№ 45 (238) декабрь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Рожденная в XVIII веке

В 1999 году исполнилось 275 лет со времени основания Петербургской академии наук (1724 г.), ставшей в начале 1917 года Российской академией наук

Петербургская академия наук должна была по плану Петра I во многом отличаться от западноевропейских учреждений подобного рода. В то время как иностранные академии являлись в основном только местом, где подводились итоги научной работы, выполнявшейся в университетах, кабинетах и лабораториях, Петербургской академии предстояло стать главным "источником" науки в России. Академия начала свою работу после смерти Петра I на основе утвержденного им Положения [1].

Действительно, почти все, что было достигнуто в области науки в России в XVIII веке, исходило из Петербургской академии, сразу ставшей важным государственным органом. Уже в первые годы своего существования академия располагала превосходными для своего времени подсобными учреждениями: большим физическим кабинетом, химической лабораторией, астрономической обсерваторией, анатомическим театром, типографией и граверной палатой, механическими и оптическими мастерскими, библиотекой. Петербургские академики

Окончание читайте на с. 16

Современное здание Президиума РАН на Ленинском проспекте в Москве

Читайте в номере

Тематический выпуск

А.Г. Кушниренко, Г.В. Лебедев, Я.Н. Зайдельман

Информатика 7—9. Избранные главы нового учебника

Информатика изучает свойства информации, но строгого определения этого понятия не дает. Однако учитель должен уметь объяснить и что такое информация, и что такое компьютер, и как компьютер обрабатывает информацию, и даже что такое правильный алгоритм.

Транслятор?.. Это очень просто!

А.А. Дуванов

Окончание. Начало в № 41, 42, 44/99

Ну а теперь рассмотрим по-настоящему сложную задачу. Она тоже предлагалась в прошлом году на турнире юных программистов, и, представьте, нашлись дети, которые ее решили. Это меня сильно огорчило. Задача была специально предложена для того, чтобы ее никто не решил. Думал, будет повод нравоучению: "Вот вам и простой Корректор! Слабо! Дело, ребята, не в среде программирования, а в методах, приемах и опыте".

Меня оконфузили ребята из волгоградской команды Владимира Анатольевича Петрова. Обычный максимальный балл в турнирных задачах — 10. Для этой задачи я смело поставил 15. И ребята его заработали!

ЗАДАЧА (15 БАЛЛОВ)

Введем следующее определение:

Определение

```
<выражение>::=<число>|                (1)
  <выражение> – <выражение>|          (2)
  <выражение> + <выражение>|          (3)      (*)
  (<выражение>)                       (4)
<число>::=1|2|3|4|5|6|7|8|9           (5)
```

Задание. Напишите программу, которая проверяет правильность записи выражения. Если запись правильная, то нужно за ней на ленте записать результат вычисления выражения, в противном случае — сообщение "ош", и установить окошко на первый неверный символ или на место обнаружения ошибки в случае ошибок со скобками.

В начальный момент окошко расположено перед записью.

Замечание 1. Значение выражения должно быть записано на ленте в виде обычного целого десятичного числа.

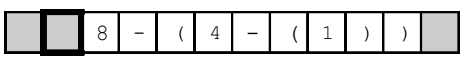
Замечание 2. Предполагается, что результат вычисления выражения и все промежуточные результаты не превышают числа, равного длине алфавита Корректора.

Замечание 3. Считается, что конечный и все промежуточные результаты вычислений неотрицательные.

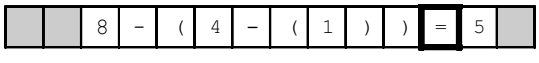
Примеры работы программы:

1

Начальное состояние

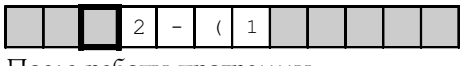


После работы программы

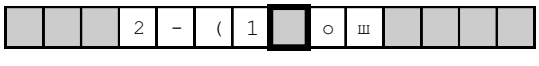


2

Начальное состояние



После работы программы



* * *

Сложность определению выражения, приведенному в условии задачи, придает наличие безобидной на первый взгляд строки с круглыми скобками.

До сих пор рекурсивность определений была простой и достаточно очевидной.

В самом деле, из записи

```
<число>::=<цифра>|<цифра> <число>
<цифра>::=0|1
```

сразу видно, что число — это любая последовательность из нулей и единиц.

Наличие строки (<выражение>) в определении (*) позволяет считать выражениями записи типа:

$$(1 + (7 - (9 + 8)))$$

$$7 + (8) - (1 - 7) + 6$$

— и так далее.

Иными словами, определение (*) почти соответствует нашему интуитивному представлению о том, какими могут быть обычные записи с круглыми скобками. Почему почти? Ну, например, запись 5—(—3) выражением в смысле определения не будет, в то время как в обычной жизни это совершенно нормальная запись, значение которой равно 8.

Понять и почувствовать это не так-то и просто для новичка.

Докажем, например, что запись 7 + (6 — (5)) является выражением в смысле определения (*).

Из (2) следует, что запись 7 + (6 — (5)) будет выражением, если выражением является запись (6 — (5)).

Из (4) следует, что запись (6 — (5)) будет выражением, если выражением является запись 6 — (5).

Из (3) следует, что запись 6 — (5) будет выражением, если выражением является запись (5).

Из (4) следует, что запись (5) будет выражением, если выражением является запись 5.

Но запись 5 — это выражение из (1), ведь 5 — это число из (5).

Поработать с рекурсивными определениями очень полезно. Эта область информатики находится на границе математики и лингвистики.

Еще несколько иллюстраций.

Примеры выражений:

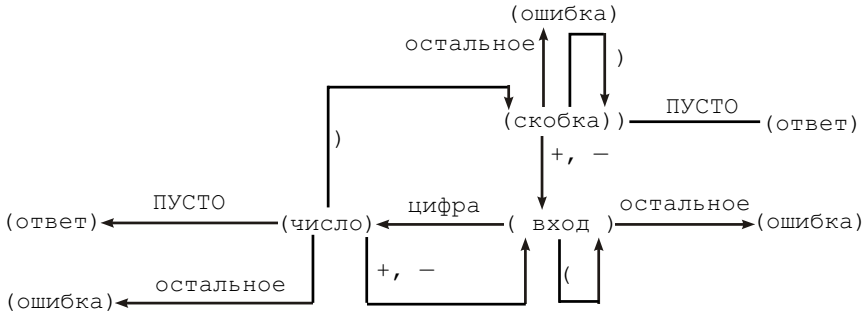
1) (2)
2) ((2))
3) (2 + 3)
4) (2 + 3 — 7)
5) (2+3) — (7+6)
6) ((2 + 3) — (7 + 6))
7) (2) — (7)
8) ((2) — (7))
9) ((2 + 3) — (7 + 6)) + 5

Примеры записей с ошибками:
1) ()
2) (2 (+3))
3) (2 + 3))
4))2(
5) (2 + 3
6) 1 + (—5)

Докажем, что запись **6** выражением не является. Она была бы выражением по (*), если бы выражением была запись (—5). А запись (—5) была бы выражением, если бы выражением была запись —5. Но последняя запись выражением не является, ибо не подходит ни под одну часть определения (*).

Последний пример расходится с нашим представлением о правильных выражениях со скобками. Это происходит потому, что число со знаком в обычном понимании не является числом в смысле определения (*). По определению (*) число — это цифра и знака не содержит.

Лексический анализатор основан на диаграмме переходов из одного состояния в другие плюс дополнительный подсчет баланса круглых скобок (на схеме он не отражен):



Для отслеживания баланса круглых скобок можно использовать ящик — Корректора (изначально пустой). На каждой открывающейся скобке нужно применить к ящику операцию ПЛЮС, а на каждой закрывающейся — МИНУС. Если после просмотра всего выражения ящик не будет пуст, это ошибка — баланс скобок в записи нарушен.

Итак, диаграмма состояний будет "следить" за правомочностью появления круглой скобки в данном месте записи, а ящик — за балансом скобок во всей записи в целом.

Если, например, за числом расположена открывающаяся скобка (ошибка!), то диаграмма состояний этот случай предусматривает.

А вот распознать ошибку в записи "2 + 3)" диаграмма бессильна! Ведь закрывающаяся скобка за числом — ситуация вполне нормальная. Но эта ошибка "не пройдет" после анализа содержимого ящика.

Основываясь на высказанных выше идеях, записать программу анализатора очень легко, и поэтому здесь она не приводится.

Сейчас нас больше всего интересует не лексический анализ выражения, а его вычисление. Вот уж здесь действительно скачок сложности сразу на несколько порядков!

Первый вариант решения (который, кстати, и реализовали ребята Владимира Анатольевича) выглядит так. Нужно искать в записи пару круглых скобок (...), таких, которые внутри себя скобок уже не содержат. Далее нужно вычислить выражение внутри этих скобок и вставить результат на ленту, заменив им всю конструкцию (...). Такие действия нужно повторять до тех пор, пока в исходном выражении скобок уже не останется.

Алгоритмы, которые просматривают запись много раз, называются многопроходными. Описанный выше алгоритм ОЧЕНЬ многопроходной, а значит, не оптимальный.

В "большой" информатике используют другой метод:
— переводят исходное выражение в обратную польскую запись;
— вычисляют польскую запись на стеке.

Этот алгоритм популярен еще по одной существенной причине: выражение в польской (бесскобочной) записи можно быстро вычислить. Это особенно актуально, если выражение содержит переменные (параметры).

ЗАПИСЬ ВЫРАЖЕНИЯ ПО-ПОЛЬСКИ И СТЕКОВЫЕ ВЫЧИСЛЕНИЯ

Самой первой темой в нашем университетском курсе "Азы программирования" обозначен блок, связанный с исполнителем Плюсик.

(В этом месте моя собака Мики, следящая за тем, что пишется на экране, удивленно посмотрела на меня и покачала головой: "Ничего себе АЗЫ! Для пятиклассников!")

Помнится мне, что кто-то из руководителей команд сомневался в нужности Плюсика для этих самых азов программирования. Мол, Плюсик с "настоящим" программированием никак не связан!

Однако именно в настоящем трансляторе без Плюсика никак не обойтись.

Плюсик — это модель стекового калькулятора. Среда исполнителя состоит из стека и вычислителя:

Стек служит для хранения чисел. Числа помещаются в стек по одному, подобно тому, как кольца детской пирамидки нанизываются на стержень. За один раз взять из стека можно только одно число с вершины числовой пирамиды. Копия числа в стеке не сохраняется, зато становится доступным следующее число, расположенное "ниже" взятого.

Система команд Плюсика

ЗАПОМНИ число — Число, записанное в команде, отправляется на хранение в стек.

Все эти команды работают одинаково: из стека последовательно извлекаются два числа и отправляются в вычислитель для выполнения соответствующей арифметической операции. Результат помещается в стек. Второе число из извлеченной из стека пары вычислитель считает первым операндом операции. Ситуация отказа возникает, когда в стеке меньше двух чисел, результат вычислений — отрицательное число или выполняется деление на 0.

СЛОЖИ
ВЫЧТИ
УМНОЖЬ
ДЕЛИ

Команда ДЕЛИ выполняет деление нацело.

ОЧИСТИ — Выполнение этой команды приводит к автоматическому удалению всех чисел из стека.

Сосчитать выражение типа 8 — (2 + 3) задача не из легких! Практически все трансляторы для вычисления таких выражений сначала переводят его в обратную польскую запись, а уж потом, подобно Плюсику, вычисляют на стеке.

Прошу прощения! Я до сих пор еще не объяснил устройство обратной польской записи арифметических выражений!

В обычной записи знак операции располагается между операндами, а в обратной польской — после. Вот и все!

Обычная запись операции: $a + b$

Обратная польская запись: $ab +$

Обычная запись выражения: $a + b \cdot (c + d/e)/f$

Обратная польская запись того же выражения:

$$abcde/+ \cdot f/+$$

Запись типа $+ ab$ выражения $a + b$ называется польской потому, что впервые была введена польским философом (!) Лукашевичем в связи с формулами символической логики. Соответственно запись типа $ab +$ для суммы двух чисел называется обратной польской записью.

Иными словами, прямая польская запись базируется на структуре:

```
<операция> <операнд> <операнд>
```

А обратная польская запись (которая нас интересует) — на структуре:

```
<операнд> <операнд> <операция>
```

В некоторых источниках обратная польская запись называется постфиксной, а прямая — префиксной. Этимология таких названий совершенно прозрачна.

У.Маккиман, Дж. Хорнинг, Д.Уортман в книге "Генератор компиляторов" называют обычную запись выражений инфиксной записью, а польскую — суффиксной.

Сравните:

— обычная запись выражения:

$$2 + 5 \cdot (1 + 3/2)/7;$$

— обратная польская запись выражения:

$$2\ 5\ 1\ 3\ 2\ /\ +\ \cdot\ 7\ /\ +;$$

— программа для Плюсика:

```
ЗАПОМНИ 2
ЗАПОМНИ 5
ЗАПОМНИ 1
ЗАПОМНИ 3
ЗАПОМНИ 2
ДЕЛИ
СЛОЖИ
УМНОЖЬ
ЗАПОМНИ 7
ДЕЛИ
СЛОЖИ
```

Мы видим, что программа для Плюсика практически является другой формой обратной польской записи выражения. Верно и обратное: обратная польская запись является программой для вычисления выражения на стеке. Исполнитель, который выполняет эту программу, должен работать совершенно так же, как Плюсик. Такие исполнители содержатся практически в любом трансляторе.

Одним словом, вычислить выражение в обратной польской записи совсем не трудно. Нужно построить Плюсик. И этот исполнитель предельно прост. Выражение типа

```
2 5 1 3 2 / + * 7 / +
```

он должен рассматривать как запись программы работы со стеком и вычислителем.

Алгоритм работы нового Плюсика должен быть таким:

```
В начальный момент стек пуст.
Продолжать, пока запись-программа не закончится:
Если очередной элемент записи — число, поместить его в стек и перейти к следующему элементу записи.
Если очередной элемент записи — операция, выполнить ее следующим образом:
извлечь из стека два числа и выполнить над ними операцию, считая при этом последнее извлеченное число первым операндом;
результат операции поместить в стек.
После завершения программы в стеке останется одно число — результат вычислений.
```

Пошаговое выполнение исполнителем Плюсик+ программы `2 5 1 3 2 / + * 7 / +`

Выполнение программы	Стек
2 5 1 3 2 / + * 7 / + ^	стек пуст, исполнитель — перед записью
2 5 1 3 2 / + * 7 / + ^	2
2 5 1 3 2 / + * 7 / + ^	2 5
2 5 1 3 2 / + * 7 / + ^	2 5 1
2 5 1 3 2 / + * 7 / + ^	2 5 1 3
2 5 1 3 2 / + * 7 / + ^	2 5 1 3 2
2 5 1 3 2 / + * 7 / + ^	2 5 1 1*

* Напомним, что деление выполняется нацело.

Продолжение таблицы со стр. 4

Выполнение программы	Стек
2 5 1 3 2 / + * 7 / +	2 5 2
2 5 1 3 2 / ^ + * 7 / +	2 10
2 5 1 3 2 / + ^ * 7 / +	2 10 7
2 5 1 3 2 / + * ^ 7 / +	2 1
2 5 1 3 2 / + * 7 ^ / +	3
2 5 1 3 2 / + * 7 / ^ + ^	3

Итак, если выражение записано "по-польски", вычислить его совсем не трудно. Даже Корректором.

А вот как перевести обычную запись в польскую?

В методичке по Плюсику написаны такие рекомендации для построения программ (читай — для перевода обычной записи в польскую):

Для преобразования обычного выражения в программу для Плюсика можно просматривать выражение последовательно слева направо, записывая команды ЗАПОМНИ число

Если после очередной записи команды ЗАПОМНИ выяснится, что над последними двумя числами в стеке можно выполнить арифметическую операцию, то записывается соответствующая команда.

Очень хорошая рекомендация! Но для человека. Как запрограммировать:

"Если после очередной записи команды ЗАПОМНИ выяснится, что над последними двумя числами в стеке можно выполнить арифметическую операцию, то записывается соответствующая команда".

Как проверить условие "можно выполнить арифметическую операцию"?

Непросто это! Особенно мешают круглые скобки. Тем более что вложение их друг в друга носит рекурсивный характер.

Посмотрите на выражение: 7 — (1 + 3).

Как перевести его в польскую запись?

Сложность в том, что исполнитель не видит запись целиком! В каждый момент ему виден только один символ (который в нашей задаче одновременно является и лексемой).

Следовательно, нужно придумать, как, перемещаясь от символа к символу, сформировать обратную польскую запись выражения, не видя его целиком.

Вы правы, это невозможно без хитрости! Нужно дополнительное средство, которое расширит "кругозор" исполнителя за пределы одного символа и позволит осуществить отложенные действия.

Конечно, первый символ записи "7" можно сразу поместить в результирующую строку. Следующий символ "—" — уже нельзя. Его нужно где-то временно запомнить, чтобы потом в нужный момент вставить в польскую запись.

Когда же наступит подходящий момент для знака "—"? Тогда, когда за первым операндом "7" будет переведен в польскую запись второй операнд операции "—". В нашем случае таким операндом будет польская запись выражения (1 + 3).

```
входная строка: 7—(1+3)
польский выход:
хранилище:

входная строка: —(1+3)    Число помещаем на
польский выход: 7         выход
хранилище:

входная строка: (1+3)     Операцию помещаем в
польский выход: 7         хранилище
хранилище:      —
```

Теперь на входе символ "(". Что с ним делать? Придется тоже его запомнить и хранить до тех пор, пока не встретится парный знак ")". После этого знак "(" из хранилища можно убрать.

```
входная строка: 1+3)      Открывающуюся скобку
польский выход: 7         помещаем в хранилище
хранилище: —(
```

Объект "1" помещаем в выходную польскую запись, а знак "+" запоминаем до лучших времен:

```
входная строка: + 3)      Число помещаем на
польский выход: 7 1       выход
хранилище:      —(

входная строка: 3)        Операцию помещаем в
польский выход: 7         хранилище
хранилище:      — ( +
```

Продолжая действовать в том же духе, получим:

```
входная строка: )         Число помещаем на
польский выход: 7 1 3     выход
хранилище:      —( +

входная строка: )         Операцию из хранилища
польский выход: 7 1 3 +   помещаем на выход
хранилище:      —(

входная строка:           Закрывающуюся скобку
польский выход: 7 1 3 +   удаляем из хранилища
хранилище: —

входная строка:            Операцию из хранилища
польский выход: 7 1 3 + —  помещаем на выход
хранилище:
```

Наши действия над "хранилищем" наглядно показывают, что эта память является стеком.

Второй вывод, который можно сделать из наших опытов, позволит написать формальный алгоритм преобразования обычного выражения в польскую запись. Вывод такой: символы перемещались из входной строки в выходную или в стек, а из стека в выходную строку в зависимости от их приоритетов.

Понятие приоритета можно формализовать, если ввести числовые характеристики, соответствующие нашим опытам.

Объект	Приоритет в стеке	Приоритет во входной строке
+, −	2	1
число	4	3
(	0	5
)		0

Замечание 1. Скобку ")" мы никогда не помещаем в стек.

Замечание 2. Приоритет открывающей скобки в стеке равен приоритету закрывающей скобки во входной строке, для того чтобы по формальному признаку равенства приоритетов открывающуюся скобку из стека можно было убрать тогда, когда парная к ней закрывающаяся появится во входной строке.

Теперь алгоритм преобразования можно записать следующим образом:

```
Поместить символ "(" в стек, а символ ")" – в конец входного выражения.
Пока входное выражение не будет исчерпано, повторять:
    Если на входе число, добавить его к выходу
    и перейти к следующему символу на входе
    Иначе, если вход > стека, поместить символ
    со входа в стек и перейти к следующему символу на входе
    Иначе, если вход < стека*, поместить символ
    со стека на выход
    Иначе, если вход = стеку, удалить символ
    со стека и перейти к следующему символу на входе
```

Инициализация перед входом в цикл введена для единообразия действий внутри цикла (когда не надо обрабатывать отдельно пустой стек и пустую входную строку).

Заметим также, что без первой ветви переключателя в цикле вполне можно обойтись (правда, ценой лишних действий исполнителя алгоритма): число в выходную строку может передаваться не сразу, а единообразно через стек:

```
Поместить символ "(" в стек, а символ ")" – в конец входного выражения.
Пока входное выражение не будет исчерпано, повторять:
    Если вход > стека, поместить символ со входа
    в стек и перейти к следующему символу на входе
    Иначе, если вход < стека, поместить символ
    со стека на выход
    Иначе, если вход = стеку, удалить символ
    со стека и перейти к следующему символу на входе
```

"Прокрутим" по этому алгоритму еще раз выражение 7 — (1 + 3).

```
вход: 7-(1+3))      вход=3 > стек=0
выход:              Помещаем число в стек.
стек: (

вход: -(1+3))       вход=1 < стек=4
выход:              Помещаем число на выход.
стек: (7

вход: -(1+3))       вход=1 > стек=0
выход: 7            Помещаем операцию в стек.
стек: (

вход: (1+3))        вход=5 > стек=2
выход: 7            Помещаем скобку в стек.
стек: (-

вход: 1+3))         вход=3 > стек=0
выход: 7            Помещаем число в стек.
стек: (-(

вход: +3))          вход=1 < стек=4
выход: 7            Помещаем число на выход.
стек: (-( 1

вход: +3))          вход=1 > стек=0
выход: 7 1          Помещаем операцию в стек.
стек: (-(

вход: 3))           вход=3 > стек=2
выход: 7 1          Помещаем число в стек.
стек: (-(+

вход: ))            вход=0 < стек=4
выход: 7 1          Помещаем число на выход.
стек: (-(+ 3

вход: ))            вход=0 < стек=2
выход: 7 1 3        Помещаем операцию на выход.
стек: (-(+

вход: ))            вход=0 < стек=0
выход: 7 1 3 +      Удаляем скобку из стека.
стек: (-(

вход: )             вход=0 < стек=2
выход: 7 1 3 +      Помещаем операцию на выход.
стек: (-

вход: )             вход=0 < стек=0
выход: 7 1 3 + −    Помещаем операцию на выход.
стек: (             Удаляем скобку из стека.

вход:               вход=0 < стек=0
выход: 7 1 3 + −    Помещаем операцию на выход.
стек:               Удаляем скобку из стека.
```

Заметим, что если бы, кроме операций "+" и "—", в записи присутствовали более "старшие" операции "*" и "/" , то алгоритм преобразования остался бы прежним, а вот таблицу приоритетов пришлось бы изменить.

* Сравниваются, разумеется, приоритеты.

Объект	Приоритет в стеке	Приоритет во входной строке
+, −	2	1
*, /	4	3
число	6	5
(	0	7
)		0

Пожалеем газетное место и не будем приводить программу этого транслятора для Корректора! Код получается не слишком сложный, но длинный.

Основные идеи рассмотрены, и теперь легко написать транслятор на Си или Паскале. Пусть Корректор займется любимым делом — роботландским чаем с сухариками!

АЗЫ ПРОГРАММИРОВАНИЯ В РОБОТЛАНДСКОМ УНИВЕРСИТЕТЕ

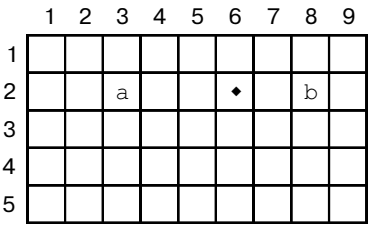
Наверное, вы решили, что "Азы программирования" — это небольшой курс по трансляторам. Это не так. Трансляторы — это, конечно, самая сложная тема курса и, вероятно, самая интересная с точки зрения практического введения в информатику. Другая любимая тема Азов — построение необычных геометрий в средах Кукарачи и Корректора. Ребятам эта тема часто нравится даже больше трансляторов.

Вот пример задачи (все геометрические определения даются отдельным текстом и здесь не приводятся в расчете на интуицию читателя) из репертуара Кукарачи:

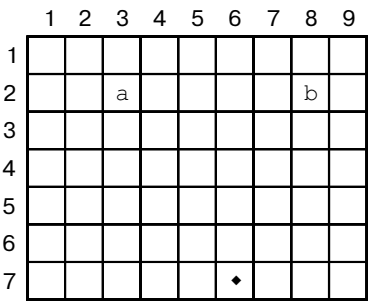
ЗАДАЧА (8 БАЛЛОВ)

Исполнитель расположен где-то внутри горизонтального отрезка *ab*. Сместить его вниз из исходного положения на *Pab*. Известно, что столбец точки *a* больше 2.

Пример начальной установки среды:



Среда после выполнения программы:



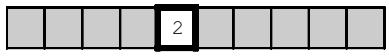
А вот два примера из "геометрии" Корректора.

ЗАДАЧА (3 БАЛЛА)

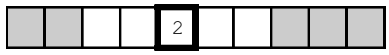
Окно установлено в центр "круга" (кругом назовем множество ячеек, удаленных от центра на расстояние меньшее или равное *r*). Закрасить пробелом все точки круга (кроме центра), если радиус круга помещен в клетку-центр в виде символа из алфавита Корректора. При этом числовое значение радиуса равно порядковому номеру символа в алфавите исполнителя.

Пример работы программы:

Начальное состояние



После работы программы



ЗАДАЧА (8 БАЛЛОВ)

На ленте Корректора заданы два круга с центрами в разных точках *a* и *b*. Известно, что $a < b$ и радиусы кругов меньше 50.

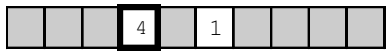
В клетках, изображающих центры кругов, записаны символы, задающие радиусы (как в условии предыдущей задачи).

Написать программу, которая закрасит пробелами все клетки пересечения кругов (кроме точек *a* и *b*, даже если они попадают в пересечение).

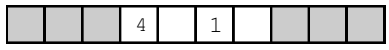
В начальный момент окошко установлено в точке *a*.

Пример работы программы:

Начальное состояние



После работы программы



Вы уже поняли, для кого наши Азы? Они для ребят, которые по-настоящему увлечены программированием, и для учителей, которые склонны поддерживать и развивать этот интерес.

Написать на Бейсике несколько строчек для вывода на экран какой-нибудь бредятины — не слишком большое достижение для программирования.

Изучить язык Си — это тоже не достижение. Вы же понимаете, чем работа с мясорубкой отличается от кулинарии! И старые добрые Кукарача и Корректор наглядно это демонстрируют.

Если у вас есть группа ребят, которым подобные вещи интересны, то Роботландский университет может быть полезен.

Нам можно написать по адресу:
kurs@robotland.users.botik.ru
и заглянуть на сайт
http://www.botik.ru/~robot

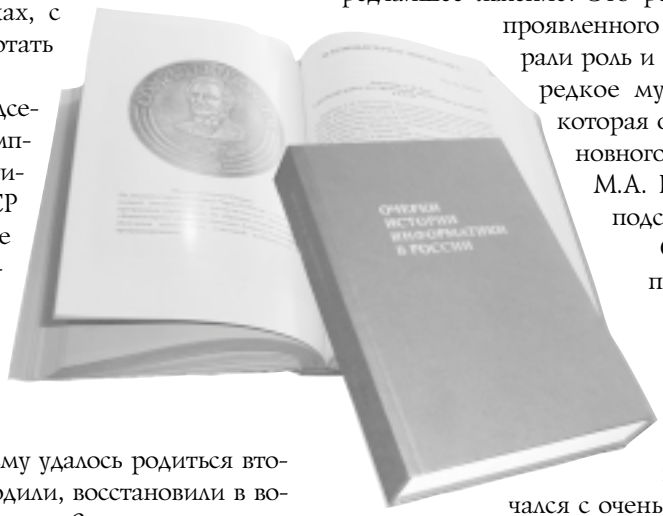
Эхо ГУЛАГа в Научном совете по кибернетике

Е.В. Маркова

Отец твердо и неуклонно верил в торжество справедливости, гуманности, нравственности.

О том, что кибернетика была в СССР репрессирована, знают все. А о том, что в ее реабилитации, в ее становлении и развитии принимали участие узники ГУЛАГа, недавно освободившиеся, — об этом знают не многие. Впрочем, о лагерной эпопее Тимофеева-Ресовского наслышаны все благодаря "Зубру" Д.Гранина.

Здесь я расскажу о нескольких судьбах — о тех кибернетиках-шестидесятниках, с которыми мне довелось работать в Совете.



Прежде всего о самом председателе Научного совета по комплексной проблеме "Кибернетика" при Президиуме АН СССР академике Акселе Ивановиче Берге (1893—1979). Он острил и на эту тему: "Мои предки совершили путь из варяг в греки, а я — из дворян в зеки!" Предками Акселя Ивановича были шведы, а он сам попал в зеки 21.12.37 г. Ему удалось родиться второй раз: 24.05.40 г. его освободили, восстановили в воинском звании инженера-флагмана 2 ранга и назначили преподавателем Военно-морской академии, профессором которой он был с 1935 г. В мае 1941 г. ему было присвоено воинское звание контр-адмирал-инженер. При этом родилась еще одна шутка Акселя Ивановича: "Из контрреволюционеров я попал в контр-адмиралы". Так за что же в тридцать седьмом доктор технических наук А.И. Берг попал в контрреволюционеры? И.Д. Морозов в сборнике воспоминаний о Берге, выпущенном к столетию со дня его рождения (Радиоэлектроника и связь, 1993, № 1, с. 16—22), выдвигает следующую версию. В 1937 г. Берг был начальником НИМИСТ — Научно-исследовательского морского института связи и телемеханики. Его обвинили во вредительстве (дело было групповое), Берг свою виновность и виновность подчиненных ему начальников отделов категорически отрицал. По проверке деятельности Берга была создана комиссия, подобранная в интересах следственных органов, она обвинила Берга в том, что по его вине государству был нанесен ущерб на сумму 154 млн руб. в виде неоправданных затрат на НИР и ОКР по созданию новой техники связи и спецтехники. Берг предъявил протест против состава комиссии. Была назначена новая комиссия, которую возглавил сам начальник НИМИСТ Я.Г. Вараксин. В комиссию вошли профессора М.А. Крупский и М.С. Бесчастнов. Вараксину удалось добиться разрешения встретиться с Бергом в тюрьме. Все, что Бергу ставили в обвинение, он опровергал со ссылкой по памяти на соответствующие документы. Комиссия пришла к заключению, что обвинения, выдвинутые против Берга, не имеют основания. Сам по себе этот факт — редчайшее явление! Это результат редкого мужества, проявленного заключенным Бергом; сыграли роль и его великолепная память, и редкое мужество членов комиссии, которая отважилась оправдать невиновного. Здесь надо еще учесть, что М.А. Крупский сам недавно был подследственным!

Сам Аксель Иванович о причине ареста и следствии в Совете не рассказывал (давал подписку о неразглашении). Но всякие "мелочи" вспоминал. За тысячу дней, которые он провел в тюрьмах, он встречался с очень интересными людьми, например, с К.К. Рокоссовским (будущим маршалом), А.Н. Туполевым (знаменитым конструктором самолетов), П.И. Лукирским (будущим академиком). Когда в камере собирались такие узники, они устраивали научные семинары. Каждый читал лекции из своей области знаний. Ну а Аксель Иванович еще дополнительно читал на память поэмы и стихи не только на русском языке, но и на немецком, французском, английском. Все, кто слушал лекции Берга по надежности в шестидесятых годах, помнят, какой эффект производил Берг на аудиторию, когда в научную лекцию вставлял строки из английской баллады о рыцаре, конь которого был плохо подкован, и о том, как по этой причине рухнуло царство. Но вернемся к тюрьме. В соседних камерах, как правило, оказывался кто-нибудь из моряков. Берг налаживал с ними связь с помощью азбуки Морзе и всячески старался поддержать бодрость духа у своих собратьев по несчастью. Через 35 лет после этих событий, в бытность Берга председателем Совета, на его имя по служебному адресу пришло письмо из Сибири. Мы, сотрудники Совета, были поражены,

© Е.В. Маркова, 1998.

ИНФОРМАТИКА 7—9

А.Г. Кушниренко,
Г.В. Лебедев,
Я.Н. Зайдельман

Избранные главы нового учебника

Выпуск 3

Выпуски 1, 2 опубликованы в № 39, 42/99

§ 12. Анализ и тестирование алгоритмов

12.1. Результаты труда программиста

Составляя алгоритмы, мы с вами фактически работали программистами. Подобную работу, только для других, намного более сложных задач выполняют профессиональные программисты.

А что является результатом труда программиста? Предположим, в результате работы программы Робот выполнил какую-то полезную работу. Можно ли считать, что эту работу сделал программист? Нет, ее выполнил Робот. Может быть, программист управлял Роботом? Нет, это делал компьютер. Получается, что, хотя без программиста работа была бы невозможна, непосредственного участия в ней он вроде бы не принимает.

Рассмотрим внимательно схему программного управления (*рис.* 2*). Видно, что единственный результат деятельности человека в этой схеме — это программа, по которой работает компьютер, точнее, даже не сама программа, а ***текст программы***, который компьютер обрабатывает и преобразует в исполняемые команды.

Итак, результат работы программиста — это текст написанной им программы или алгоритма.

12.2. Что такое правильный алгоритм

Если нам точно известна начальная ситуация, в которой будет выполняться тот или иной алгоритм (см., например, задачи 6 и 9 из § 7), то составить его нетрудно. Достаточно просто записать последовательность необходимых команд и оформить ее в соответствии с правилами языка. В полученном алгоритме не будет команд-вопросов — они не нужны, ведь ситуация известна полностью, следовательно, не будет в нем ни циклов **пока**, ни ветвлений. Такие алгоритмы называют ***линейными***.

Проверить правильность линейного алгоритма очень легко — достаточно выполнить одну за другой все входящие в него команды, чтобы убедиться в достижении необходимого результата или же найти ошибку.

Однако значительно чаще бывает так, что задано только общее описание начальных условий, а точная ситуация неизвестна. С такими задачами мы встречались в § 10—11. В таких задачах правильно составленный алгоритм должен работать в любой ситуации, соответствующей условию.

Как проверить такой алгоритм? Конечно, для конкретной ситуации его можно исполнить и убедиться, что он работает (или не работает, тогда нужно искать ошибку). Но ведь разных ситуаций может быть очень много, перебрать их все просто невозможно. Даже в простейшей задаче о движении Робота до стены (§ 10) количество ситуаций теоретически бесконечно — ведь расстояние до стены может быть любым.

Это может быть не только с алгоритмами. Невозможно, например, перебрать все существующие треугольники, но, доказав, что сумма углов треугольника всегда равна 180 градусам, мы можем использовать это свойство для любого треугольника без дополнительной проверки. Этот прием очень часто применяется в математике: если какое-то свойство удается *доказать*, то потом его можно использовать без дополнительных проверок.

Точно так же можно поступать и с алгоритмами. Если доказать, что алгоритм правильный, его можно не проверять отдельно для каждой ситуации.

А что такое правильный алгоритм? Интуитивно мы понимаем это, но для доказательства нужны четкие формулировки.

Правильность алгоритма означает, что для любой допустимой (соответствующей условию **дано**) начальной ситуации:

1) при выполнении алгоритма не может возникнуть отказ;
2) при выполнении алгоритма не может возникнуть зацикливание;
3) после завершения алгоритма будет достигнут правильный (соответствующий условию **надо**) результат.

Существуют специальные методы ***доказательства правильности алгоритмов***, которые позволяют строго доказывать справедливость данных утверждений для конкретных алгоритмов. К сожалению, эти методы достаточно сложны и требуют серьезных математических знаний, выходящих за рамки школьного курса, поэтому мы будем вместо строгих доказательств проводить ***рассуждения***, подтверждающие правильность алгоритмов.

12.3. Пример рассуждения, подтверждающего правильность алгоритма

```
алг вправо до стены                                    (А33)
   дано | где-то справа от Робота
        | есть стена
   надо справа стена | Робот подошел к стене
нач
| нц пока справа свободно
|  | вправо
| кц
кон
```

Докажем правильность этого алгоритма (для краткости мы будем пользоваться словом "докажем", не оговаривая каждый раз, что доказательство не является математически строгим).

1) При выполнении этого алгоритма невозможен отказ. В самом деле, единственная команда в алгоритме, при выполнении которой возможны отказы, — это команда вправо. Команда эта стоит первой в цикле **пока** с условием справа свободно и по свойству цикла выполняется только при выполнении этого условия. Следова-

* См. № 39/99.

тельно, команда вправо выполняется только когда шаг вправо возможен, значит, отказа не произойдет.

2) При выполнении этого алгоритма невозможно зацикливание. По условию стена справа от Робота есть. При каждом выполнении цикла Робот делает один шаг вправо, в результате расстояние до стены уменьшается на 1. Когда-нибудь это расстояние станет равным 0 и цикл, а вместе с ним и алгоритм, завершится.
3) После завершения алгоритма условие в **надо** обязательно выполняется. Это следует из свойства цикла **пока**: после завершения цикла его условие не может быть верным.

12.4. Использование промежуточных утверждений для доказательства правильности алгоритмов

Для более сложных и длинных алгоритмов бывает трудно сразу доказать правильность алгоритма в целом. В этом случае часто помогают промежуточные утверждения. Алгоритм получается примерно такой:

```
алг имя
   дано условие0
   надо результат
нач
| серия1
| утв условие1
| серия2
| утв условие2
| ...
| серияN
| утв условиеN
| серияN+1
кон
```

Остается доказать, что если изначально было выполнено условие0, то после выполнения серии1 будет верно условие1, из условия1 после выполнения серии2 следует условие2 и т.д.

Пример. Робот находится на прямоугольном поле без внутренних стен. Необходимо закрасить клетку в правом верхнем углу и перевести Робота в левый верхний угол.

```
алг углы                                          (А34)
  дано | Робот на прямоугольном поле без
       | внутренних стен
  надо сверху стена и слева стена
       | Робот в левом верхнем углу
       | клетка в правом верхнем углу
       | закрашена
нач
| вверх до стены
| утв сверху стена
| вправо до стены
| утв сверху стена и справа стена
|    | Робот в правом верхнем углу
|    | закрасить
| утв сверху стена и справа стена
|     и клетка закрашена
|   влево до стены
кон
```

В полученном алгоритме несложно доказать правильность перехода от каждого утверждения к следующему.

12.5. Утверждения внутри цикла. Инвариант

В рассмотренном алгоритме каждое утверждение проверяется ровно один раз, поэтому достаточно доказать правильность фрагментов алгоритма, заключенных между утверждениями.

Иногда возникает необходимость разобраться в работе фрагмента, расположенного внутри цикла. Если при этом использовать утверждения, они будут проверяться многократно — при каждом выполнении тела цикла, поэтому здесь надо доказать, что утверждения остаются правильными при каждом выполнении цикла.

Утверждение, расположенное в теле цикла и остающееся правильным при каждом выполнении цикла, называется ***инвариантом*** цикла. Инварианты цикла часто используются для доказательства правильности циклических алгоритмов.

12.6. Пример алгоритма с инвариантом цикла

```
алг к правой стене с закрашиванием          (А35)
  дано  | где-то справа от Робота есть стена
  надо справа стена | Робот подошел к стене,
        | все клетки от начального положения
        | Робота (включая его)
        | до стены закрашены
нач
| закрасить
| нц пока справа свободно
|   утв клетка закрашена
|   | закрашены все клетки от Робота и левее
|   вправо
|   закрасить
| кц
кон
```

Докажем правильность инварианта цикла, записанного в команде **утв**. В инвариант входят и формальное утверждение, проверяемое компьютером (клетка закрашена), и дополнительный комментарий (закрашены все клетки от Робота и левее).

При первом выполнении цикла утверждение, очевидно, правильно. Клетка с Роботом закрашена, потому что перед циклом была выполнена соответствующая команда. Левее Робота интересующих нас клеток нет, поэтому можно сказать, что все клетки левее Робота (на самом деле не существующие) закрашены.

Докажем теперь, что если при каком-то выполнении цикла утверждение верно, то оно верно и при следующем выполнении. Для этого “развернем” цикл в линейную последовательность команд, то есть выпишем команды в том порядке, в котором они будут выполняться компьютером. Получим такой фрагмент:

```
утв клетка закрашена
   | закрашены все клетки от Робота и левее
  вправо
  закрасить
   | здесь заканчивается одно выполнение
   |тела цикла
   | и начинается следующее
утв клетка закрашена
   | закрашены все клетки от Робота и левее
```

Предположим, что первое утверждение верно. Тогда после выполнения команды вправо закрашены все клетки левее Робота, а после команды закрасить закрашена и клетка, где стоит Робот. Следовательно, второе утверждение верно.

Таким образом, мы доказали, что инвариант цикла справедлив при первом выполнении цикла и при каждом последующем.

Отметим важное отличие инварианта от условия цикла, которое записывается после **пока**. Справедливость условия в **пока** зависит от обстановки, в которой выполняется алгоритм, предсказать правильность этого условия заранее невозможно, а после окончания цикла оно обязательно будет нарушено. Инвариант же — это условие, которое обязательно будет выполнено, независимо от конкретной обстановки при выполнении. Особенно важно, что инвариант остается верным и после того, как выполнение цикла завершено.

12.7. Использование инварианта при создании алгоритмов

В предыдущем примере мы использовали инвариант для доказательства правильности уже написанного алгоритма. Иногда, особенно в сложных задачах, инвариант полезно применять на стадии написания алгоритма. Сформулировав и записав в виде инварианта условие, которое должно соблюдаться при каждом выполнении цикла, можно уменьшить возможность ошибки при записи тела и условия цикла.

Пример. Робот находится в коридоре. Снизу от него — непрерывная горизонтальная стена, которая тянется до конца коридора. Сверху — ряд горизонтальных стен различной длины, разделенных проходами разной ширины (*рис.* 41). Получить на экране Счетчика количество горизонтальных стен над коридором. (Например, на *рис.* 41 это количество равно 4.)

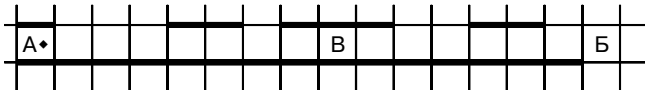


Рис. 41

Общая идея решения очевидна. Робот должен идти вправо по коридору, а Счетчик — увеличивать значение при прохождении каждой стены. Возникает вопрос: в какой момент следует учитывать очередную стену? Когда она начинается или когда заканчивается? Чтобы сделать правильный выбор, построим инвариант цикла. Если считать "начала" стен, то инвариант будет таким:

```
утв | число в Счетчике показывает количество
    | стен, пройденных Роботом, включая ту,
    | у которой он сейчас находится
```

Если же считать "окончания" стен, инвариант получится немного другим:

```
утв | число в Счетчике показывает количество
    | стен, полностью пройденных Роботом
```

Поясним разницу между этими условиями. Если мы будем следовать первому из них, то в клетке В (*рис.* 41) значение Счетчика должно быть равно 3, а если второму — то 2.

Чтобы выбрать в качестве инварианта более подходящее условие, вспомним, что инвариант должен быть верным и до, и после окончания цикла. Ясно, что, когда коридор пройден до конца, количество начатых и законченных стен одинаково, то есть верны оба условия. А вот до выполнения цикла, когда Робот стоит в клетке А, они оказываются неравноценными. Если следовать первому условию, то до начала цикла в Счетчике должен быть либо 0, либо 1, в зависимости от того, есть ли стена над начальной клеткой коридора. Второе условие более определенно: в начале в Счетчике обязательно должен быть ноль, так как ни одна стена еще не пройдена до конца.

Понятно, что второе условие удобнее, оно позволяет построить более простой и понятный алгоритм.

```
алг подсчет горизонтальных стен                (А36)
   дано | Робот в коридоре, снизу — сплошная
        | стена, сверху — горизонтальные стены,
        | разделенные проходами
   надо снизу свободно | коридор пройден
        | на экране Счетчика количество
        | стен сверху от Робота
нач
| сбросить
| нц пока снизу стена
|  утв  | число в Счетчике показывает
|       | количество стен, полностью
|       | пройденных Роботом
|  если сверху стена
|   то | проверить, не последняя ли это
|      | клетка стены
|      вправо
|      если сверху свободно
|       то | стена кончилась
|          | увеличить
|      все
|   иначе вправо
|  все
| кц
| показать
кон
```

12.8. Тестирование алгоритмов

Независимо от того, доказана ли правильность алгоритма, для реального использования на компьютере алгоритм надо проверить — ведь и в доказательстве могут быть ошибки.

Чтобы проверить алгоритм, его надо исполнить и сравнить полученный результат с тем, который требуется получить. При этом несовпадение результатов означает, что в алгоритме есть ошибки, а вот совпадение... Совпадение не значит ничего! Ведь алгоритм должен давать правильный результат для любой начальной ситуации, поэтому нельзя делать выводы после одной проверки.

Понятно, что реальных ситуаций может быть очень много и провести проверку каждой из них невозможно. Поэтому проверяют не все, а только некоторые специально подобранные ситуации, для каждой из которых *заранее известен правильный результат*. Каждая такая ситуация и сама проверка называется ***тестом***, а процесс проверки — ***тестированием***.

Подбор тестов для проверки — задача трудная. Необходимо проверить не только обычные, очевидные ситуации, но и так называемые ***крайние случаи***, в которых чаще всего можно ожидать проявления ошибок в алгоритме. Понятие крайнего случая поясним на нескольких примерах.

Пример 1. Где-то справа от Робота есть стена. Крайний случай — Робот уже у стены.

Пример 2. Робот где-то в коридоре. Крайние случаи — Робот в крайней левой клетке коридора; Робот в крайней правой клетке коридора; Робот в коридоре из одной клетки.

Пример 3. Некоторые клетки коридора закрашены. Крайние случаи — закрашенных клеток нет; закрашена ровно одна клетка; закрашены все клетки коридора.

Примеры решения задач

Пример 1. Дан фрагмент алгоритма:

```
нц пока справа свободно
| вправо
| если клетка закрашена
| | то влево
| | иначе вверх
| все
кц
```

Придумайте ситуации, в которых при выполнении этого фрагмента

а) произойдет отказ;
б) произойдет зацикливание;
в) компьютер не даст Роботу ни одной команды-приказа;
г) компьютер даст Роботу ровно одну команду-приказ;
д) компьютер даст Роботу ровно две команды-приказа.

Решение

а) Из всех команд данного фрагмента отказ возможен только при выполнении команд движения Робота. Этих команд во фрагменте три.

Команда вправо выполняется только при соблюдении условия цикла справа свободно, значит, при ее выполнении отказ невозможен.

Команда влево выполняется сразу после команды вправо. Если Робот смог пройти вправо, он сможет и вернуться обратно. Значит, и здесь отказ невозможен.

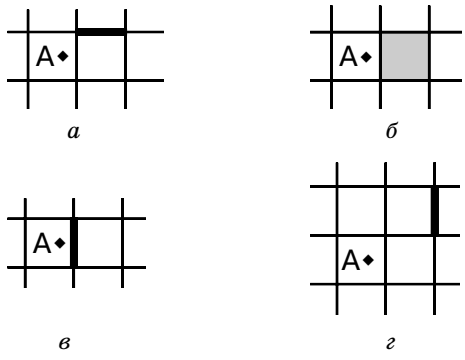
А вот команда вверх никак не защищена, при ее выполнении отказ действительно может случиться. Это произойдет, если, шагнув вправо, Робот попадет на чистую клетку, выше которой расположена стена (*рис.* 42*а*).

б) Одна из возможных причин зацикливания — отсутствие изменений в обстановке при выполнении цикла. В данном фрагменте такое возможно, если Робот будет делать шаг вправо, а затем, попав на закрашенную клетку, возвращаться влево (*рис.* 42*б*).

в) Робот не получит команд-приказов, если условие цикла с самого начала не будет выполнено, то есть в начальном положении справа от Робота будет находиться стена (*рис.* 42*в*).

г) Если условие цикла с самого начала не выполнено, Робот вообще не получит приказов. Если условие выполнено, Робот получит приказ вправо, а затем, в зависимости от того, закрашена ли клетка, в которую он попал, приказ вверх или вниз. Таким образом, ситуация, в которой Робот получит ровно один приказ, для данного фрагмента невозможна.

д) Приказов будет ровно два, если цикл будет выполнен один раз. Соответствующая ситуация изображена на *рис.* 42*г*.



а *б* *в* *г*

Рис. 42

Пример 2. Что можно сказать о правильности такого фрагмента:

```
нц пока клетка закрашена
| если справа свободно
| | то вправо; закрасить
| все
кц
```

Решение. На первый взгляд задание бессмысленно: что можно сказать о правильности фрагмента, если неизвестно, какую задачу он должен решать?

Действительно, если цель неизвестна, невозможно утверждать, что действия верны. Однако бывают ситуации, когда ошибку можно увидеть, даже не зная, для чего составлялся алгоритм.

Посмотрим внимательно на приведенный фрагмент. Если тело цикла будет исполнено хотя бы один раз, то обязательно произойдет зацикливание! В самом деле, если справа от Робота свободно, то он перейдет на новую клетку и закрасит ее, после чего условие цикла

клетка закрашена будет заведомо выполнено. Если же справа от Робота стена, он останется в той же клетке. Эта клетка закрашена, так как в ней выполнено условие цикла, значит, это условие останется справедливым и при следующей проверке.

Итак, данный цикл либо не будет выполнен ни разу, либо приведет к зацикливанию. Понятно, что в правильных алгоритмах такая ситуация встречаться не должна, так что фрагмент, вероятно, содержит ошибку.

Задачи и упражнения

1. Дан фрагмент алгоритма:

```
если клетка чистая
| то нц пока справа свободно
|     | закрасить; вправо
|   кц
| иначе
| влево
все
```

Придумайте ситуации, в которых при выполнении этого алгоритма
а) произойдет отказ;
б) произойдет зацикливание;
в) Робот не получит ни одного приказа;
г) Робот получит ровно два вопроса;
д) Робот получит нечетное число приказов.

2. Составьте набор тестов для проверки решения задач 7 и 9 из § 10.

3. Составьте набор тестов для проверки решения задач 4—6 из § 11.

4. Робот находится внутри прямоугольника, с четырех сторон огороженного стенами. Внутри прямоугольника стен нет. В левой вертикали закрашены некоторые клетки. Закрасить соответствующие им клетки в правой вертикали.

5. Робот находится внутри прямоугольника, с четырех сторон огороженного стенами. Внутри прямоугольника стен нет. Некоторые клетки в прямоугольнике закрашены. Закрасить горизонтальные ряды, в которых
а) закрашена левая клетка;
б) закрашены левая и правая клетки;
в) закрашены левая и правая клетки и еще хотя бы одна;
г) закрашены левая и правая клетки и еще ровно одна;
д) есть хотя бы одна закрашенная клетка;
е) есть ровно одна закрашенная клетка;
ж) есть хотя бы две закрашенные клетки;
з) есть ровно две закрашенные клетки;
и) четное число закрашенных клеток;
к) нечетное число закрашенных клеток.

6. Робот находится внутри прямоугольника, с четырех сторон огороженного стенами. Внутри прямоугольника стен нет. Одна клетка в прямоугольнике закрашена.
а) Составить алгоритм, приводящий Робота в закрашенную клетку.
б) Закрасить горизонталь и вертикаль, на пересечении которых находится эта клетка.

7. Робот находится внутри прямоугольника, с четырех сторон огороженного стенами. Внутри прямоугольника нет стен и закрашенных клеток. Закрасить часть клеток так, чтобы оставшаяся часть имела форму квадрата максимально возможных размеров (*рис.* 43).

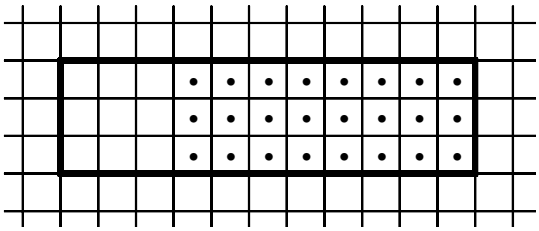


Рис. 43

8. Робот находится на поле без стен.
а) Где-то правее Робота и где-то выше Робота есть закрашенные клетки. Закрасить прямоугольник с вершинами в этих клетках (*рис.* 44).

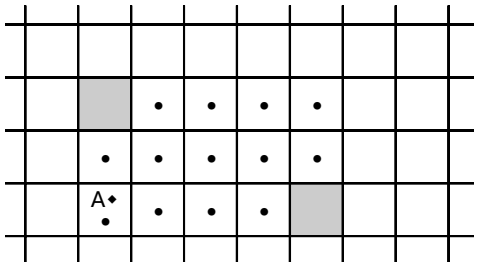


Рис. 44

б) Где-то правее Робота есть закрашенная клетка. Закрасить квадрат с вершинами в этой клетке и в клетке начального положения Робота (*рис.* 45).

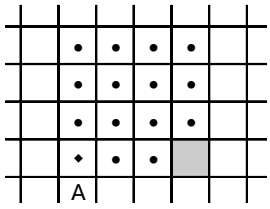


Рис. 45

9. Робот находится внутри прямоугольника, с четырех сторон огороженного стенами. Внутри прямоугольника стен нет. Закрасить клетки прямоугольника в шахматном порядке.

10. Робот находится внутри прямоугольника, с четырех сторон огороженного стенами. Внутри прямоугольника стен нет, некоторые клетки закрашены. Закрасить горизонтали и вертикали всех закрашенных клеток.

Глава 3. Алгоритмы и величины

§ 13. Исполнитель Чертежник и работа с ним

13.1. Особенности записи чисел в информатике

В информатике для отделения целой части числа от дробной используется точка, а не запятая, как в школьной математике. Это позволяет записывать несколько рядом стоящих чисел через запятую без риска вызвать путаницу. При задании точек плоскости координаты *x* и *y* в информатике разделяются запятой (*рис.* 46).

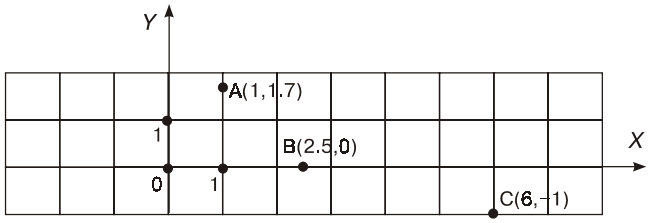


Рис. 46

13.2. Исполнитель Чертежник

Чертежник предназначен для построения рисунков, чертежей, графиков и т.д. на бесконечном листе бумаги. Чертежник имеет перо, которое можно поднимать, опускать и перемещать. При перемещении опущенного пера за ним остается след — отрезок от старого положения пера до нового. Всего Чертежник умеет выполнять 4 команды:

```
опустить перо
поднять перо
сместиться в точку (x,y)
сместиться на вектор (a,b)
```

По команде `опустить перо` Чертежник опускает перо. Если перо уже было опущено, никаких действий Чертежник не выполняет, но и отказа не происходит. Таким образом, после выполнения команды `опустить перо` перо оказывается опущенным (готовым к рисованию) независимо от его предыдущего положения.

Аналогично по команде `поднять перо` перо оказывается поднятым. Выполнение этой команды тоже не может привести к отказу.

Команды `сместиться в точку` и `сместиться на вектор` перемещают перо Чертежника. Если при этом перо опущено, на бумаге остается след. Таким образом, эти команды позволяют строить чертежи и рисунки.

13.3. Команды с параметрами

В отличие от Робота, который всегда смещался ровно на одну клетку, смещение Чертежника может быть произвольным. Поэтому для выполнения команд `сместиться в точку` и `сместиться на вектор` необходимо задать дополнительную информацию — указать, куда конкретно необходимо переместить перо Чертежника.

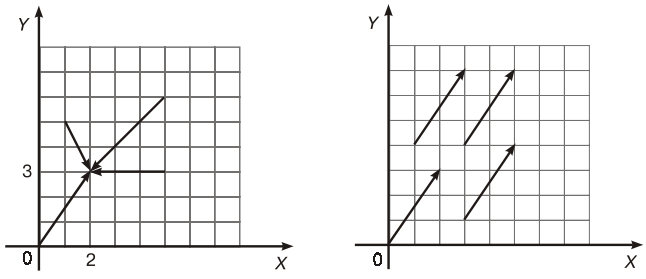
Эта дополнительная информация записывается в команде в виде ***аргументов*** — чисел, которые записываются в скобках после имени команды. Например: сместиться в точку (2, 3) или сместиться на вектор (1.4, 2.3).

Необходимость аргументов указывается в описании команды:

```
сместиться в точку (x,y)
сместиться на вектор (a,b)
```

13.4. Абсолютное и относительное смещение

В команде `сместиться в точку` в качестве аргументов указываются координаты той точки, в которую попадет перо после выполнения команды. На *рис.* 47*а* показаны результаты выполнения команды `сместиться в точку (2, 3)` при различных положениях пера до этой команды. Видно, что независимо от предыдущего положения перо оказывается в точке (2, 3), но длина и направление отрезка, который при этом чертится (конечно, если перо опущено), могут быть различны. Команду `сместиться в точку` называют командой ***абсолютного смещения***, так как в ней указываются ***абсолютные координаты*** пера.



а *б*

Рис. 47

Несколько иначе работает команда `сместиться на вектор`. Если перо Чертежника находится в точке `(x, y)`, то по команде `сместиться на вектор(a, b)` Чертежник сместит перо в точку с координатами `(x + a, y + b)`. Таким образом, координаты, указанные в команде, измеряются не от начала координат, а *относительно* текущего положения пера Чертежника. Поэтому команду `сместиться на вектор` называют командой ***относительного смещения***.

На *рис.* 47*б* показаны результаты выполнения команды `сместиться на вектор (2, 3)` при различных положениях пера до этой команды. Из рисунка видно, что положение пера после этой команды зависит от его предыдущего положения, но зато в результате получается отрезок, длина и направление которого постоянны. В математике такой отрезок называется ***вектором***, отсюда и происходит название команды.

Команды абсолютного смещения создают рисунок в строго определенном месте координатной плоскости. Они обычно используются, когда рисунок привязан к месту, например, при построении графиков функций.

Команды относительного смещения позволяют создавать рисунок в любом месте. Они используются для создания рисунков, у которых точное место не играет роли или которые нужно воспроизводить в различных местах.

13.5. Пример алгоритма управления Чертежником

Попробуем с помощью Чертежника нарисовать прямоугольник с двумя диагоналями (*рис.* 48*а*). Как это сделать? Можно начать с левого нижнего угла и, двигаясь против часовой стрелки, нарисовать все 4 стороны прямоугольника, не отрывая пера от бумаги, после чего нарисовать диагонали (*рис.* 48*б*).

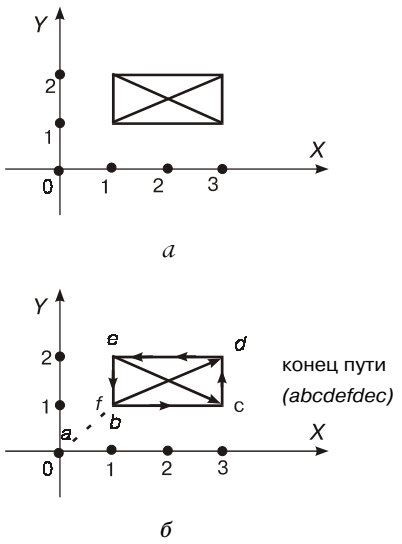


Рис. 48

Какие же команды и в какой последовательности нужно для этого дать Чертежнику? Пусть в начальный момент перо поднято. В левый нижний угол можно попасть, скомандовав

```
сместиться в точку (1,1)
```

Теперь надо нарисовать прямоугольник. Для этого прежде всего следует вызвать команду

```
опустить перо
```

После этого перо будет расположено в точке (1, 1) и опущено. Для рисования прямоугольника воспользуемся командой `сместиться в точку`:

```
сместиться в точку (3,1)
сместиться в точку (3,2)
сместиться в точку (1,2)
сместиться в точку (1,1)
```

После выполнения этих команд прямоугольник будет нарисован, а опущенное перо будет расположено снова в левом нижнем углу — в точке (1, 1).

Осталось нарисовать диагонали. Это можно сделать так:

```
сместиться в точку (3,2)
сместиться в точку (1,2)
сместиться в точку (3,1)
```

При этом, однако, верхняя сторона прямоугольника будет нарисована второй раз. Если мы хотим этого избежать, то перед командой

```
сместиться в точку (1,2)
```

нужно поднять перо, а потом его опустить:

```
сместиться в точку (3,2)
поднять перо
сместиться в точку (1,2)
опустить перо
сместиться в точку (3,1)
```

В этот момент вся картинка изображена и осталось поднять перо, чтобы лист бумаги можно было вынуть:

```
поднять перо
```

Запишем полученный алгоритм на алгоритмическом языке:

```
алг прямоугольник с диагоналями                    (A37)
  дано | перо поднято
  надо | нарисован прямоугольник
       | с диагоналями (рис.48а), перо поднято
нач
  | сместиться в точку (1,1)
  | опустить перо
  | сместиться в точку (3,1)
  | сместиться в точку (3,2)
  | сместиться в точку (1,2)
  | сместиться в точку (1,1)
  | утв | нарисован прямоугольник
  | сместиться в точку (3,2)
  | поднять перо
  | сместиться в точку (1,2)
  | опустить перо
  | сместиться в точку (3,1)
  | поднять перо
кон
```

13.6. Рисование букв

С помощью Чертежника можно рисовать любые фигуры, составленные из отрезков, например, буквы. Составим алгоритм, при выполнении которого Чертежник рисует на клетчатой бумаге букву "М" (*рис.* 49; размеры каждой клетки 1×1). Поскольку начальное положение пера на плоскости не задано, то придется воспользоваться командой `сместиться на вектор`:

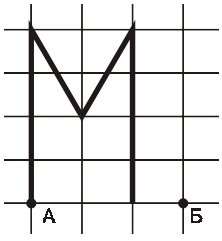


Рис. 49

```
алг буква М                                        (A38)
  дано | перо в точке А (рис. 49) и поднято
  надо | нарисована буква М (рис. 49),
       | перо в точке Б и поднято
```

```
нач
| опустить перо
| сместиться на вектор (0,  4)
| сместиться на вектор (1, -2)
| сместиться на вектор (1,  2)
| сместиться на вектор (0, -4)
| поднять перо
| сместиться на вектор (1,  0)
кон
```

Аналогично можно составить алгоритмы буква И, буква Р и буква У (*рис.* 50).

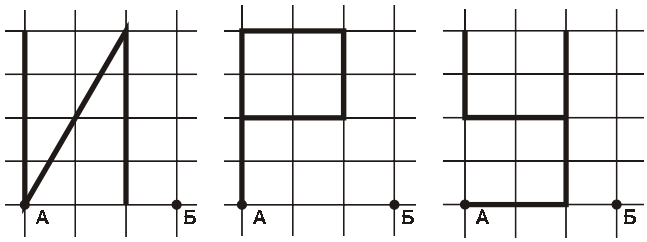


Рис. 50

13.7. Использование вспомогательных алгоритмов при управлении Чертежником

Составив алгоритмы для рисования букв, можно использовать их как вспомогательные и составлять алгоритмы рисования слов.

```
алг МИР                                          (А39)
  дано | перо поднято
  надо | нарисовано слово МИР, перо поднято
       | и расположено в конце слова
       | (начале следующей буквы)
нач
| буква М
| буква И
| буква Р
кон
```

Подобным образом можно строить с помощью Чертежника любые сложные изображения, используя вспомогательные алгоритмы для изображения стандартных деталей.

Разбивая изображение на части и составляя отдельные алгоритмы для каждой из них, важно точно учитывать положение пера до и после рисования каждого фрагмента. Например, при рисовании букв мы соблюдали следующее соглашение: перед началом рисования перо находится в левом нижнем углу буквы, после окончания — в левом нижнем углу следующей буквы слова. Подобные соглашения рекомендуется принимать при разработке любого проекта. Лучше всего заносить их в условия **дано** и **надо**.

Задачи и упражнения

1. Петя записал через запятую несколько вещественных и целых чисел, по привычке поставив десятичные запятые внутри чисел. Вот что у него получилось:
а) 3, 5, 7; б) 7, 3, 5, 0, 1.
Сколькими способами можно прочесть эти записи, если в **а** записано два числа, а количество чисел, записанных в **б**, неизвестно?

2. Нарисуйте результат выполнения следующего алгоритма:

```
алг домик                                        (А40)
  дано | перо поднято
  надо | нарисован домик, перо в исходном
       | положении и поднято
нач
| опустить перо
| сместиться на вектор (4, 0)
| сместиться на вектор (0, 4)
| сместиться на вектор (-4, 0)
| сместиться на вектор (0,-4)
| поднять перо
| сместиться на вектор (0, 4)
| опустить перо
| сместиться на вектор (2, 2)
| сместиться на вектор (2,-2)
| поднять перо
| сместиться на вектор (-4,-4)
кон
```

3. Измените алгоритм домик (А40) так, чтобы домик рисовался с окошком.

4. Дан основной алгоритм улица из трех домиков:

```
алг улица из трех домиков                        (А41)
нач
| домик; сместиться на вектор(6, 0)
| домик; сместиться на вектор(6, 0)
| домик
кон
```

Этот алгоритм использует вспомогательный алгоритм домик (А40). Нарисуйте результат выполнения алгоритма А41 (полученную картинку и положение пера Чертежника).

5. Составьте алгоритм рисования улицы из шести домиков.

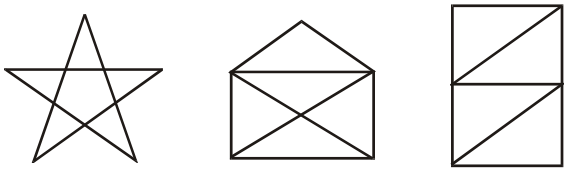
6. Петя зачеркнул последнюю команду сместиться на вектор(—4, —4) в алгоритме домик (А40). Как Коля должен изменить алгоритм улица из трех домиков (А41), чтобы рисовалась та же картинка, что и раньше?

7. Составьте алгоритм управления Чертежником, после выполнения которого будут нарисованы:
а) отрезок с концами в точках (1, 2) и (—1, 1);
б) квадрат со сторонами длины 4, параллельными координатным осям, так, чтобы левый нижний угол квадрата совпадал с начальным положением пера Чертежника;
в) квадрат со сторонами длины 6, параллельными координатным осям, так, чтобы левый нижний угол квадрата совпадал с начальным положением пера Чертежника;
г) какой-нибудь отрезок длины 3, проходящий через точку (2, 2);
д) какой-нибудь квадрат со сторонами длины 2 и центром в начале координат;

е) какой-нибудь прямоугольник с длинами сторон 3 и 4, содержащий внутри себя начало координат;
ж) какой-нибудь параллелограмм.

8. Составьте алгоритм управления Чертежником, после исполнения которого будут нарисованы:
а) инициалы полководца Кутузова;
б) ваши инициалы;
в) буква "Ф";
г) зеркальные отражения букв "И"и "Р" относительно горизонтальной оси;
д) число "12" римскими цифрами;
е) слово "МГУ";
ж) почтовый индекс 161110 (цифры индекса должны быть написаны, как на почтовых конвертах).

9. Составьте алгоритм для рисования фигур, изображенных на *рис.* 51, так, чтобы в процессе рисования перо не отрывалось от бумаги и ни одна линия не проводилась дважды.



звезда *раскрытый конверт* *шаблон цифры почтового индекса*

Рис. 51

10. Составьте десять алгоритмов для рисования десяти цифр почтового индекса так, чтобы при их последовательном вызове цифры рисовались друг за другом. Используя эти алгоритмы как вспомогательные, нарисуйте ваш почтовый индекс.

11. По образцу алгоритма "МИР" (A39) составьте алгоритмы:
а) РИМ;
б) МИМ.

12. Измените алгоритмы рисования букв "М", "И", "Р" так, чтобы при последовательном вызове этих алгоритмов слово "МИР" оказалось написанным:
а) с удвоенным расстоянием между буквами;
б) буквами удвоенного размера;
в) сверху вниз;
г) сверху вниз буквами удвоенного размера.

13. Дан алгоритм:

```
алг фигура                                   (A42)
 дано | перо в начале координат и поднято
нач
  сместиться в  точку  (2, 1)
  опустить перо
  сместиться на вектор (0, 3)
  сместиться на вектор (1, 0)
  сместиться на вектор (0,-1)
  сместиться на вектор (1, 0)
  сместиться на вектор (0,-1)
  сместиться на вектор (1, 0)
  сместиться на вектор (0,-1)
  сместиться в  точку  (2, 1)
  поднять перо
кон
```

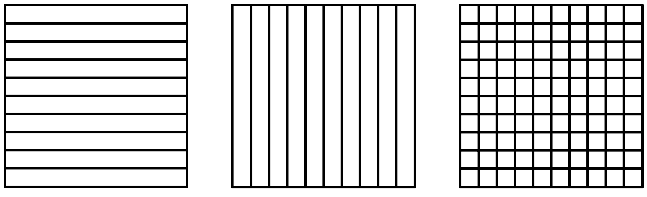
а) не выполняя алгоритма и не рисуя получившейся фигуры, определите, где будет расположено перо после выполнения алгоритма, будет ли оно поднято или опущено;
б) выполните алгоритм, нарисуйте получившуюся фигуру;
в) переделайте алгоритм так, чтобы он рисовал где-нибудь на плоскости фигуру вдвое большего размера;
г) переделайте алгоритм так, чтобы он рисовал фигуру, симметричную первой относительно оси *y*;
д) определите, что будет нарисовано, если в алгоритме изменить знаки всех аргументов на противоположные.

14. Дан алгоритм:

```
алг ломаная                                  (A43)
 дано | перо в начале координат и поднято
нач
  опустить перо
  сместиться на вектор (1, 3)
  сместиться на вектор (1, 2)
  сместиться на вектор (1, 1)
  сместиться на вектор (1, 0)
  сместиться на вектор (1,-1)
  сместиться на вектор (1,-2)
  сместиться на вектор (1,-3)
  поднять перо
кон
```

Не выполняя алгоритма и не рисуя получившейся ломаной, определите:
а) будет перо после выполнения поднято или опущено;
б) координату *x* конечного положения пера;
в) координату *y* конечного положения пера;
г) будет ли ломаная замкнута;
д) расстояние между концами ломаной. Нарисуйте ломаную, проверьте ваши ответы.

15. Составьте алгоритм управления Чертежником, после исполнения которого будет нарисован квадрат 4×4, заштрихованный горизонтальными и (или) вертикальными линиями следующим образом, как на *рис.* 52 (расстояние между линиями равно 0,4).



а *б* *в*

Рис. 52

16. Вспомогательный алгоритм картинка рисует некоторую картинку в квадрате 1×1 и возвращает перо в начальное положение — левый нижний угол квадрата. Составьте алгоритм, рисующий 100 экземпляров картинки в квадрате 10×10.

§ 14. Алгоритмы с аргументами

14.1. Пример алгоритма с аргументом

Вспомните упражнения 7*б* и 7*в* из предыдущего параграфа. Для рисования квадратов с длинами сторон 4 и 6 вы составляли два разных алгоритма, отличающихся только *числами* в командах.

А как быть, если нужно рисовать много разных квадратов с разными длинами сторон? Можно, конечно, составить множество похожих алгоритмов, но делать этого очень не хочется! Было бы намного удобнее создать единый алгоритм-образец, в который компьютер сам подставлял бы каждый раз нужные числа.

Для подобных целей в алгоритмическом языке существуют алгоритмы с ***аргументами***. Например, алгоритм рисования квадрата с произвольной длиной стороны может выглядеть так:

```
алг квадрат (арг вещ a)                    (A44)
  дано | перо Чертежника в левом нижнем
       | углу будущего квадрата, перо поднято
  надо | нарисован квадрат с длиной стороны a
       | перо Чертежника в исходной точке,
       | поднято
нач
| опустить перо
| сместиться на вектор (a, 0)
| сместиться на вектор (0, a)
| сместиться на вектор (-a, 0)
| сместиться на вектор (0,-a)
| поднять перо
кон
```

Запись **алг** квадрат (**арг** **вещ** a) означает, что у алгоритма квадрат есть один аргумент (**арг**) a, который может быть произвольным числом.

Слово **вещ** — это сокращение от “вещественный”, этим словом в информатике принято обозначать числа, которые могут быть целыми или дробными (в математике такие числа называют действительными). Это слово описывает ***тип*** аргумента. С понятием типа мы уже встречались при изучении баз данных. Напомним, что тип указывает, какие значения может принимать величина и какие действия можно с ней выполнять.

Позже мы познакомимся с алгоритмами, аргументы которых могут быть лишь целыми (**цел**) числами; с алгоритмами, аргументы которых вообще не являются числами, а также с алгоритмами, у которых, кроме аргументов, есть результаты (**рез**). Именно этим объясняется необходимость слов **арг** и **вещ** в заголовке алгоритма квадрат.

14.2. Выполнение вспомогательного алгоритма с аргументами

Если в основном алгоритме написать вызов квадрат (4), то компьютер запомнит, что аргумент a равен 4, и при выполнении алгоритма квадрат (**арг** **вещ** a) скомандует Чертежнику:

```
опустить перо
сместиться на вектор (4, 0)
сместиться на вектор (0, 4)
сместиться на вектор (-4, 0)
сместиться на вектор (0,-4)
поднять перо
```

Если же в основном алгоритме написать вызов квадрат (6), то компьютер запомнит, что аргумент a равен 6, и скомандует Чертежнику:

```
опустить перо
сместиться на вектор (6, 0)
сместиться на вектор (0, 6)
сместиться на вектор (-6, 0)
сместиться на вектор (0,-6)
поднять перо
```

14.3. Модель памяти компьютера

Поясним подробнее, как компьютер запоминает значения аргументов в своей памяти. Память компьютера удобно представлять в виде обычной классной доски, на которой можно записывать информацию, читать, стирать, записывать заново и т.д. Место, отводимое в памяти компьютера для запоминания информации, удобно изображать прямоугольником, внутри которого записывается сама информация (значение аргумента). Каждый такой прямоугольник мы будем называть ***ячейкой*** памяти.

При выполнении каждого алгоритма ему выделяется отдельный участок памяти, в котором создаются ячейки для аргументов. После окончания выполнения алгоритма эта память освобождается. Память алгоритма удобно изображать в виде большого прямоугольника, внутри которого находятся прямоугольники меньшего размера, соответствующие ячейкам.

Возможный вид памяти при исполнении алгоритма квадрат показан на *рис.* 53 (здесь a = 4).

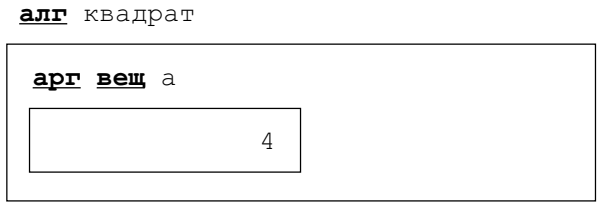


Рис. 53

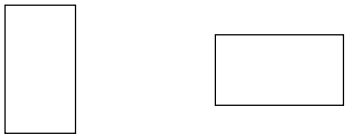
14.4. Алгоритмы с несколькими аргументами

Алгоритм может иметь несколько аргументов. Рассмотрим пример.

```
алг прямоугольник (арг вещ a,b)            (A45)
  дано | перо Чертежника в левом нижнем
       | углу A будущего прямоугольника
       | и поднято
  надо | нарисован прямоугольник со
       | сторонами a и b
       | перо Чертежника в точке A и поднято
```

```
нач
| опустить перо
| сместиться на вектор (a, 0)
| сместиться на вектор (0, b)
| сместиться на вектор (-a, 0)
| сместиться на вектор (0,-b)
| поднять перо
кон
```

При вызове такого алгоритма важно правильно задать порядок следования аргументов. Соответствие между аргументами в команде вызова и в заголовке алгоритма устанавливается по порядку их следования. На *рис.* 54*а* показано изображение, полученное при вызове прямоугольник (3,7), а на *рис.* 54*б* — изображение, полученное при вызове прямоугольник (7,3).



а *б*

Рис. 54

14.5. Аргументы в заголовке цикла n раз

Аргументы алгоритма можно использовать в любых командах алгоритмического языка, где могут быть числа, в том числе в заголовке цикла **n раз**.

Используя цикл **n раз** с аргументом, можно составить алгоритмы для смещения Робота на заданное число клеток в нужном направлении, например:

```
алг влево на (арг цел n)                          (A46)
  дано | на поле Робота стен нет
  надо | Робот сместился на n клеток влево
нач
   нц n раз
      влево
   кц
кон
```

Аналогично можно составить алгоритмы вниз на (**арг цел n**), вверх на (**арг цел n**) и вправо на (**арг цел n**). В дальнейшем мы часто будем пользоваться этими алгоритмами как вспомогательными.

Обратите внимание, что, не используя цикл **n раз**, алгоритм (A46) составить нельзя. Таким образом, применение аргументов открывает новую сторону в использовании этой простой конструкции: цикл **n раз** с аргументом не только сокращает запись, но и позволяет решать задачи, для которых невозможен простой линейный алгоритм.

14.6. Закрашивание прямоугольника

В § 9 мы решали задачу о закрашивании Роботом прямоугольника. Тогда мы вынуждены были указывать в алгоритме конкретные размеры прямоугольника. Используя аргументы, мы можем переписать этот алгоритм так, чтобы при его выполнении закрашивался прямоугольник произвольного размера.

```
алг закрасить прямоугольник (арг цел m,n)  (A47)
  дано | на поле Робота стен нет
  надо | закрашен прямоугольник размером m*n.
       | Робот в исходном положении
нач
| нц n раз
| |   закрасить ряд(m)
| |   вниз
| кц
| вверх на(n)
кон
```

```
алг закрасить ряд (арг цел m)                    (A48)
  дано | на поле Робота стен нет
  надо | Робот закрасил m клеток вправо
       | и вернулся в исходное положение
нач
| нц m раз
|  | закрасить; вправо
| кц
| влево на(m)
кон
```

14.7. Заголовок алгоритма с аргументами

Как мы уже знаем (п. 7.6), заголовок алгоритма описывает условие задачи, а тело алгоритма — ее решение. Чтобы записать заголовок алгоритма, обычно достаточно внимательно изучить условие, не думая пока о решении.

При построении алгоритмов с аргументами важно точно определить количество аргументов и их типы. Для этого нужно изучить условие задачи и выделить в нем ту информацию, которую необходимо задать, прежде чем приступать к решению. Этой информации будут соответствовать аргументы алгоритма.

Например, в задаче построить квадрат такой дополнительной информацией будет сторона квадрата, поэтому у алгоритма появляется один аргумент.

В общем случае переменным в условии задачи соответствуют аргументы в заголовке алгоритма.

Задачи и упражнения

1. Какую последовательность команд выполнит Чертежник и что будет нарисовано при вызовах:

а) квадрат (0);

б) квадрат (—1)?

2. Составьте алгоритм прямоугольник (**арг вещ** x,y,a,b), который рисует прямоугольник с длинами сторон a и b, начиная и заканчивая в углу — точке (x, y) (*рис.* 55).

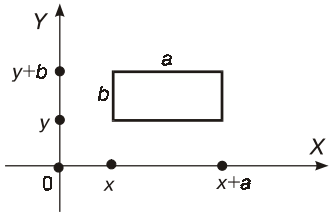
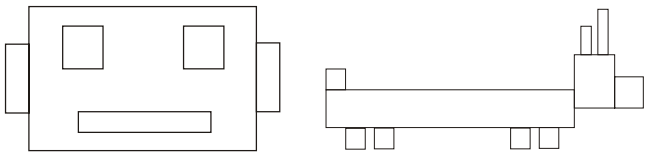


Рис. 55

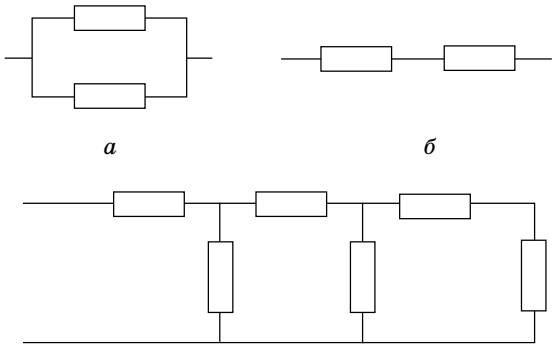
3. Используя алгоритм прямоугольник (упр. 2), составьте алгоритмы рисования робота и собачки (*рис.* 56).



а) Робот *б) Собачка*

Рис. 56

4. Составьте алгоритмы рисования схем (*рис.* 57).



а *б* *в*

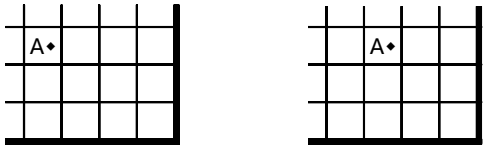
Рис. 57

5. Придумайте какую-нибудь картинку, составленную из прямоугольников. Составьте алгоритм для рисования этой картинки.

6. Сколько клеток будет закрашено и сколько команд компьютер выдаст Роботу при выполнении вызова

а) `закрасить прямоугольник (1, 1);`
б) `закрасить прямоугольник (0, 11);`
в) `закрасить прямоугольник (9, 0);`
г) `закрасить прямоугольник (9, 11)`?

7. Опишите, как будет выполняться вызов `закрасить прямоугольник (3, 3)` в ситуациях, изображенных на *рис.* 58.



а *б*

Рис. 58

8. Измените алгоритмы `закрасить ряд (A48)` и `закрасить прямоугольник (A47)` так, чтобы при вызове `закрасить прямоугольник(3, 3)` в ситуации на *рис.* 58*б* отказа не возникало, а оказался закрашенным квадрат 3 на 3 клетки.

9. Нарисуйте результат выполнения алгоритма:

а)

```
алг тоннель                                   (A49)
нач
| квадрат(10); сместиться на вектор(1, 1)
| квадрат(7);  сместиться на вектор(1, 1)
| квадрат(4);  сместиться на вектор(1, 1)
| квадрат (1)
кон
```

б)

```
алг спираль                                   (A50)
нач
| опустить перо
| виток(1); виток(3); виток(5);
| виток(7); виток(9)
| поднять перо
кон
```

```
алг виток(арг вещ a)                          (A51)
нач
| сместиться на вектор(a, 0)
| сместиться на вектор(0, -a)
| сместиться на вектор(-a-1, 0)
| сместиться на вектор(0, a+1)
кон
```

10. Что нарисует Чертежник при выполнении алгоритма `спираль (A50)`, если в алгоритме `виток (A51)` заменить `– a – 1` на `– a – a` и `a + 1` заменить на `a + a`?

11. Измените алгоритм `виток (A51)` так, чтобы спираль в алгоритме `(A50)` раскручивалась против часовой стрелки.

12. Составьте алгоритм рисования спирали, изображенной на *рис.* 59.

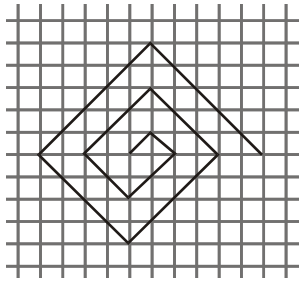


Рис. 59

13. Измените ваше решение упражнения 12 так, чтобы расстояние между витками при каждом новом витке увеличивалось.

14. Нарисуйте результат выполнения алгоритма орнамент:

```
алг орнамент                                  (A52)
  дано | перо Чертежника в левом верхнем
       | углу будущего орнамента размером
       |  12 × 12 и поднято
  надо | нарисован орнамент, перо в левом
       | нижнем углу и поднято
нач
|      ряд; ряд; ряд
кон
```

```
алг ряд                                       (A53)
  дано | перо Чертежника в левом верхнем
       | углу будущего ряда размером 12 × 4
       | и поднято
  надо | нарисован ряд, перо в левом нижнем
       | углу ряда и поднято
нач
|    фрагмент; фрагмент; фрагмент
|    сместиться на вектор(-12,-4)
кон
```

```
алг фрагмент                                    (А54)
  дано | перо Чертежника в левом верхнем
       | углу будущего фрагмента
       | размером 4 × 4 и поднято
  надо | нарисован фрагмент, перо в правом
       | верхнем углу и поднято
нач
    опустить перо
    сместиться на вектор(2,-2)
    сместиться на вектор(-2,-2)
    поднять перо;
    сместиться на вектор(4,0);
    опустить перо
    сместиться на вектор(0, 1)
    сместиться на вектор(-2, 0)
    сместиться на вектор(0, 2)
    сместиться на вектор(2, 0)
    сместиться на вектор(0, 1)
    поднять перо
кон
```

15. По образцу упражнения 13 составьте алгоритмы рисования орнаментов (*рис.* 60).

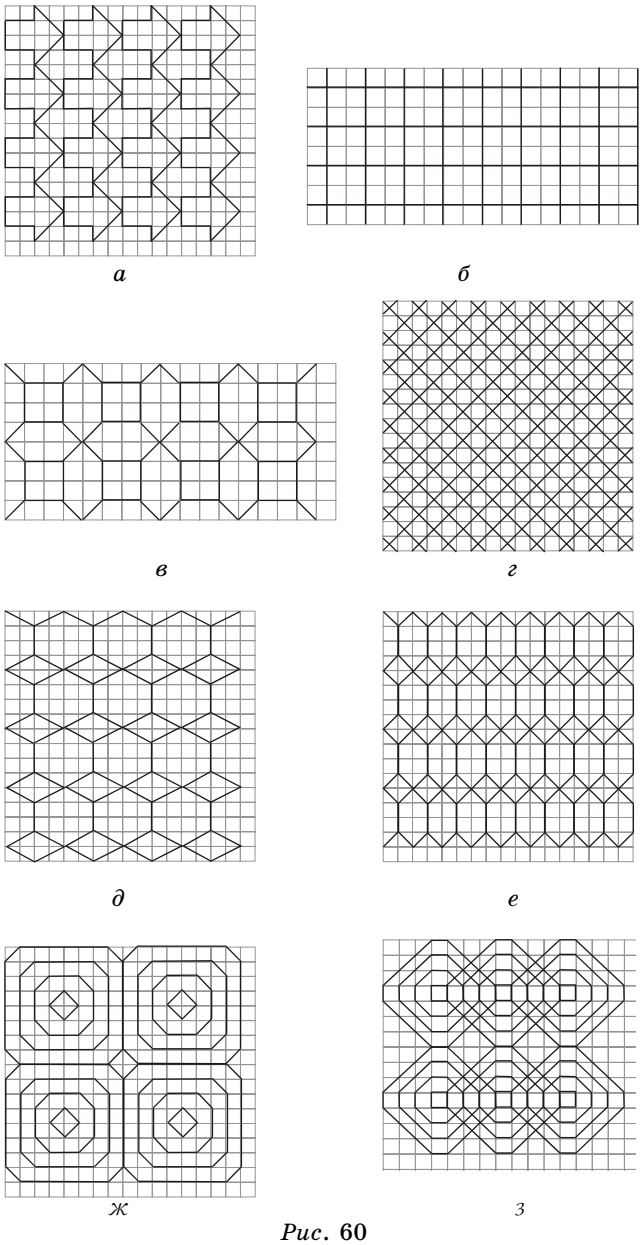


Рис. 60

16. Составьте алгоритм `горизонтальная ломаная` (**арг цел** `n`, **арг вещ** `a`), рисующий с помощью Чертежника ломаную линию с $2n$ звеньями, показанную на *рис.* 61.

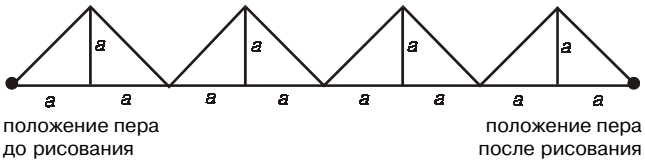


Рис. 61

17. Составьте алгоритм `вертикальная ломаная` (**арг цел** `n`, **арг вещ** `a`), рисующий ломаную из упражнения 16, повернутую вокруг начального положения пера на 90 градусов по часовой стрелке.

18. Используя алгоритмы из упражнений 16 и 17 как вспомогательные, составьте алгоритмы, рисующие:

а) m горизонтальных ломаных с $2n$ звеньями одна под другой на расстоянии b друг от друга;

б) m вертикальных ломаных с $2n$ звеньями одна под другой на расстоянии b друг от друга.

19. Компьютер выполнил последовательность команд:

```
горизонтальная ломаная(5, 1)
вертикальная   ломаная(7, 1)
горизонтальная ломаная(5, -1)
вертикальная   ломаная(7, -1)
```

Что нарисовал Чертежник?

20. Составьте алгоритм с целыми аргументами `m` и `n`, который с помощью Робота закрашивает клетки, отмеченные на *рис.* 62.

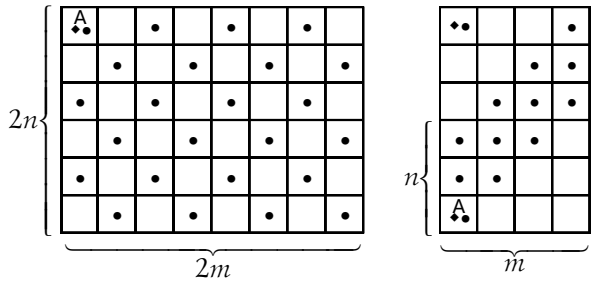


Рис. 62

§ 15. Арифметические выражения и правила их записи

15.1. Арифметические выражения в алгоритмическом языке

В алгоритмическом языке можно использовать не только числа, но и числовые и алгебраические выражения (формулы). В информатике эти выражения называются ***арифметическими***. Правила алгоритмического языка позволяют при записи алгоритма всюду, где можно написать число, написать и произвольное арифметическое выражение. Рассмотрим пример:

```
алг нарисовать М размером (арг вещ a,b)  (A55)
  дано | перо Чертежника в точке А (рис.70)
       | и поднято
  надо | нарисована буква М ширины a
       | и высоты b, перо поднято
       | и находится в точке Б (рис.70)
нач
|   опустить перо
|   сместиться на вектор (0, b)
|   сместиться на вектор (a/2, -b/2)
|   сместиться на вектор (a/2,  b/2)
|   сместиться на вектор (0, -b)
|   поднять перо
|   сместиться на вектор(a/2, 0)
кон
```

Знак "/" здесь означает деление. Таким образом, запись a/2 означает $a : 2$, (a+b)/c означает $(a + b) : c$ и т.д.

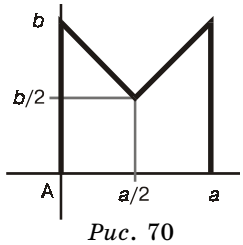


Рис. 70

15.2. Выражения вычисляет компьютер

При выполнении вызова нарисовать М размером (3, 5) компьютер запомнит, что a = 3, b = 5, вычислит все выражения и скомандует Чертежнику:

```
опустить перо
сместиться на вектор (0, 5)
сместиться на вектор (1.5, -2.5)
сместиться на вектор (1.5,  2.5)
сместиться на вектор (0, -5)
поднять перо
сместиться на вектор (1.5, 0)
```

Чертежник никаких вычислений не производит — он лишь выполняет последовательно поступающие команды с конкретными числовыми аргументами.

15.3. Правила записи арифметических выражений в алгоритмическом языке

Арифметические выражения в алгоритмическом языке должны быть записаны в так называемой ***линейной*** записи согласно следующим правилам:

— выражение должно быть записано в виде линейной цепочки символов (вместо x_1 и v_0 надо писать x1, v0);

— для обозначения операции умножения используется звездочка (*), для операции деления — косая черта (/), для операции возведения в степень — две звездочки (**);

— нельзя опускать знаки операций, например, писать 4*a. Для записи произведения чисел 4 и a надо писать 4*a;

— аргументы функций (sin, cos и др.), как и аргументы вспомогательных алгоритмов, записываются в круглых скобках, например, sin(x), cos(4*a);

— для изменения порядка действий используются круглые скобки.

Линейная запись позволяет вводить выражения в память компьютера, последовательно нажимая на соответствующие клавиши на клавиатуре.

15.4. Операции и стандартные функции алгоритмического языка

Основные операции и функции алгоритмического языка приведены в табл. 14.

Таблица 14

Операции и стандартные функции алгоритмического языка

Название операции или функции	Форма записи
сложение	x + y
вычитание	x − y
умножение	x * y
деление	x / y
возведение в степень	x ** y
корень квадратный $\sqrt{x}$	sqrt(x)
абсолютная величина x	abs(x)
знак числа (−1, 0 или 1)	sign(x)
синус sin x	sin(x)
косинус cos x	cos(x)
тангенс tg x	tg (x)
котангенс ctg x	ctg(x)
арксинус arcsin x	arcsin(x)
арккосинус arccos x	arccos(x)
арктангенс arctg x	arctg (x)
арккотангенс arcctg x	arcctg(x)
натуральный логарифм ln x	ln(x)
десятичный логарифм lg x	lg(x)
степень числа e ($e \approx 2{,}718281$) e^x	exp(x)
минимум из чисел x и y	min(x,y)
максимум из чисел x и y	max(x,y)
остаток от деления x на y (x, y — целые)	mod(x,y)
частное от деления x на y (x, y — целые)	div(x,y)
целая часть x, т.е. максимальное целое число, не превосходящее x	int(x)
случайное число от 0 до x	rnd(x)

15.5. Порядок действий в арифметических выражениях

При вычислении арифметических выражений компьютер выполняет действия в следующем порядке:

1) вычисляются выражения в скобках (в том числе аргументы функций); порядок действий внутри скобок определяется теми же правилами (то есть сначала вычисляются скобки внутри скобок и т.д.);
2) вычисляются значения функций;
3) справа налево выполняются возведения в степень;
4) слева направо выполняются умножения и деления;
5) слева направо выполняются сложения и вычитания.

Возведение в степень справа налево означает, что запись `a**b**c` следует понимать как `a^(b^c)`, но не как `(a^b)^c`.

Умножение, деление, сложение и вычитание выполняются слева направо. Например, запись `a + b — c` означает `(a + b) — c`, но не `a + ( b — c)`.

15.6. Примеры записи арифметических выражений на алгоритмическом языке

1) $\frac{-1}{x^2}$ `-1/x**2`

2) $\frac{a}{bc}$ `a/(b*c)`

3) $\frac{a}{b}c$ `a/b*c` или `(a/b)*c`

4) $2^{2^{2^n}}$ `2**2**2**n` или `2**(2**(2**n))`

5) x^{y^z} `x**y**z` или `x**(y**z)`

6) $\frac{-b+\sqrt{b^2-4ac}}{2a}$ `(-b+sqrt(b**2—4*a*c))/(2*a)`

7) $\sqrt{p(p-a)(p-b)(p-c)}$ `sqrt(p*(p-a)*(p-b)*(p-c))`

8) $\frac{a+b+c}{2}$ `(a+b+c)/2`

9) $\sqrt{a^2+b^2-2ab\cos\gamma}$ `sqrt(a**2+b**2—2*a*b*cos(gamma))`

10) $\frac{ad+bc}{bd}$ `(a*d+b*c)/(b*d)`

11) $\sin\alpha\cos\beta+\cos\alpha\sin\beta$ `sin(alfa)*cos(beta)+cos(alfa)*sin(beta)`

Задачи и упражнения

1. Вычислить значение выражения, записанного на алгоритмическом языке:

а) 24 / (3 * 4) — 24 / 3 / 4 + 24 / 3 * 4;
б) 40 / (4 * 5) — 40 / 4 / 5 + 40 / 4 * 5;
в) (2 + 3 * 4) / 2 + 5 — (2 + sqr(4));
г) 60 / (23 — (2 + 3 * sqrt(5 — abs(1 — 3) + + sqr(—1))));
д) 71 + abs((16 — 7 * 2) / 2) — sqr(sqrt(225)).

2. Переведите из линейной записи в обычную:

а) *a* / *b* / *c*;
б) *a* * *b* / *c*;
в) *a* / *b* * *c*;
г) *a* / *b* ** *c*;
д) *a* + *b* / *c*;
е) (*a* + *b*) / *c*;
ж) *a* / *b* ** *c* ** *d*;
з) 1 / (1 + *x* * *x*);
и) 1 / (1 + *x* ** 2).

3. Переведите из линейной записи в обычную:

а) 1 / sqrt(1 + *x* ** 2);
б) sqrt(*x* ** 2 + *y* ** 2);
в) *x* ** (1 / 3);
г) *x* ** (—1 / 3);
д) 1 / *x* ** (1 / 3);
е) sin(*x*) ** 2 + sin(*y*) ** 2;
ж) sin (*x* ** 2) + sin(*y* ** 2);
з) *a* + *b* / *c* + *d*;
и) (*a* + *b*) / (*c* + *d*);
к) *a* / sin(*A*).

4. Переведите из линейной записи в обычную:

а) sqrt(tg(*A* + *B*)) / sqrt(tg(*A* — *B*));
б) 1 / 2 * *a* * *b* *sin(*C*);
в) sqrt(*b* ** 2 + *c* ** 2 + 2 * *b* * *c* * cos(*A*)) / 2;
г) 2 * *b* * *c* * cos(*A* / 2) / (*b* + *c*);
д) sqrt((*p* — *a*) * (*p* — *b*) * (*p* — *c*) * *p*);
е) 4* *R* * sin(*A* / 2) * sin(*B* / 2) * sin(*C* / 2);
ж) (*a* * *x* + *b*) / (*c* * *x* + *d*);
з) sqrt(*a* * *x* ** 2 + *b* * *x* + *c*);
и) arctg(*x* / sqrt(1 — *x* ** 2));
к) 2 * sin((*alfa* + *beta*) / 2) * cos((*alfa* — *beta*) / 2).

5. Запишите по правилам алгоритмического языка выражения:

а) $\sqrt{x_1^2+x_2^2}$

б) $x_1x_2+x_1x_3+x_2x_3$

в) $v_0t+\frac{at^2}{2}$

г) $\frac{1}{R_1}+\frac{1}{R_2}$

д) $|x|+|x+1|$

е) $\sqrt{a^2+b^2-2ab\cos\gamma}$

ж) $a\sin\beta+b\cos\beta$

з) $\frac{1}{\sqrt{ax^2+bx+c}}$

и) $\frac{\sqrt{x+1}-\sqrt{x-1}}{2\sqrt{x}}$

к) $\big|1-|x|\big|$

л) $\sqrt{a-b\sqrt{x+y}}$

м) $\frac{1+\sqrt{x^2-y^2}}{1-|x+y|}$

н) $\frac{\sin(x-y)}{\sqrt{\cos(x+y)+2}}$

Продолжение следует

насколько это письмо взволновало Берга. Писал ему старый, очень больной человек, который оказался "коллегой" Берга по одной из тюрем. Вернее, он диктовал письмо дочери, потому что был слепым. Он узнал об Акселе Ивановиче из прессы. В письме содержались воспоминания о тех днях, когда Берг был старостой тюремной камеры, и выражалась сердечная благодарность Бергу за его оптимизм, доброжелательность, готовность помочь. Автор письма уверял, что выжил в тех страшных застенках только благодаря моральной поддержке Акселя Ивановича. В книге Ирины Радунской "Аксель Берг — человек XX века" и в сборнике воспоминаний "Путь в большую науку: академик Аксель Берг" тюремная эпопея Берга не рассматривается (цензура не пропустила). Воспоминания дочери Берга об аресте отца по этим соображениям не были включены в сборник. С.С. Масчан сберегла эту рукопись, и, когда подготавливался сборник воспоминаний к столетию со дня рождения Берга, нам удалось впервые опубликовать эти воспоминания (Академик Аксель Иванович Берг. М.: Государственный политехнический музей, 1993).

А.И. Берг (слева), 1963 г.

Но вернемся к Совету шестидесятых годов.

Среди тех, кто боролся за кибернетику, развивал новые направления, был академик АМН СССР Василий Васильевич Парин — основатель и председатель секции "Медицинская кибернетика". Бывший узник ГУЛАГа, отморозивший ноги на лесоповале в Сибири, он передвигался медленно, с трудом. ГУЛАГ не давал себя забыть. Мы, сотрудники Совета, часто наблюдали такую сцену: академик Парин медленно входит в берговский кабинет, в углу которого стояла огромная пальма. Берг быстро поднимается со своего кресла, стремительно направляется навстречу Парину, крепко пожимает руку. И два бывших узника тоталитарной системы долго беседуют за длинным берговским столом, покрытым зеленой скатертью из тонкого сукна, о путях развития кибернетики.

Имя Василия Васильевича Налимова (1910—1997) стало известно в кибернетических кругах после опубликования его статьи "Научная и техническая информация как одна из задач кибернетики" в соавторстве с Г.Э. Влэдуцем и Н.И. Стяжкиньм в 1959 г. в "Успехах физических наук".

По предложению Берга в 1961 г. была организована секция "Химическая кибернетика", председателем которой стал В.В. Налимов. В 1963 г. он защитил докторскую диссертацию "Метрологические аспекты химической кибернетики". В 1965 г. академик Колмогоров, создавая в МГУ Межфакультетскую лабораторию статистических методов, пригласил Налимова работать в этой лаборатории в качестве его заместителя. Лаборатория стала ведущей организацией в области математической статистики и планирования эксперимента в нашей стране и получила мировое признание. Ну а при чем здесь ГУЛАГ? А при том, что до этого В.В. Налимов 18 лет провел в заключении и в ссылках (Колыма — Казахстан). И отец его, профессор МГУ, в 1939 г. погиб в сыктывкарской тюрьме.

Василий Васильевич Налимов был арестован в Москве в 1937 г. по групповому делу "анархистов-мистиков". Дело это столь необычно, что о нем стоит рассказать хотя бы вкратце. Анархисты-мистики являлись религиозно-философским направлением, которое в России было связано с именами профессора Аполлона Андреевича Карелина (1887—1926) и математика, доцента МВТУ (в то время это был Механико-машиностроительный институт) Алексея Александровича Солоновича (1887—1937). После смерти Карелина Солонович возглавлял это движение. Одним из его помощников был Сергей Романович Лящук (1887—1968), доцент кафедры высшей математики МВТУ, где работал и Солонович. Таким образом, кафедра высшей математики МВТУ стала своего рода центром анархистов-мистиков. Налимов входил в число молодых членов этой организации. Во время следствия религиозно-философская группа трактовалась как контрреволюционная организация, поставившая перед собой цель — свержение советской власти. По делу анархистов-мистиков к высшей мере наказания было приговорено 9 человек, среди них жена А.А. Солоновича Агния Солонович, ученый-геофизик и историк А.А. Синягин (арестован в Томске), ученый-востоковед Ю.К. Щуцкий (арестован в Ленинграде). Сам А.А. Солонович скончался в тюремной больнице в марте 1937 г. В своей автобиографической книге "Канатоходец" (1993) Налимов подробно описал дело "анархистов-мистиков" (следственное дело № 10719) и привел тексты допросов. Начало шестидесятых годов было тем временем, когда под кибернетической крышей специалисты разных направлений учились взаимодействовать друг с другом. Секция

"Химическая кибернетика" собрала химиков, технологов, математиков, статистиков, экономистов, специалистов по вычислительной технике. Перед ними была поставлена задача оптимального управления химическими и химико-технологическими процессами и задача оптимизации экспериментальных исследований. В вопросах математического описания и моделирования члены секции разделились на два лагеря. "Трубадуры черного ящика" (выражение это придумано А.В. Нетушилом — по его словам) придерживались принципа: абстрагируясь от неизученной сущности процесса, будем строить математическую модель на основе информации о входных и выходных переменных. Классически воспитанные химики, "детерминисты", не желали "абстрагироваться от сущности". Они хотели ее изучать и строить модель на основе уравнений химической кинетики, законов переноса реагирующих веществ и продуктов реакции, передачи тепла и гидродинамики. Они были "трубадурами прозрачного ящика". Среди них был и академик АН СССР Алексей Александрович Баландин (1898—1967) — автор мультиплетной теории гетерогенного катализа. Ученик академика Н.Д. Зелинского, он окончил в 1923 г. МГУ по специальности "физическая химия". Арестовали его в декабре 1936 г., он был в это время профессором МГУ. Обвинение — "подготовка взрывчатых веществ для теракта". Накануне "террорист" стал лауреатом премии им. Д.И. Менделеева. В марте 1937 г. Баландин был сослан в Оренбург на 5 лет. Нужно отдать должное коллегам-химикам: за Баландина ходатайствовали Н.Д. Зелинский, А.Н. Бах, В.И. Вернадский, А.Н. Фрумкин. В 1939 году Баландина освободили, он вернулся в Москву и стал руководителем группы "Кинетика контактных процессов" в лаборатории органического катализа Института органического синтеза АН СССР. В 1948 г. Баландину было присвоено академическое звание и он стал деканом химфака МГУ. Казалось бы, черные дни миновали, можно спокойно работать, заниматься любимым делом. Но нет — в 1949 г. второй арест, 10 лет ИТЛ (исправительно-трудовых лагерей), Норильск, 4 года тяжелых подконвойных работ. В 1951 г. Баландину удалось устроиться химиком на Норильском горно-металлургическом комбинате. Смерть Сталина сократила десятилетний срок заключения, и в 1953 г. Баландин вернулся в Москву. Он создал в МГУ первую в мире лабораторию органического катализа. Его сотрудничество с секцией "Химическая кибернетика" относится к первой половине шестидесятых годов. К сожалению, в 1967 г. Алексей Александрович Баландин покинул этот мир. (О гулаговской судьбе Баландина рассказал Ю.И. Соловьев в сборнике "Трагические судьбы: репрессированные ученые Академии наук СССР", М.: Наука, 1995, с. 182—194.)

Если председатель секции "Химическая кибернетика" В.В. Налимов был в прошлой гулаговской жизни колымчанином, то ученый секретарь секции — воркутянкой. Речь пойдет об авторе этих строк. Меня арестовали... Но нет, обо мне несколько позже. Я ведь принадлежу к репрессированным в третьем поколении, сначала о том, что было до меня.

Мой дедушка с отцовской стороны, священник Платон Дмитриевич Иванов, погиб в 1920 г. Тогда же погиб и мой дядя, тоже священник. Мой дедушка по материнской линии, Михаил Васильевич Корибут-Дашкевич, дворянин, поляк, погиб в 1929 г. Его обвинили в польском буржуазном национализме. Он якобы собирался продать Украину Польше. Мой отец, Владимир Платонович Иванов, учитель, расстрелян в 1937 г. Моя мама, Вацлава Михайловна, урожденная Корибут-Дашкевич, арестована в 1938 г.

Я окончила 10 классов в роковой 1941 год. Послала аттестат в ЛГУ. С раннего детства я мечтала быть астрономом. Меня влекло небо. Отличников тогда принимали без экзаменов. Но жизнь моя пошла по другому сценарию. Оккупация, гестаповская тюрьма, освобождение, советская тюрьма, 15 лет каторжных работ, этап на Воркуту. Я в моей прошлой гулаговской жизни была не заключенной, а каторжанкой. А это большая разница. Каторжане находились на более низком круге ада, чем заключенные. В гулаговских документах они проходили по отдельным спискам и имели свою аббревиатуру — "КТР" (почти что КТН). Заключенные же обозначались "З/К", отсюда — "зеки". Каторга была введена специальным указом от 22.04.43 г. Под этот указ я и попала. Вместо ЛГУ мне пришлось обучаться в Воркутинской академии им. Лаврентия Берии. Что же касается Совета по кибернетике, то все годы, которые мне выпало счастье работать там (1961—1981), я была штатным сотрудником Совета: в шестидесятые годы — ученым секретарем секции "Химическая кибернетика", в семидесятые — заместителем председателя секции "Математическая теория эксперимента". Кандидатскую диссертацию я защитила в 1965 г. на тему "Планирование эксперимента при оптимизации процессов тонкого органического синтеза", докторскую — в 1971 г. по применению комбинаторного анализа в планировании эксперимента. Многие из моих опубликованных книг посвящены проблеме планирования многофакторных экспериментов и анализа данных. Например: "Комбинаторные планы в задачах многофакторного эксперимента" (Наука, 1979, в соавторстве с А.Н. Лисенковым); "Дисперсионный анализ и синтез планов на ЭВМ" (Наука, 1982, в соавторстве с В.И. Денисовым, И.А. Полетаевой, В.В. Пономаревым — сотрудниками кафедры прикладной математики НЭТИ); "Планирование и анализ многофакторных экспериментов на основе комбинаторных схем" (издательство Иркутского университета, 1993, в соавторстве с Л.Н. Ежовой). Что же касается неопубликованных книг, то они, конечно, посвящены воспоминаниям о прошлой гулаговской жизни.

Таким образом, возникновение и развитие таких научных направлений, как химическая кибернетика и математическая теория эксперимента, было представлено жителями самых страшных, самых северных островов архипелага ГУЛАГ — Колымы, Норильска и Воркуты.

21.06.97 г.

Психологические особенности восприятия кодовой таблицы ASCII

П.И. Совертков

В школьных учебниках [1, с. 35; 2, с. 40; 3, с. 85] рассматривается кодирование информации с помощью таблицы ASCII.

В [2] говорится, что каждому символу сопоставлен десятичный код. В тексте понятие "десятичный" не употребляется, хотя ясно, что именно это имеется в виду. Причем учитель обязательно должен знать, что машина с таким кодом не работает и воспринимать его следует как номер символа в таблице. При необходимости код можно перевести в шестнадцатеричную или двоичную систему счисления. Таким образом, изложение кодировки в [2] оказалось логически не завершенным. Правда, такое объяснение можно считать допустимым на начальной ступени обучения, т.к. учащиеся не знакомы с переводом чисел в двоичную систему счисления.

В [1] предпринята попытка полностью объяснить, как кодируется каждый символ для представления в компьютере, т.е. как преобразуется информация, представленная в естественном, удобном для восприятия человеком виде. Но, к сожалению, в этом учебнике на одной странице (с. 36) собрано чересчур много сведений — двоичная, десятеричная и шестнадцатеричная системы счисления, объяснение, как пользоваться кодовой таблицей ASCII, — что и привело самих авторов к ошибкам на с. 37.

При использовании в тексте число 45 (лучше было бы назвать это не числом, а набором символов) по таблице ASCII зашифровано авторами так:

Символ 4 в шестнадцатеричной системе получил код 43, а затем в двоичной системе 01000011.

Символ 5 в шестнадцатеричной системе получил код 53, а затем в двоичной системе 01010011.

Жаль, что мысль авторов оказалась логически прерванной. В учебнике нет окончательного варианта представления 45 в виде 01000011 01010011.

Таким образом, авторы учебника фактически воспользовались следующим правилом кодировки:

Для символа в таблице определяются строка и столбец, на пересечении которых находится рассматриваемый символ. Потом выписывается вначале номер строки, а затем номер столбца.

Опытные программисты поймут, что здесь переставлена последовательность считывания номеров строки и столбца.

Правильное решение:

$4 \rightarrow 34 \rightarrow 00110100$

$5 \rightarrow 35 \rightarrow 00110101$

$45 \rightarrow 00110100 \;\; 00110101$

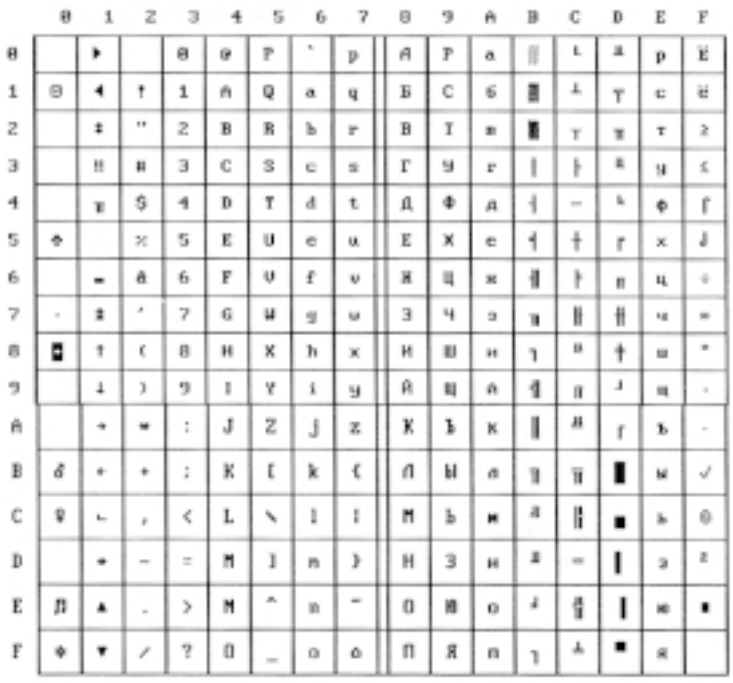
Еще раз подчеркиваю, предположение о том, что авторы учебника пользуются именно таким, неверным, методом работы с таблицей, как мы описали выше, сделан нами на основании приведенных кодов. Справедливости ради заметим, что перед этим на стр. 35 приведен правильный код буквы "*S*".

При пользовании таблицей ASCII надо обязательно понимать ее отличие от большинства таблиц с двумя входами, с которыми раньше работали учителя и учащиеся.

Наше дальнейшее разъяснение предназначено для учителей. Мы хотим обратить их внимание на психологический момент при объяснении этого отличия.

Пользуясь разнообразными таблицами, например, возводя число 48 в квадрат по четырехзначным таблицам Брадиса, школьники приучаются к правилу: старший разряд десятков считывается слева (по строке), а младший разряд единиц сверху (по столбцу).

Правило кодировки по предложенной в учебнике [1] таблице ASCII предполагает другую, непривычную, последовательность действий. Это порождено тем, что нумерация символов в таблице ASCII в десятичной системе счисления проводится по столбцам, а не по строкам, как это принято в большинстве таблиц (например, при записи чисел для решета Эратосфена). Человек психологически настроен считывать информацию по строкам, так же ее и передавать, если только сама форма представленной информации не указывает на другой порядок.



В учебнике [4, с. 368] предложена кодовая таблица ANSI с построчной нумерацией символов и нумерацией в шестнадцатеричной системе счисления (номер строки, затем номер столбца), но в этом случае верхняя половина таблицы содержит символы английской версии, а нижняя половина таблицы содержит национальные символы, псевдографические и математические символы. Символы таблиц в [1], [4] в основном симметричны относительно главной диагонали, хотя и имеются некоторые отклонения от симметрии в связи с перестановкой некоторых столбцов.

Сопоставляя каждому символу из кодовой таблицы ASCII код в шестнадцатеричной системе счисления, школьники должны понимать, что это не набор двух символов — цифр для шифровки, а число, записанное двумя знаками. Каждый знак имеет свое значение в позиционной шестнадцатеричной системе счисления:

$$K_{ASCII} \rightarrow 4B_{16} = 4 \cdot 16^1 + B \cdot 16^0 = 75_{10}$$

Любой двухмерный массив может быть преобразован в одномерный массив (матрицу-строку) либо "построчным прочтением", либо "постолбцовым прочтением".

Авторы учебника, предлагая задание в конце темы, должны понимать, что в тексте должны быть комментарии по пользованию таблицей. Таблицу с двумя входами можно прочесть по-разному. Вызывает сожаление, что девять авторов и рецензент не смогли найти такой недочет. Ошибка, вообще говоря, не очень значительная, но повторенная дважды в школьном учебнике для 7-го класса, дезориентирует ученика.

Имеются недочеты и в выделенном для учащихся, итоговом заключении [1, с. 35] о назначении таблицы ASCII, где сказано, что любой символ в таблице ASCII кодируется с помощью восьми двоичных разрядов или двух шестнадцатеричных разрядов.

В предложенной таблице ASCII в учебнике видны только символы шестнадцатеричной системы и нет кодов двоичной системы. Этот перевод из шестнадцатеричной в двоичную систему требует другой таблицы.

После методической обработки заключение должно принять вид: любой символ по таблице ASCII кодируется с помощью двух шестнадцатеричных разрядов, а затем может быть представлен восьмью двоичными разрядами (1 шестнадцатеричный разряд представляется четырьмя битами). Либо авторы должны были представить также таблицу ASCII с двоичными кодами, что для школьного учебника кажется нереальным. Внешне это замечание кажется незначительным, но непонятно, почему учитель, работающий по этому учебнику, должен заниматься "шлифовкой" выводов вместо авторов.

Литература

1. Информатика. 7—8-е классы. / Под редакцией *Н.В. Макаровой*. СПб.: Питер, 1999.

2. *Симонович С.В., Евсеев Г.А., Алексеев А.Г.* Общая информатика. М.: Аст-Пресс, 1998.

3. *Шафрин Ю.А.* Основы компьютерной технологии. М.: ABF, 1996.

4. *Фигурнов В.Э.* IBM PC для пользователя. Краткий курс. М.: ИНФРА, 1997.

"Информатика" в Калуге

16 ноября в школе № 6 г. Калуги прошла встреча читателей с главным редактором "Информатики" С.Л. Островским. Встреча была организована региональным представителем нашей газеты в г. Калуге Л.К. Матвеевой.

На встрече с читателями "Информатики" в Калуге. Крайняя справа — региональный представитель "Информатики" Л.К. Матвеева.

Мы уже сообщали о том, что "Информатика" регулярно проводит встречи с читателями в различных регионах России. О том, что такие встречи оказывают большое влияние на развитие нашей газеты, мы уже писали многократно, сейчас хочется отметить другое: насколько же увлеченно и продуктивно работают учителя информатики, как интересно то, что делают они и их ученики! И вовсе это не зависит от того, насколько удален от Москвы тот или иной регион. Всюду, куда бы мы ни приезжали, мы встречали заинтересованных, талантливых людей, которые стремятся качественно учить детей в любых, порой самых тяжелых, условиях. И тем крепче наша убежденность, что никто никуда ничего уже не вернет и не повернет. И информатика останется в наших школах самостоятельным предметом, и техника рано или поздно появится, и учебники новые будут написаны, и сети дотянутся до каждой школы. Давайте ради этого вместе работать.

Об итогах выступления команды школьников России на XI международной олимпиаде по информатике

С 9 по 16 октября в г. Анталья (Турция) состоялась XI международная олимпиада школьников по информатике. В ней приняли участие 247 школьников из 65 стран мира. По сравнению с прошлым годом, когда в олимпиаде в Португалии приняло участие рекордное количество участников (267 участников из 66 стран), ситуация практически не изменилась. Хотя и были опасения, что участников будет меньше из-за неблагоприятной сейсмологической обстановки в Турции.

Команду школьников России представляли:

— Мартьянов Владимир, ученик 11-го класса из Нижнего Новгорода, участник и абсолютный чемпион двух последних олимпиад в Кейптауне (Южная Африка) и Сетубале (Португалия);
— Баутин Михаил, ученик 10-го класса из Нижнего Новгорода, второй раз участвовал в международной олимпиаде;
— Бабенко Максим, ученик 11-го класса из Саратова;
— Пастухов Роман, ученик 9-го класса из Оренбурга.

Научным руководителем команды, как и в прошлые годы, был Владимир Михайлович Кирюхин, доцент Московского государственного инженерно-физического института (технического университета).

Олимпиада проводилась в два тура. На каждом туре предлагалось по три задачи, на решение которых отводилось пять часов. Все шесть предложенных задач в обоих турах оценивались одинаково — по 100 баллов. Таким образом, наивысшая оценка за решение всех задач составила 600 баллов и по каждому туру отдельно — 300.

Команда школьников России выступила успешно, получив три золотых и одну бронзовую медали. Золотые медали получили Пастухов Роман (457 очков), Мартьянов Владимир (340) и Бабенко Максим (340). Михаил Баутин получил бронзовую медаль (190 очков). Наивысший результат на этой олимпиаде получил школьник из Китая Hong Chen. Он впервые участвовал в международном соревновании и набрал 480 баллов по результатам двух туров. Наш Роман Пастухов занял вторую строчку в итоговой таблице.

Анализируя итоги олимпиады в целом, следует отметить, что уровень заданий был достаточно высоким. Это видно и по результатам призеров, которые далеки от наивысшей оценки. Задачи были трудными, но решаемыми. Требовалось достаточное количество времени, чтобы определиться с выбором алгоритма и методом решения. Все участники нашей команды до последней минуты во время обоих туров были заняты решением и отладкой предложенных задач. Формулировки задач были достаточно ясными и соответствовали международному уровню.

Следует особо отметить отличную организацию олимпиады. Школьников разместили в комфортабельной гостинице. Рабочие залы для школьников и для руководителей команд были расположены неподалеку. Проверка заданий была организована самым разумным образом. Несмотря на то, что в программах тестирования, подготовленных оргкомитетом, были обнаружены ошибки, организаторы очень быстро их исправили и перепроверили результаты решения задач всех участников.

Неофициальные общекомандные результаты выступления наших школьников таковы:
по очкам — 3 место (1-е — Китай, 2-е — Вьетнам);
по медалям — 2 место (1-е — Вьетнам).

Главный специалист
Департамента общего среднего образования
Л.Е. САМОВОЛЬНОВА

Книжный шкаф

Рубрику "Книжный шкаф" мы планировали ввести и сделать регулярной с Нового года. Изменить это решение заставило появление уникальной книги, которую с полным правом можно назвать бесценной (хотя приходится признать, что и в рублевом исчислении книга имеет немалую цену — в книжных магазинах Москвы она стоит около 120 руб.).

Речь идет о книге "Конкретная математика (основание информатики)" Рональда Грэхема, Дональда Кнута и Орена Паташника. Имя одного из авторов, безусловно, широко известно, а два других являются известными специалистами в области информатики, математики и компьютерных технологий. Эта оригинальная как по содержанию, так и по форме книга представляет собой введение в математику, которая служит основой информатики и анализа алгоритмов. Ее название произошло из слияния двух терминов — КОНтинуальная и дисКРЕТНАЯ математика. Название книги можно понимать и буквально: обучение общим методам ведется на многочисленных конкретных примерах и упражнениях (их более 500) разной степени сложности. Все упражнения снабжены ответами. В предисловии В.Арнольда говорится, что эта книга "раскрывает тайну одного феномена американского образования — как превращать малограмотных школьников в блестящих математиков".

Первое сентября

Индексы: 32024 и 32586

Газета для учителей и родителей. Создатель — Симон Соловейчик. Выходит 2 раза в неделю.

20 ПРЕДМЕТНО-МЕТОДИЧЕСКИХ ПРИЛОЖЕНИЙ

1. **Английский язык** (индексы: 32025 и 32587)
2. **Биология** (32026 и 32588)
3. **Воскресная школа** (32742 и 32743)
4. **География** (32027 и 32589)
5. **Дошкольное образование** (33373 и 33374)
6. **Здоровье детей** (32033 и 32590)
7. **Информатика** (32291 и 32591)
8. **Искусство** (32584 и 32585)
9. **История** (32028 и 32592)
10. **Литература** (32029 и 32593)
11. **Математика** (32030 и 32594)
12. **Начальная школа** (32031 и 32598)
13. **Немецкий язык** (32292 и 32599)
14. **Русский язык** (32383 и 32601)
15. **Управление школой** (32652 и 32653)
16. **Физика** (32032 и 32596)
17. **Французский язык** (33371 и 33372)
18. **Химия** (32034 и 32597)
19. **Школьный психолог** (32898 и 32899)
20. **Спорт в школе** (32384 и 32595)

Газеты распространяются только по подписке во всех регионах России и странах СНГ.

Первое сентября
Объединение педагогических изданий
www.1september.ru

Адрес: 121165, Москва, ул. Киевская, д. 24
Телефон/факс: (095) 249-31-38, 249-31-84
Internet: www.1september.ru
E-mail: gazeta@1september.ru

Новые газеты — по старым ценам

Помощь учителю в его каждодневной работе – при подготовке к проведению занятий и непосредственно на уроках – главная задача предметно-методических приложений к газете «Первое сентября». Читатели приложений еженедельно получают практический материал для работы в классе: поурочные планы, опорные конспекты, подборки вопросов и заданий, разработки учебных тем, варианты контрольных работ и многое другое.

На каждое издание можно подписаться отдельно.

Подписаться на газеты можно в любом почтовом отделении по каталогу «Газеты, журналы» агентства «Роспечать» (первое полугодие 2000 года, стр. 14)

НАШИ НОВИНКИ

С 1 января 2000 года

новые приложения

индексы: 33373 и 33374 **индексы: 33371 и 33372**

новый журнал

«Я иду на урок. Начальная школа. Прописи. Тесты»

Сборник методических пособий для работы с детьми в школе и дома. Индекс подписки — 79384 (каталог агентства «Роспечать», с. 251).

ИЗДАТЕЛЬСКИЙ ДОМ ПИТЕР®
WWW.PITER-PRESS.RU

Внимание!

Комплект создан по инициативе Центра информационных систем обучения Университета педагогического мастерства Санкт-Петербурга. Методика прошла испытания в ряде школ города на специально созданных экспериментальных площадках и опирается на опыт педагогов-практиков.

256 стр., код 1170, 368 стр., код 1169, 304 стр., код 1168, 304 стр., код 1167,

комплект учебников по информатике

под редакцией проф. Н.В. Макаровой (Санкт-Петербург)

Важно!

Курс информатики с 6-го по 11-й класс:

- ✓ опирается на объектно-информационный подход;
- ✓ подробно освещает современные компьютерно-информационные технологии;
- ✓ является наиболее полным из существующих на сегодняшний день учебных пособий по информатике для средних и старших классов;
- ✓ получил гриф «Рекомендовано Комитетом по образованию С.-Петербурга».

Чем хороши эти учебники?

- ✓ Сложные вопросы изложены доступным для детей языком, живо и образно.
- ✓ Книги отлично иллюстрированы.
- ✓ Содержание учебников соответствует проекту образовательного стандарта по информатике, созданного под руководством А.А. Кузнецова и признанного победителем Всероссийского конкурса Министерства образования Российской Федерации в 1997 г.

ПО ВОПРОСАМ ЗАКУПОК ОБРАЩАЙТЕСЬ ПО АДРЕСАМ:

Москва, 1-й Шипковский пер., 3, оф. 207, тел. (095) 235 55 83, факс 234 38 15,
С.-Петербург, ул. Благодатная, 67, тел.: (812) 327 93 37, 294 54 65,
e-mail: sales@piter-press.ru

Вы можете заказать книги наложенным платежом через службу **«КНИГА—ПОЧТОЙ»**. В этом случае книги обойдутся вам дешевле, а почтовые расходы будут оплачиваться при получении. Помните, что почтовые расходы на каждую книгу **уменьшаются** при заказе нескольких экземпляров. Кроме того, при заказе 10 книг цена уменьшается на 5%, 20 книг — на 7%, 30 книг и более — на 10%. Отправьте почтовую карточку с заказом по адресу: Россия, 197198, Санкт-Петербург, а/я 619-ИО; Украина, 310093, Харьков, а/я 9130-ИО; Беларусь, 220012, Минск, а/я 104-ИО. Укажите названия, коды и количество заказываемых книг, ваш индекс и адрес и, если вы ранее уже пользовались услугами службы «Книга—почтой», ваш регистрационный номер.

Фамилия, И., О. _____ *Тел.* _____
Адрес: _____

Заказываю:

цена	*название книги*	*код*	*кол-во*
56 руб.	*Информатика. 6–7 класс*	*1170*	
56 руб.	*Информатика. 7–8 класс*	*1169*	
56 руб.	*Информатика. 9 класс*	*1168*	
56 руб.	*Информатика. 10–11 класс*	*1167*	
15 руб.	*Информатика. Учебно-методическое пособие*	—	

Рожденная в XVIII веке

Окончание. Начало на с. 1

занимались обучением молодежи, в том числе гимназическим, читали "публичные" лекции. С первых лет своего существования академия развернула широкую издательскую деятельность, выпуская как научную, так и научно-популярную литературу. В XVIII веке Академия наук была почти единственным центром, где переводили и печатали древнюю и новую научную и художественную литературу. В середине XVIII века там издавался первый в России научно-популярный журнал "Ежемесячные сочинения". Наряду с отечественными учеными в Петербургской академии в XVIII столетии работало много иностранцев. В числе первых академиков, приехавших в Россию из-за границы по приглашению, были выдающиеся ученые — Леонард Эйлер, для которого Россия стала второй родиной, братья Николай и Даниил Бернулли.

В Петербурге с большим успехом изучались многие важные научные вопросы, в частности, исследовалось строение вещества, развивалось учение о теплоте, о капиллярных явлениях. В Петербургской академии наук впервые была заморожена ртуть. Здесь Михаил Васильевич Ломоносов доказал с помощью опыта закон сохранения вещества. В 1748 году он писал Эйлеру: "Все встречающиеся в природе изменения происходят так, что если к чему-либо нечто прибавилось, то это отнимается у чего-то другого. Так, сколько материи прибавляется какому-либо телу, столько же теряется у другого, сколько часов я затрачиваю на сон, столько же отнимаю у бодрствования, и т.д. Так как это всеобщий закон природы, то он распространяется и на правила движения: тело, которое своим толчком возбуждает другое к движению, столько же теряет от своего движения, сколько сообщает другому, им двинутому" [2].

Ломоносов и его ученики начали развивать физическую химию как особую науку. Русская академическая наука шла в первых рядах в области учения об электричестве и магнетизме. Изготовлялись новые оптические приборы. Из трудов русских академиков мир узнал о богатейшей флоре и фауне нашей страны, о ее географии и этнографии. В первый век своего существования академия сделала большой шаг вперед в изучении истории России.

В XIX веке академия не снижала своего научного уровня. В ее составе из поколения в поколение сменялись выдающиеся химики — В.М. Севергин, Я.Д. Захаров; физики — В.В. Петров, Э.Х. Ленц, Б.С. Якоби; математики — М.В. Остроградский, В.Я. Буняковский, П.Л. Чебышев; выдающиеся биологи, историки, филологи, востоковеды.

В 1839 году академия "обогатилась" Пулковской обсерваторией, получившей мировую известность. Были созданы специальные музеи академии — зоологический, ботанический, минералогический.

В XX столетии развитие отечественной науки идет по многим направлениям, причем во многих областях она заняла ведущие позиции (в чем огромную роль сыграла академия, ставшая в начале 1917 года Российской академией наук (с июля 1925 года по 1991 год — Академия наук СССР). Достаточно сказать, что ряд отечественных ученых в разные годы был удостоен самой почетной международной награды — Нобелевской премии [3]: И.П. Павлов — по физиологии и медицине (1904 г.), И.И. Мечников (совместно с П.Эрлихом) — по физиологии и медицине (1908 г.), Н.Н. Семенов (совместно с С.Н. Хиншелвудом) — по химии (1956 г.), П.А. Черенков, И.М. Франк и И.Е. Тамм — по физике (1958 г.), Л.Д. Ландау — по физике (1962 г.), Н.Г. Басов и А.М. Прохоров — по физике (1964 г.), Л.В. Канторович (совместно с Т.Купмансом) — по экономике (1975 г.), П.Л. Капица — по физике (1978 г.).

Особенно большой интерес в академии проявляется к совсем новым направлениям науки и техники. Так, в разное время, начиная с конца 1940-х годов, целый ряд учреждений системы Академии наук принимал активное участие в создании компьютеров и компьютерных систем [4]. Например, на Украине это Институт электротехники (первая в Европе электронная вычислительная машина МЭСМ) и Институт кибернетики ("Киев", "Проминь", семейство малых электронных вычислительных машин МИР, семейство управляющих вычислительных машин "Днепр"); в Белоруссии — Институт математики; в Москве — Институт точной механики и вычислительной техники (БЭСМ, БЭСМ-2, М-20, БЭСМ-3М, БЭСМ-4, М-220, М-222, БЭСМ-6, семейство вычислительных комплексов "Эльбрус"), Энергетический институт, из которого потом выделился Институт электронных управляющих машин (М-1, М-2, М-3), Вычислительный центр (созданный в 1955 г. "для ведения научной работы в области машинной математики и для предоставления открытого вычислительного обслуживания другим организациям академии"), Институт проблем управления (многопроцессорные системы ПС-2000 и ПС-3000), Институт проблем передачи информации; в Новосибирске — Институт математики (система "Минск-222" — работа велась совместно с конструкторским бюро Минского завода вычислительных машин) и Вычислительный центр (исследования систем с разделением времени).

И сегодня, несмотря на тяжелые экономические условия, работают институты, ведутся исследования, готовятся научные кадры, выпускается литература. Более того, за рубежом увеличивается спрос на российские академические журналы, многие из которых переводятся на иностранные языки. Среди них "Вестник Российской академии наук", "Ядерная физика", "Теплофизика высоких температур", "Лазерная физика", "Журнал неорганической химии", "Журнал физической химии", "Электрохимия", "Неорганические материалы", "Кинетика и катализ", "Физиология растений", "Водные ресурсы", "Микроэлектроника", "Автоматика и телемеханика", "Теория и системы управления", "Проблемы передачи информации", "Программирование" и другие.

Таким образом, переживая вместе со страной трудное время, Российская академия наук, имеющая почти трехсотлетнюю историю, продолжает оставаться флагманом нашей науки.

Литература

1. Большая советская энциклопедия. 2-е изд. Т. 1. М.: Гл. научное издательство "Большая советская энциклопедия", 1949.

2. *Кириллин В.А.* Страницы истории науки и техники. М.: Наука, 1986.

3. *Чолаков В.* Нобелевские премии. Ученые и открытия: Пер. с болг. М.: Мир, 1986.

4. *Частиков А.П.* От калькулятора до суперЭВМ// Вычислительная техника и ее применение. № 1/88.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
Е.Б. Докшицкая
Редакция:
И.Н. Фалина,
Н.Л. Беленькая,
Н.П. Медведева
Дизайн и компьютерная верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

©ИНФОРМАТИКА 1999
выходит четыре раза в месяц
При перепечатке ссылка
на ИНФОРМАТИКУ обязательна,
рукописи не возвращаются

121165, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

Учредитель: ООО "Чистые пруды"
Регистрационный номер 012868
Отпечатано в типографии ОАО ПО "Пресса-1".
125865, ГСП, Москва, ул. "Правды", 24.
Тираж 5500 экз.
Заказ №

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Тел. (095)249 3138, 249 3386. Факс (095)249 3184

Internet: inf@1september.ru
Fidonet: 2:5020/69.32
WWW: http://www.1september.ru

ОБЪЕДИНЕНИЕ ПЕДАГОГИЧЕСКИХ ИЗДАНИЙ «ПЕРВОЕ СЕНТЯБРЯ»

Первое сентября (А.С. Соловейчик), индекс подписки — 32024; **Английский язык** (Е.В. Громушкина), индекс подписки — 32025; **Биология** (Н.Г. Иванова), индекс подписки — 32026; **Воскресная школа** (монах Киприан (Ященко), индекс подписки — 32742; **География** (О.Н. Коротова), индекс подписки — 32027; **Здоровье детей** (А.У. Лекманов), индекс подписки — 32033; **Информатика** (С.Л. Островский), индекс подписки — 32291; **Искусство** (Н.Х. Исмаилова), индекс подписки — 32584; **История** (А.Ю. Головатенко), индекс подписки — 32028; **Литература** (Г.Г. Красухин), индекс подписки — 32029; **Математика** (И.Л. Соловейчик), индекс подписки — 32030; **Начальная школа** (М.В. Соловейчик), индекс подписки — 32031; **Немецкий язык** (М.Д. Бузоева), индекс подписки — 32292; **Русский язык** (Л.А. Гончар), индекс подписки — 32383; **Спорт в школе** (Н.В. Школьникова), индекс подписки — 32384; **Управление школой** (Н.А. Широкова), индекс подписки — 32652; **Физика** (Н.Д. Козлова), индекс подписки — 32032; **Химия** (О.Г. Блохина), индекс подписки — 32034; **Школьный психолог** (М.Н. Сартан), индекс подписки — 32898.