затем будем поочередно проходить то полосу препятствий, то свободную полосу, пока не дойдем до восточной стены. Двигаться на восток будем вдоль северной стены. Перед каждым шагом на восток будем проводить разведку ближайшего восточного ряда — является ли он занятм или свободным. При разведке ряда будем идти на юг рядом с ним до тех пор, пока не дойдем до стены или не обнаружим угол — занятую клетку ниже свободной или свободную клетку ниже занятой. Дойдя до южной стены или обнаружив угол, вернемся к северной стене. Если угол не обнаружен и верхняя в ряду клетка свободна — значит, ряд свободен. Если угол не обнаружен и верхняя клетка занята — значит, это восточная стена. Если обнаружен угол — значит, это ряд с препятствиями; он начинает полосу препятствий неизвестной ширины.

При прохождении полосы препятствий найдем сначала ближайший к северной стене проход. Если есть препятствие у северной стены, будем двигаться на восток вдоль южного его края, подсчитывая ширину. Если нет препятствия у северной стены, найдем сначала самое северное препятствие и будем двигаться вдоль северного его края с подсчетом ширины. Пройдя полосу препятствий по проходу на восток, вернемся к северной стене.

Теперь нам нужно вычислить общую площадь препятствий в пройденной полосе препятствий. Спустимся до южной стены вдоль восточного края занятого ряда, подсчитывая занятые справа клетки — это и будет общая ширина препятствий в ряду. Потом вернемся к северной стене.

ПРОГРАММА ПОДСЧЕТ ПЛОЩАДИ ПРЕПЯТСТВИЙ

- . ДАНО: „ИМЯ КВАДРАНТА РАВНО Б28/4-БИС
- . ПОЛУЧИТЬ: „ВЫЧИСЛЕНЫ И НАПЕЧАТАНЫ ПЛОЩАДИ ПРЕПЯТСТВИЙ
- . „В КАЖДОЙ ЗАНЯТОЙ ПОЛОСЕ И СУММАРНАЯ ПЛОЩАДЬ
- . „ВСЕХ ПРЕПЯТСТВИЙ. РАБОТА РОБОТА ОКОНЧЕНА
-
- . ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. УСТАНОВИТЬ В НОЛЬ
- . РОБОТ. НАЧАТЬ РАБОТУ В КВАДРАНТЕ „Б28/4-БИС
- . ОБРАБОТЧИК ПОЛОС. ПРОХОД СВОБОДНОЙ ПОЛОСЫ
- . ЦИКЛ ПОКА ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. НЕ КРАЙ
-
- . . . „РОБОТ В СЕВЕРО-ЗАПАДНОМ УГЛУ СВОБОДНОЙ ПОЛОСЫ
- . . . „РАЗВЕДАН РЯД ВОСТОЧНЕЕ РОБОТА

.. „ОБЩ_ПЛОЩАДЬ = ПЛОЩАДЬ ПРЕПЯТСТВИЙ В ПРОЙДЕННОЙ
ЧАСТИ КВАДРАНТА
.. ОБРАБОТЧИК ПОЛОС. ПРОХОД ПОЛОСЫ ПРЕПЯТСТВИЙ
.. ОБРАБОТЧИК ПОЛОС. ВЫЧИСЛЕНИЕ ПЕЧАТЬ ПЛОЩАДИ
.. „ПЛОЩАДЬ = ПЛОЩАДЬ ПРЕПЯТСТВИЙ В ОБРАБОТАННОЙ ПОЛОСЕ
.. „ОБЩ_ПЛОЩАДЬ. УВЕЛИЧИТЬ НА ПЛОЩАДЬ
.. ОБРАБОТЧИК ПОЛОС. ПРОХОД СВОБОДНОЙ ПОЛОСЫ
КОНЕЦ ЦИКЛА
ОБЩ_ПЛОЩАДЬ. ПОКАЗАТЬ ЗНАЧЕНИЕ
РОБОТ. КОНЧИТЬ РАБОТУ
КОНЕЦ ПРОГРАММЫ

ИСПОЛНИТЕЛЬ ОБРАБОТЧИК ПОЛОС
СП:
.. ПРОХОД СВОБОДНОЙ ПОЛОСЫ
.. ПРОХОД ПОЛОСЫ ПРЕПЯТСТВИЙ
.. ВЫЧИСЛЕНИЕ ПЕЧАТЬ ПЛОЩАДИ
.. РАЗВЕДКА РЯДА ВОСТОЧНЕЕ РОБОТА
ИСПОЛЬЗУЕМЫЕ ИСПОЛНИТЕЛИ:
.. РОБОТ
.. ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ
КОНЕЦ ОПИСАНИЙ
ПРЕДПИСАНИЕ ПРОХОД СВОБОДНОЙ ПОЛОСЫ
ДАНО: .. „РОБОТ В СЕВЕРО-ЗАПАДНОМ УГЛУ СВОБОДНОЙ ПОЛОСЫ
ПОЛУЧИТЬ: .. „РОБОТ В СЕВЕРО-ВОСТОЧНОМ УГЛУ ТОЙ ЖЕ ПОЛОСЫ
.. „РАЗВЕДАН РЯД ВОСТОЧНЕЕ РОБОТА

ОБРАБОТЧИК ПОЛОС. РАЗВЕДКА РЯДА ВОСТОЧНЕЕ РОБОТА
ЦИКЛ ПОКА ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. РЯД СВОБОДЕН
.. ВЫПОЛНЯТЬ
.. „РОБОТ У СЕВЕРНОГО КРАЯ СВОБОДНОГО РЯДА
.. „РАЗВЕДАН РЯД ВОСТОЧНЕЕ РОБОТА
.. РОБОТ. ПОВЕРНУТЬСЯ НА ВОСТОК
.. РОБОТ. СДЕЛАТЬ ШАГ
.. ОБРАБОТЧИК ПОЛОС. РАЗВЕДКА РЯДА ВОСТОЧНЕЕ РОБОТА
.. КОНЕЦ ЦИКЛА
.. „РЯД ВОСТОЧНЕЕ РОБОТА РАЗВЕДАН И НЕ СВОБОДЕН
КОНЕЦ ПРЕДПИСАНИЯ
ПРЕДПИСАНИЕ ПРОХОД ПОЛОСЫ ПРЕПЯТСТВИЙ
ДАНО: .. „РОБОТ У СЕВЕРНОГО КРАЯ В РЯДУ
.. „ЗАПАДНЕЕ ПОЛОСЫ ПРЕПЯТСТВИЙ
ПОЛУЧИТЬ: .. „РОБОТ У СЕВЕРНОГО КРАЯ В РЯДУ
.. „ВОСТОЧНЕЕ ПОЛОСЫ ПРЕПЯТСТВИЙ
.. „ШИРИНА = ШИРИНА ПРОЙДЕННОЙ ПОЛОСЫ

ШИРИНА. УСТАНОВИТЬ В НОЛЬ
РОБОТ. ПОВЕРНУТЬСЯ НА ВОСТОК
ЕСЛИ РОБОТ. ВПЕРЕДИ СВОБОДНО
.. ТО .. „ЕСТЬ ПРОХОД У ВЕРХНЕГО КРАЯ
.. РОБОТ. СДЕЛАТЬ ШАГ
.. РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
.. РОБОТ. ШАГАТЬ ДО УПОРА
.. РОБОТ. ПОВЕРНУТЬСЯ НА ВОСТОК
ЦИКЛ ПОКА РОБОТ. СПРАВА ЗАНЯТО
.. РОБОТ. СДЕЛАТЬ ШАГ
.. ШИРИНА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
КОНЕЦ ЦИКЛА
ИНАЧЕ .. „НЕТ ПРОХОДА У ВЕРХНЕГО КРАЯ
РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
ЦИКЛ ПОКА РОБОТ. СЛЕВА ЗАНЯТО

ВЫПОЛНЯТЬ „ПОИСК ПРОХОДА
РОБОТ. СДЕЛАТЬ ШАГ
КОНЕЦ ЦИКЛА
РОБОТ. ПОВЕРНУТЬСЯ НА ВОСТОК
РОБОТ. СДЕЛАТЬ ШАГ
ЦИКЛ ПОКА РОБОТ. СЛЕВА ЗАНЯТО
ВЫПОЛНЯТЬ „ВДОЛЬ ЮЖНОГО КРАЯ ПРЕПЯТСТВИЯ
РОБОТ. СДЕЛАТЬ ШАГ
ШИРИНА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
КОНЕЦ ЦИКЛА
КОНЕЦ ЕСЛИ
РОБОТ. ПОВЕРНУТЬСЯ НА СЕВЕР „ВОЗВРАЩЕНИЕ К СЕВЕРНОЙ
РОБОТ. ШАГАТЬ ДО УПОРА „СТЕНЕ КВАДРАНТА
КОНЕЦ ПРЕДПИСАНИЯ
ПРЕДПИСАНИЕ РАЗВЕДКА РЯДА ВОСТОЧНЕЕ РОБОТА
ДАНО: „РОБОТ У СЕВЕРНОГО КРАЯ В СВОБОДНОМ РЯДУ
ПОЛУЧИТЬ: „РАЗВЕДАН РЯД ВОСТОЧНЕЕ РОБОТА
„РЕЗУЛЬТАТЫ РАЗВЕДКИ — ВО ВСТРОЕННОМ
„ВЫЧИСЛИТЕЛЕ
„ПОЛОЖЕНИЕ РОБОТА НЕ ИЗМЕНЕНО

РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
ЕСЛИ РОБОТ. СЛЕВА СВОБОДНО
ТО ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО РЯД
СВОБОДЕН
ИНАЧЕ ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО КРАЙ
КОНЕЦ ЕСЛИ
ЦИКЛ ПОКА РОБОТ. ВПЕРЕДИ СВОБОДНО И
ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. НЕ НАЙДЕН УГОЛ
ВЫПОЛНЯТЬ
РОБОТ. СДЕЛАТЬ ШАГ
ВЫБОР
??? ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. КРАЙ И РОБОТ. СЛЕВА СВОБОДНО
=> ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО НАЙДЕН УГОЛ
L
??? ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. РЯД СВОБОДЕН И
РОБОТ. СЛЕВА ЗАНЯТО
=> ВСТРОЕННЫЙ ВЫЧИСЛИТЕЛЬ. ЗАПОМНИТЬ ЧТО НАЙДЕН УГОЛ
L
КОНЕЦ ВЫБОРА
КОНЕЦ ЦИКЛА
РОБОТ. ПОВЕРНУТЬСЯ НА СЕВЕР „ВОЗВРАЩЕНИЕ К СЕВЕРНОМУ
РОБОТ. ШАГАТЬ ДО УПОРА „КРАЮ КВАДРАНТА
КОНЕЦ ПРЕДПИСАНИЯ
ПРЕДПИСАНИЕ ВЫЧИСЛЕНИЕ ПЕЧАТЬ ПЛОЩАДИ
ДАНО: „РОБОТ У СЕВЕРНОГО КРАЯ ВОСТОЧНЕЕ ПОЛОСЫ
„ПРЕПЯТСТВИЙ
„ШИРИНА = ШИРИНА ЭТОЙ ПОЛОСЫ
ПОЛУЧИТЬ: „ПЛОЩАДЬ = ПЛОЩАДЬ ПРЕПЯТСТВИЙ ЭТОЙ ПОЛОСЫ
„ПЛОЩАДЬ ПРЕПЯТСТВИЙ ПОЛОСЫ НАПЕЧАТАНА
„ПОЛОЖЕНИЕ РОБОТА НЕ ИЗМЕНИЛОСЬ

РОБОТ. ПОВЕРНУТЬСЯ НА ЮГ
ВЫСОТА. УСТАНОВИТЬ В НОЛЬ
ЕСЛИ РОБОТ. СПРАВА ЗАНЯТО
ТО ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
ИНАЧЕ
КОНЕЦ ЕСЛИ
ЦИКЛ ПОКА РОБОТ. ВПЕРЕДИ СВОБОДНО
ВЫПОЛНЯТЬ

```
. . . „ВЫСОТА=ВЫСОТА ПРЕПЯТСТВИЙ В ПРОЙДЕННОЙ ЧАСТИ ПОЛОСЫ
. . . РОБОТ. СДЕЛАТЬ ШАГ
. . . ЕСЛИ РОБОТ. СПРАВА ЗАНЯТО
. . . ТО ВЫСОТА. УВЕЛИЧИТЬ НА ЕДИНИЦУ
. . . ИНАЧЕ
. . . КОНЕЦ ЕСЛИ
. . . КОНЕЦ ЦИКЛА
. . ПЛОЩАДЬ. УСТАНОВИТЬ В ШИРИНА ВЫСОТА
. . ПЛОЩАДЬ. ПОКАЗАТЬ ЗНАЧЕНИЕ
. . РОБОТ. ПОВЕРНУТЬСЯ НА СЕВЕР
. . РОБОТ. ШАГАТЬ ДО УПОРА
. . КОНЕЦ ПРЕДПИСАНИЯ
КОНЕЦ ИСПОЛНИТЕЛЯ
```

Послесловие

Мы, с вами заканчиваем изучение первой и самой важной части нашего курса, посвященной основным понятиям программирования. За это время мы написали и выполнили на ЭВМ немало программ. Хотя каждая программа была написана для какого-то конкретного исполнителя, мы не имели дела ни с роботом, умеющим поворачиваться и шагать, ни с ручным калькулятором. Их роль выполняла ЭВМ. Для каждого исполнителя в ЭВМ имелась программа, полностью имитирующая его действия. Постараемся разобраться, как происходило выполнение программ в ЭВМ.

Между нашей программой и программой, имитирующей нужный нам исполнитель, всегда находился посредник — некоторая специальная программа. Поскольку программа представляет собой либо последовательность предписаний, либо некоторое формальное описание ее, роль посредника сводилась к тому, чтобы из формального описания (содержащего комментарии, управляющие конструкции и т.п.) получить настоящую последовательность предписаний исполнителя. Поэтому программа-посредник

— умеет распознавать допустимые языком МИНИ конструкции, не являющиеся предписаниями (комментарии, управляющие конструкции, конструкции ПРЕДПИСАНИЕ — КОНЕЦ ПРЕДПИСАНИЯ и т.д.);

— проверяет корректность употребления языковых конструкций (соответствие ЕСЛИ — КОНЕЦ ЕСЛИ, ЦИКЛ — КОНЕЦ ЦИКЛА и т.д.);

— при наличии ошибок выдает о них сообщение, при отсутствии — строит последовательность предписаний исполнителя.

Программа, имитирующая работу исполнителя, хранит в памяти состояние всех объектов исполнителя, изменяет их при выполнении очередного предписания и отображает некоторым образом состояние объектов на экране дисплея или на бумаге.

Конечно, при выполнении наших заданий на ЭВМ работали не только эти две программы. ЭВМ представляет собой исполнитель с очень небогатыми возможностями: она может сло-

жить два числа, хранящиеся в двоичном виде; проверить, будет ли результат арифметической операции больше нуля; прочитать один символ с экрана дисплея в память машины и т.п. Как всегда, когда нужно выполнить сложную задачу при помощи простого исполнителя, работала целая иерархия исполнителей: все необходимые нам сложные функции были реализованы в конце концов на базе простых аппаратных возможностей машины.

(спускаться вдоль западной стены; двигаться по ближайшему проходу и т.п.), тогда программа была бы, конечно, другой.

2. Мы могли бы придумать такой алгоритм, что возможностей нашего исполнителя не хватило бы, пришлось бы подыскивать другой исполнитель (например, умеющий находить ближайший проход).

3. Наш исполнитель мог бы обладать другими возможностями, например, уметь делать 10 шагов или поворачиваться только налево, тогда программа (даже при использовании одного и того же алгоритма) выглядела бы иначе.

=.....=

=.....=

=XXXXXX=

=.XXXXXX.=

=.....=

=.....=.=

=XXXXXX.=

=XXXXXX.=

=.....=

.=XXXXXX.=

=XXXXXX.=

=.....=

=.....=

=XX XXXXXXXX.=

=XX XXXXXXXX.=

=.....=

=.....=

=XX XXXXXXXX.=

=XX XXXXXXXX.=

=.....=

=.....=

=XXXXXX.=

=XXXXXX.=

=.....=

=.....=

=.....=

=.....=

=====XXXXXX=====

=====XXXXXX===== -(запасные)

АЛЬФА-ПРАКТИКУМ. МЕХМАТ МГУ, ЛВМ. 1985

РЕЗЧИК МЕТАЛЛА
СТЕКОВЫЙ КАЛЬКУЛЯТОР

ВЫБЕРИТЕ ИСПОЛНИТЕЛЯ, С КОТОРЫМ ХОТИТЕ РАБОТАТЬ, УСТАНОВИТЕ КУРСОР В СТРОКУ С ЕГО ИМЕНЕМ КЛАВИШАМИ «ВВЕРХ» И «ВНИЗ» И НАЖМИТЕ КЛАВИШИ «СПЕЦ» «ВНИЗ» КОМАНДА «ВНУТРЬ».

ДЛЯ ОКОНЧАНИЯ РАБОТЫ В АЛЬФА-ПРАКТИКУМЕ НАЖМИТЕ КЛАВИШИ «СПЕЦ» «ВВЕРХ» КОМАНДА «ВЫЙТИ». УСПЕХОВ В РАБОТЕ!

Рис. 1

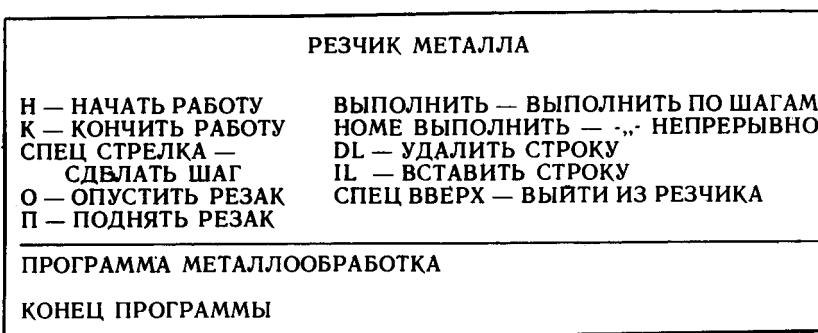


Рис. 2

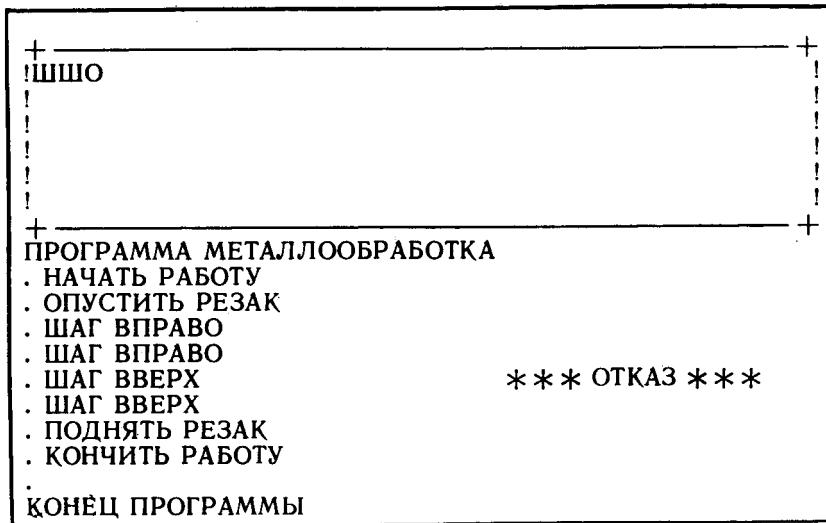


Рис. 3

В левом верхнем углу экрана расположена система предписаний исполнителя РЕЗЧИК МЕТАЛЛА. Этот базовый исполнитель работает с листом металла и имеет резак, который можно поднимать, опускать и перемещать вправо, влево, вверх и вниз. При движении в опущенном состоянии под резаком в металле образуется прорезь. По предписанию НАЧАТЬ РАБОТУ резак устанавливается в верхний левый угол листа металла в поднятом положении. Предписания для перемещения резака вставляются с помощью последовательного нажатия на клавишу [СПЕЦ], а затем на стрелку в нужном направлении. Например, при нажатии на [СПЕЦ] [ВЛЕВО] будет вставлено предписание ШАГ ВЛЕВО. Для вставки остальных предписаний следует ввести их первую букву (например, Н для НАЧАТЬ РАБОТУ).

Можно также вставлять в программу пустые строки (клавиша [IL]), удалять любые строки (клавиша [DL]) и двигать курсор по тексту программы с помощью клавиш [ВВЕРХ] и [ВНИЗ]. Обратите особое внимание на то, что вставка пустых строк и предписаний производится *перед* той строкой, в которой стоит курсор.

После ввода программы можно приступить к ее выполнению. Выполнять программу можно в двух режимах: пошаговым (клавиша [ВЫПОЛНИТЬ]) и непрерывном (клавиши [НОМЕ] [ВЫПОЛНИТЬ]). В пошаговом режиме каждое нажатие на клавишу [ВЫПОЛНИТЬ] приводит к выполнению строки, в которой стоит курсор, и к переходу на следующую строку. Выполнение в любой момент можно прервать. Для этого нужно нажать левой рукой на клавишу [CTRL] и, не отпуская ее, правой рукой нажать на клавишу С (в дальнейшем такое одновременное нажатие обозначается [CTRL-C]).

Если во время выполнения программы исполнитель не сможет выполнить требуемое действие (например, при попытке вывести резак за край листа металла), возникнет ситуация ОТКАЗ, и в строке, которую не удалось выполнить, появится надпись «*** ОТКАЗ ***» (рис. 3).

Отказ в последней (пустой) строке программы означает, что в программе отсутствует предписание КОНЧИТЬ РАБОТУ.

Исправив программу, можно запустить ее еще раз, затем снова исправить и т.д.

Пример задания: составьте программу, прорезающую в металле картинку, изображенную на рис. 4.

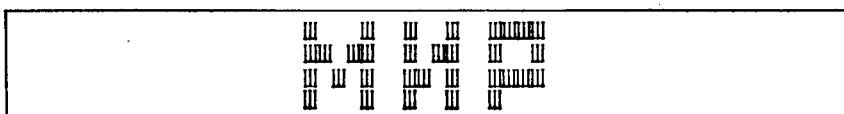


Рис. 4

Для завершения работы с РЕЗЧИКОМ МЕТАЛЛА введите команду «ВЫЙТИ» (т.е. нажмите клавишу [СПЕЦ], а затем клавишу ВВЕРХ). На экране вновь возникнет картинка рис. 1.

Для работы с исполнителем СТЕКОВЫЙ КАЛЬКУЛЯТОР следует встать на строку СТЕКОВЫЙ КАЛЬКУЛЯТОР и ввести команду «ВНУТРЬ» (т.е. нажать клавиши [СПЕЦ] и [ВНИЗ]). На экране появится картинка рис. 5.

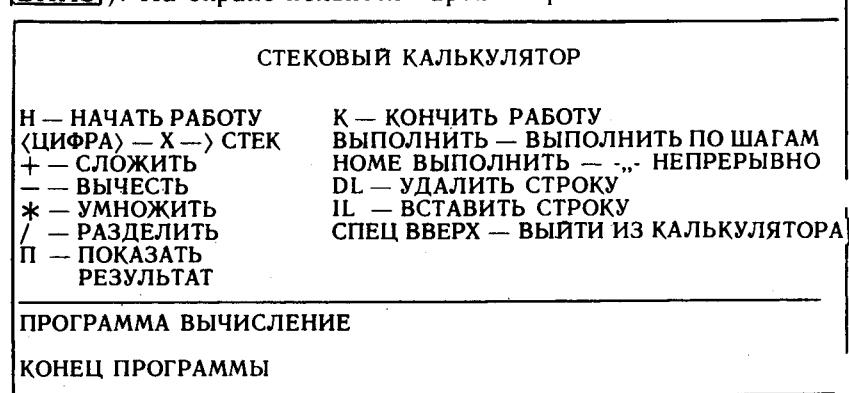


Рис. 5

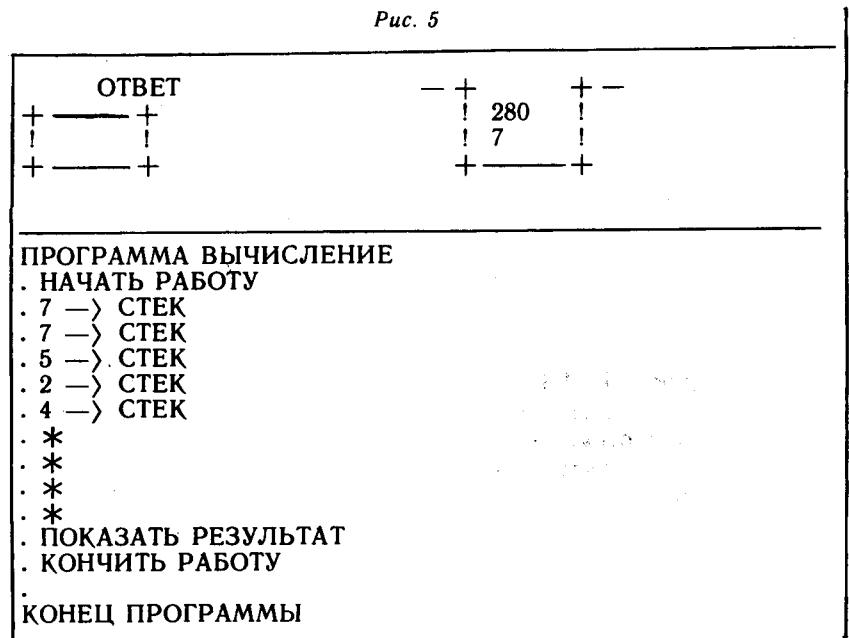


Рис. 6

Разумеется, при работе со СТЕКОВЫМ КАЛЬКУЛЯТОРОМ можно пользоваться теми же командами изменения текста программы, что и при работе с РЕЗЧИКОМ МЕТАЛЛА. Си-

стема предписаний СТЕКОВОГО КАЛЬКУЛЯТОРА позволяет помещать в стек однозначные числа (например, нажатие на клавишу 2 приводит к вставке вызова предписания 2 → СТЕК), производить арифметические операции над числами в стеке и показывать результат (вершину стека).

На рис. 6 приведен вид экрана во время пошагового выполнения программы для вычисления числа 1960.

Пример задания: составьте программу для вычисления и печати года вашего рождения.

Микромир-85

Основу любой работы программиста на ЭВМ составляет редактирование текстов. Ниже излагаются несколько приемов редактирования текстов в системе «Микромир-85».

Текст представляется в виде прямоугольника из символов фиксированной ширины (обычно 63 или 132 буквы) и неограниченной высоты. Вы рассматриваете этот текст через «окно» — экран терминала. На экране имеется курсор, соответствующий вашей позиции в тексте. С помощью клавиш-стрелок курсор можно перемещать по тексту. При этом окно будет двигаться по тексту так, чтобы всегда содержать курсор. Для замены одного символа другим следует подогнать курсор к нужному месту и нажать клавишу с новым символом. При необходимости можно удалить символ, под которым стоит курсор (клавишей «УДАЛИТЬ СИМВОЛ» **[DC]**) или вставить перед ним пробел (клавишей «ВСТАВИТЬ СИМВОЛ» **[IC]**). Как и в АЛЬФА-практикуме можно удалить строку или вставить пустую строку клавишами **[DL]** и **[IL]** соответственно. Для запоминания символов используется клавиша **[CTRL-F]**. Запомнив несколько символов, их можно все вместе вставить в текст с помощью клавиши **[CTRL-R]**. Для запоминания и вспоминания строк используются клавиши **[CTRL-G]** и **[CTRL-V]** соответственно.

Для долговременного хранения текстов на ЭВМ используются специальные хранилища информации — магнитные диски. Информация в этих хранилищах обычно сохраняется даже в случае выключения или отказа ЭВМ. Для завершения работы с сохранением результатов на диске служит команда «ВЫЙТИ» (**[СПЕЦ] [ВВЕРХ]**). Иногда вы желаете отменить результат редактирования (нечаянно удалив, например, десяток строк). В этом случае, нажав на клавиши **[CTRL/T]** **[CTRL/C]**, можно закончить работу, оставив исходный текст нетронутым. Для сохранения результатов работы на диске без выхода из практикума нажмите **[HOME]** **[СПЕЦ]**.

«Микромир-85» имеет также много других возможностей, облегчающих и ускоряющих обработку текстов. Их можно узнать из соответствующих инструкций.

Имеется также игра, обучающая быстрой и безошибочной

работе на клавиатуре терминала, и игра «попрыгун», обучающая командам перемещения курсора в системе «Микромир-85».

БЕТА-практикум

В начале работы с БЕТА-практикумом на экране появляется картинка (рис. 7).

```
=БЕТА-ПРАКТИКУМ= МЕХМАТ МГУ, ЛВМ= 7 ОКТЯБРЯ 1985= ФЛАГ ОПУЩЕН
= X X XX XXX XX = РЯД СВОБОДЕН
= XX X X XX = Ч= ОЧ=
= XX X X XX = III= ОШ=
= XX X XX X X XX XX = В= ОВ=
= XX X XX XX XX = П= ОП=
= XX X XX XX X XX = S= OS=
===== РЕДАКТИРОВАНИЕ
*** БЕТА-ПРАКТИКУМ *** МЕХМАТ МГУ, ЛВМ *** 01.08.85 ***
ПРОГРАММА
. ДАНО :
. ПОЛУЧИТЬ:
. _____
КОНЕЦ ПРОГРАММЫ
```

Рис. 7

Для ввода текста программы в основном используются описанные выше команды редактирования текстов системы «Микромир-85». Для вставки конструкций языка программирования (ЕСЛИ, ЦИКЛ и т.п.) надо нажать клавишу **СПЕЦ** и первую букву конструкции:

СПЕЦ И —	для вставки ИСПОЛНИТЕЛЬ—КОНЕЦ ИСПОЛНИТЕЛЯ
СПЕЦ П —	ПРЕДПИСАНИЕ—КОНЕЦ ПРЕДПИСАНИЯ
СПЕЦ Е —	ЕСЛИ—ТО—ИНАЧЕ—КОНЕЦ ЕСЛИ.
СПЕЦ И —	ИНАЧЕ (ВНУТРИ ЕСЛИ)
СПЕЦ Ц —	ЦИКЛ — КОНЕЦ ЦИКЛА
СПЕЦ У —	УТВЕРЖДЕНИЕ
СПЕЦ В —	ВЫБОР—КОНЕЦ ВЫБОРА
СПЕЦ ? —	=> (вариант выбора)

Вставленные таким образом конструкции вместе с точками вертикальной разметки защищены от случайного изменения. Например, строку КОНЕЦ ЕСЛИ нельзя удалить, а сами слова КОНЕЦ ЕСЛИ нельзя изменить. Для удаления всей конструкции (вместе с вложенным в нее текстом) надо встать в ее первую строку и нажать **СПЕЦ**.

При перемещении курсора из измененной строки БЕТА-прак-

тикум проверяет ее и в случае обнаружения ошибки выводит сообщение в поле диагностики, расположеннное справа за «забором» из восклицательных знаков. Курсор при этом устанавливается в позицию около предполагаемого места ошибки. Вы, однако, не обязаны немедленно исправлять эту ошибку, а можете продолжить редактирование программы. Заметим, что при устранении причины ошибки диагностика об этой ошибке исчезает во всех строчках программы независимо от местоположения курсора.

В верхней части экрана изображается лабиринт с роботом и значения объектов встроенного вычислителя. Робот изображается одним из символов «⟨», «⟩», «A», «V» в зависимости от направления. Названия счетчиков сокращены до одной буквы (Ш — ШИРИНА, ОШ — ОБЩАЯ ШИРИНА). ПЛОЩАДЬ обозначена буквой S.

Так же как и в АЛЬФА-практикуме с помощью клавиш **ВЫПОЛНИТЬ** и **НОМЕ ВЫПОЛНИТЬ** можно запустить пошаговое выполнение. При пошаговом выполнении в поле диагностики визуализируются значения счетчиков (по предписанию ПОКАЗАТЬ ЗНАЧЕНИЕ) и результаты вычисления условий. В пошаговом режиме вызов каждого предписания выполняется как один неделимый шаг. Есть, однако, дополнительная возможность, позволяющая «зайти внутрь» предписания при его выполнении. Для этого, установив курсор на вызове предписания, надо нажать клавиши **СПЕЦ ВНИЗ**. Вы попадете внутрь предписания, после чего сможете выполнять его по шагам обычным образом. Клавиши **НОМЕ ВЫПОЛНИТЬ** или **СПЕЦ ВВЕРХ** приводят в этой ситуации к выполнению текущего предписания до конца.

При желании в БЕТА-практикуме можно редактировать лабиринт и положение робота в нем. Для перехода в редактирование лабиринта и обратно нужно нажать клавиши **СПЕЦ ВЫПОЛНИТЬ**. При редактировании лабиринта клавиша **X** используется для размещения препятствий, **ПРОБЕЛ** — для их удаления, клавиши **⟨**, **⟩**, **A**, **V** — для установки нового положения РОБОТА.

При завершении работы текст программы и лабиринт сохраняются на диске в файлах с именами .BET и .LAB. и при повторном запуске практикума считаются из этих файлов. Тем самым программа и лабиринт сохраняются от одного сеанса вашей работы до другого.

Редактирование—компиляция

Для перевода текста программы из того вида, в котором ее удобно читать человеку, в тот вид, в котором ее удобно выполнять ЭВМ, следует проделать некоторую работу (так называемую компиляцию). Этую работу можно проделать несколькими способами.

Первый из них — пакетная компиляция — характеризуется тем, что после ввода текста с помощью текстового редактора или с перфокарт вы выдаете команду для компиляции и ждете от нескольких минут до нескольких суток. По истечении этого срока вы получаете листинг и сообщения об ошибках вида

«LINE 135: TOKEN NOT EXPECTED».

После этого вы ищете и исправляете ошибку и повторяете компиляцию. После успешной компиляции вы выполняете так называемое редактирование связей, получая листинг распределения памяти, сообщения об ошибках вида

«COMPLEX ZERO DIVIDE»

и предупреждения вида

«*CO DOES NOT EXIST BUT HAS BEEN ADDED
TO DATA SET».**

После исправления этих ошибок вы опять должны компилировать вашу программу, т.е. все начинается сначала. Именно так (редактирование текста, компиляция, редактирование связей, выполнение) традиционно происходит работа с языками программирования Фортран и Паскаль.

Второй способ — интерпретация — отличается тем, что локальные ошибки внутри строки (пропущенные запятые) обычно выявляются при вводе текста, редактирование связей отсутствует вовсе, а обнаружение глобальных ошибок (незавершенные или неверно вложенные конструкции и т.п.) откладывается до выполнения. В таком режиме обычно используется язык Бэйсик.

Третий способ — редактирование-компиляция — отличается тем, что в процессе создания программы человеком она автоматически компилируется и выявленные ошибки сразу диагностируются. Программа, как и при использовании интерпретации, всегда готова к выполнению. Такая параллельная работа человека и компилятора возможна благодаря разнице в скоростях человека и машины: человек нажимает 1—2 клавиши в секунду, а ЭВМ за секунду выполняет сотни тысяч операций.

Примером редактора-компилятора является БЕТА-практикум.

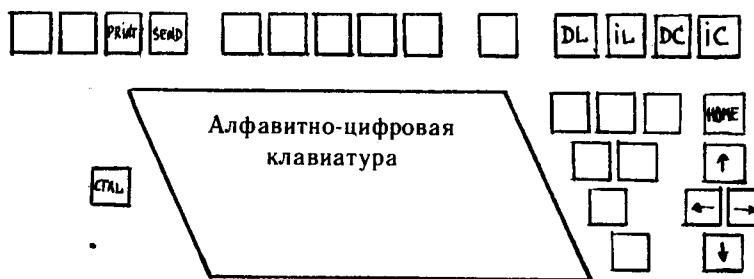


Рис. 8. Клавиатура дисплея VIDEOTON-340. Клавиши ВЫПОЛНИТЬ, СПЕЦ — слева вверху. На клавише ВЫПОЛНИТЬ написано PRINT, на клавише СПЕЦ — SEND. Клавиши DL, IL, DC, IC — справа вверху. Клавиша CTRL — слева

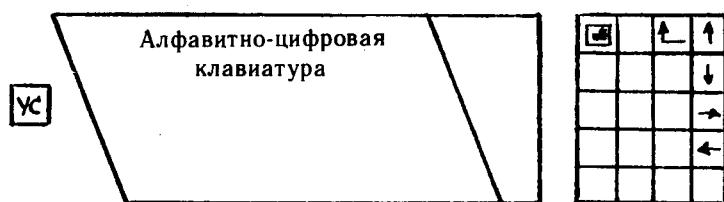


Рис. 9. Клавиатура дисплея VTA-2003. Функциональная клавиатура — справа. Клавиша СПЕЦ — левая в верхнем ряду (частично заштрихованный квадрат), HOME — вторая справа в верхнем ряду (изогнутая стрелка вверх). Клавиша CTRL — слева, на ней написано YC

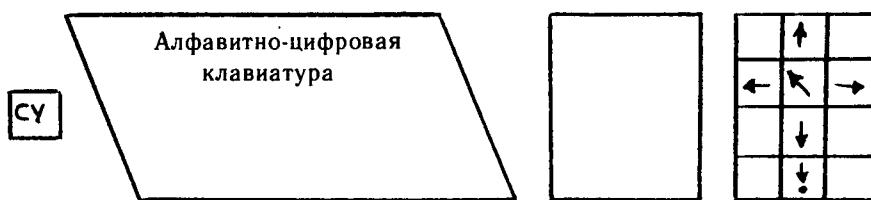


Рис. 10. Клавиатура дисплея 15ИЭ00-13 (Фрязино). Функциональная клавиатура справа. Клавиша СПЕЦ — средняя в нижнем ряду, HOME — средняя во втором ряду. Клавиша CTRL — слева, на ней написано CY

СОДЕРЖАНИЕ

Предисловие	3
ТЕМА 1. ОСНОВНЫЕ ПОНЯТИЯ ПРОГРАММИРОВАНИЯ	
Алгоритмы, исполнители, предписания	5
Объекты	11
Параметры	13
Задание для исполнителя ПУТНИК	14
Исполнитель СТЕКОВЫЙ КАЛЬКУЛЯТОР	15
Задание для исполнителя СТЕКОВЫЙ КАЛЬКУЛЯТОР	17
Исполнитель РЕДАКТОР СЛОВА	19
Задание № 1 для исполнителя РЕДАКТОР СЛОВА	20
Задание № 2 для исполнителя РЕДАКТОР СЛОВА	25
Задание № 3 для исполнителя РЕДАКТОР СЛОВА	25
Задание № 4 для исполнителя РЕДАКТОР СЛОВА	26
Исполнитель ЧЕРТЕЖНИК	26
Задание для исполнителя ЧЕРТЕЖНИК	28
ТЕМА 2. ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ СВЕРХУ—ВНИЗ	
Теоретический материал	30
Программа рисования слова	38
Задание по модификации программы рисования слова	40
Задание по рисованию простого орнамента	42
Задание по рисованию сложного орнамента	45
Дополнительное задание	45
ТЕМА 3. УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ	
Постановка вопроса	46
Повторение заданное число раз	46
Повторение по условию	47
Выбор из двух вариантов	49
Выбор из нескольких вариантов	52
Сложные условия	53
Построение отрицаний к сложным условиям	55
Вложенные конструкции	56
Задание № 1 по теме УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ	61
Задание № 2 по теме УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ	61
Исполнитель ПЕШЕХОД	63
Задание № 3 по теме УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ	64
Исполнитель РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ	66
Задание № 1 для исполнителя РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ	68
Задание № 2 для исполнителя РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ	71
Задание № 3 для исполнителя РОБОТ с ВСТРОЕННЫМ ВЫЧИСЛИТЕЛЕМ	73
Послесловие	77
ПРИЛОЖЕНИЕ. Практикумы по начальному курсу программирования. Д.В.Варсаноффев	
АЛЬФА-практикум	79
Микромир-85	83
БЕТА-практикум	84
Редактирование-компиляция	85