Алгоритмизация

Оглавление

Краткие теоретические сведения

Понятие алгоритма является одним из центральных в программировании. Существует множество его определений, данных известными учеными в разные времена.

Слово «алгоритм» происходит от имени математика Аль Хорезми, разработавшего в IX веке правила (алгоритм) для выполнения четырех арифметических действий над десятичными числами. Впоследствии это слово стало собирательным названием для отдельных правил (последовательности действий) определенного вида, причем не только арифметических. В течение длительного времени его употребляли только математики, обозначая словом «алгоритм» пошаговые правила решения различных задач.

Алгоритм – это строго определенная последовательность действий для некоторого исполнителя, приводящая к конкретному результату за конечное число шагов.

Понятие алгоритма связано с понятием исходных (входных) данных и искомого результата (выходных данных), поскольку назначением любого алгоритма является преобразование исходных данных в искомый результат.

Входные данные задаются до начала работы алгоритма или определяются динамически во время его работы. Число входных данных может быть равным нулю.

В результате выполнения алгоритма получают **выходные данные** – величины, имеющие определенную алгоритмом связь с входными данными.

Свойства алгоритмов

Здесь мы рассмотрим только важные для решения задач ЕГЭ свойства, описанные в большинстве учебников и учебных пособий.

- **Дискретность** – алгоритм представляет собой последовательность отдельных шагов, каждый из которых называется командой. Каждая команда начинает выполняться только после завершения выполнения предыдущей команды.
- **Понятность** – каждая команда алгоритма должна быть понятна тому, кто его исполняет (исполнителю). Необходимо знать, какие команды исполнителю известны, а какие нет. Полный список команд, которые может выполнить исполнитель, называется системой команд исполнителя. При составлении алгоритма для определенного исполнителя можно использовать лишь те команды , которые имеются в его системе команд.
- **Определенность (детерминированность)** – каждая команда алгоритма должна быть точно и однозначно определена, также должно быть однозначно определено, какая команда будет выполняться на следующем шаге. Результат выполнения

команды не должен зависеть от какой-либо дополнительной информации. Алгоритм не должен оставлять места для произвола исполнителя.

- **Конечность (результативность)** – алгоритм всегда должен заканчиваться после выполнения конечного числа шагов, при этом должен быть получен результат.
- **Массовость** – один и тот же алгоритм может применяться к разным наборам входных данных.

Разработка алгоритма – один из основных этапов решения различных задач на компьютере. Общие свойства и закономерности алгоритмов, их разработку и анализ, формальные модели их представления изучает теория алгоритмов

Формальные исполнители

Исполнителями алгоритма могут быть живые существа или технические устройства:

- машины: станки, роботы, бытовые приборы, компьютеры;
- животные: дрессированная собака, тигры и медведи в цирке;
- люди: ученик, токарь, кассир и т.д.

Исполнитель алгоритма способен выполнять команды, заданные алгоритмом. Животные и человек как исполнители отличаются тем, что могут понимать команды в различных вариантах, одни и те же команды выполнять по-разному, отказаться исполнять команду. Поэтому их называют неформальными исполнителями.

Формальный исполнитель может не понимать смысла алгоритма, но все равно правильно выполнить его и получить нужный результат.

Формальный исполнитель алгоритма выполняет в строгой последовательности все предписанные алгоритмом команды, не вникая в содержание поставленной задачи, не задумываясь о ее цели, результате и необходимости. Формальный исполнитель не привносит в алгоритм ничего нового и не отбрасывает никаких действий.

К формальным исполнителям относят автоматы, роботы и другие технические устройства, исполняющие инструкции (алгоритмы). На уроках информатики изучают формальные исполнители «Черепашка», «Паркетчик», «Робот», «Вычислитель» и др.

При записи алгоритмов для формального исполнителя используют команды, входящие в систему команд данного исполнителя.

Система команд формального исполнителя (СКИ) – это совокупность команд, понятных и выполнимых конкретным исполнителем.

Среда - это «место обитания» исполнителя. Например, исполнитель Робот имеет клеточное поле. Расположение стен и закрашенных клеток, положение самого Робота задают конкретное состояние среды.

Среда исполнителя – это совокупность объектов и связей между ними, над которыми данный исполнитель может выполнять команды.

Компьютер тоже можно считать формальным исполнителем, он имеет систему команд и может исполнять алгоритмы, записанные на языке программирования – программы.

Формы записи алгоритма

Основные формы записи алгоритма

– словесная, на естественном языке;

– графическая, например, в виде блок-схем;

– на алгоритмическом языке (псевдокоде).

Запись алгоритма на естественном языке

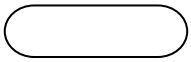
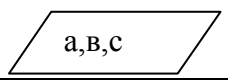
Самым простым способом является словесная запись алгоритма на естественном языке. Если формальным исполнителем алгоритма является человек, то запись алгоритма естественным языком может быть приемлема. Но при изложении алгоритма на естественном языке появляется опасность неточного понимания команд исполнителем, то есть нарушается свойство определенности.

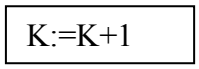
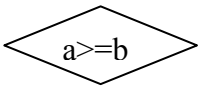
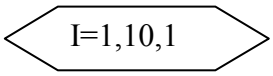
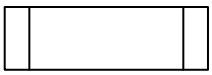
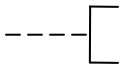
Если речь идет о формальном исполнителе, например, роботе или автомате, то естественного языка он может не понимать вовсе.

Запись алгоритма в виде блок-схем

Для графического представления алгоритмов в виде **блок-схем** используются стандартные обозначения элементов (ГОСТ 19.701 – 90). В таблице приведены основные виды блоков, которые мы будем использовать.

Обозначения элементов блок-схем

Обозначение и пример заполнения	**Пояснение**
	Начало или конец алгоритма
	Блок ввода / вывода данных

	Блок обработки информации
	Логический блок (проверка истинности или ложности логического выражения (условия) и выбор следующего блока)
	Организация циклического процесса – заголовок цикла с параметром. Установка переключателя
	Подпрограмма (состоит из нескольких команд, определенных в другом месте)
	Комментарий

Между собой блоки соединяются линиями переходов. По умолчанию блоки выполняются сверху вниз слева направо. Если последовательность выполнения должна быть другой, используются линии со стрелками.

Запись алгоритма на алгоритмическом языке

Для представления алгоритмов были разработаны специальные алгоритмические языки – разновидности формальных языков. Они позволяют наглядно, просто и в удобной форме отображать содержательный смысл алгоритма и проводить его анализ. По алгоритмам, представленным на алгоритмическом языке, можно составить программу на любом языке программирования.

Алгоритмический язык – это система обозначений, предназначенных для записи алгоритмов.

В алгоритмическом языке используются формальные конструкции, но нет строгих синтаксических правил для записи команд.

В алгоритмическом языке есть служебные слова. Они имеют вполне определенный смысл, и в печатном тексте выделяются жирным шрифтом, а в рукописном - подчеркиванием. Можно встретить различные алгоритмические языки, отличающиеся набором служебных слов и формой записи основных конструкций. Здесь мы будем использовать школьный алгоритмический язык (школьный АЯ), одной из разновидностей которого является язык КуМир.

Основные служебные слова школьного АЯ, приведены в таблице.

Общие	**алг** (алгоритм); **арг** (аргумент); **рез** (результат); **нач** (начало алгоритма); **кон** (конец алгоритма);
Операции ввода и вывода	**ввод; вывод**

Описание типов данных	**цел** (целый); **вещ** (вещественный); **сим** (символьный); **лит** (литерный); **лог** (логический)
Операции целочисленного деления	**div mod**
Логические операции	**и или не**
Логические значения	**да нет**
Организация ветвления	**если**; **то**; **иначе**; **выбор**; **при**; **все**
Организация цикла	**пока**; **для; от; до; нц; кц,**

При описании алгоритмического языка и его конструкций будем использовать следующие соглашения.

- Если текст записан в угловых скобках, например, <команда>, <аргумент>, значит, при составлении алгоритма на это место следует записать конкретную команду, аргумент и т.п.
- Если в записи используется конструкция в квадратных скобках [ ], то эта конструкция необязательна.

В общем виде алгоритм обычно выглядит так:

```
алг <название_алгоритма>(<аргументы>,<результаты>)
нач
        <команды>
кон
```

Здесь <аргументы> – это входные данные, <результаты> – выходные данные алгоритма. Для их описания используются служебные слова **арг** и **рез**, после которых записывается тип значения входных (выходных) данных (**цел**, **вещ**, **лог** и т.д.) и наименования переменных. Например,

```
алг Площадь_круга(арг вещ R, рез вещ S)
алг Задача(арг цел a, b, арг вещ c, рез вещ d)
```

В первом примере входная переменная – радиус окружности – и результат вычислений – площадь круга – имеют вещественный тип.

Во втором примере на вход алгоритму подаются два аргумента целого типа, один аргумент вещественного типа. В результате работы алгоритма получаем одно выходное значение вещественного типа.

Переменные и константы в алгоритмах

Данные в алгоритмах представляются в виде переменных и констант. **Переменная** используется для хранения в процессе выполнения алгоритма входных и выходных данных, результатов промежуточных вычислений и другой информации. Значение

переменной может изменяться в ходе выполнения алгоритма. **Константа** отличается тем, что ее значение не может быть изменено в ходе выполнения алгоритма.

Переменные и константы имеют три основных атрибута: имя, значение и тип. Переменная или константа обозначается именем (идентификатором) в соответствии с правилами языка. Например, в языке КуМир имена переменных могут состоять из символов латинского и русского алфавитов, цифр и знаков @ и _. Имя не может начинаться с цифры и не должно совпадать со служебным словом. Примеры имен: A, B, СЛОВО, степ, чис1 и проч.

Тип данных в алгоритмах определяет

- множество возможных значений;
- совокупность операций, допустимых над этими значениями;

Переменные целочисленных типов можно складывать, умножать, сравнивать и т.д. Переменные литерного (строкового) типа невозможно умножать и делить, их можно сцеплять и сравнивать между собой. Таким образом, для разных типов данных допустимы разные наборы операций.

При разработке алгоритма нужно помнить об ограничениях выполняющего его исполнителя. Если не задумываться об этом, можно получить неожиданные результаты.

Предположим, исполнитель алгоритма – компьютер, целочисленные беззнаковые переменные **A** и **B** занимают в памяти компьютера по одному байту. Выполним операцию сложения **A+B** при разных значениях **A** и **B**.

A	56	$0011\ 1000_2$
B	15	$0000\ 1111_2$
A+B	71	$0100\ 0111_2$

A	250	$1111\ 1010_2$
B	15	$0000\ 1111_2$
A+B	265	$1\ 0000\ 1001_2$

В первом примере будет получен верный результат 56+15=71. Во втором примере единица в старшем бите результата сложения будет отброшена – ей не хватает места в выделенном байте памяти, получим результат 9, а не 265. В таких случаях говорят о **переполнении разрядной сетки**: результат становится больше максимально возможного значения для переменной.

В школьном АЯ используются следующие простые типы:

- **цел** – целый;
- **вещ** – вещественный;
- **сим** – символьный;
- **лит** – литерный;

- **лог** – логический.

Значением символьной (**сим**) переменной или константы является один символ. Обычно для хранения символа в памяти выделяется один байт, поэтому количество допустимых символов равно 256, символы пронумерованы от 0 до 255. В алгоритме и программе символьные константы записываются в апострофах, например, ′a′, ′+′, ′9′.

Значением литерной (**лит**) переменной или константы является последовательность символов. Иногда ее называют строкой или цепочкой символов. Количество символов в строке называется длиной строки.

Для указания типа переменной ее необходимо **объявить, или описать** в программе. В школьном АЯ объявление переменных производится в начале программы. Несколько переменных одного типа могут быть объявлены после имени типа через запятую. Например,

```
цел A, B
лог BOOL
```

Более сложные структурированные типы рассмотрим в следующей теме.

Команда присваивания

Для записи в переменную некоторого значения используется команда присваивания. В школьном АЯ команда присваивания имеет вид

<имя_переменной>:=<выражение>

В школьном АЯ в левой части команды присваивания может находиться только имя переменной. В правой части может быть записана константа, переменная или выражение. (Константа и переменная являются частным случаем выражения).

Команда присваивания выполняется справа налево, то есть выражение справа от знака «:=» вычисляется, и результат сохраняется в переменной, имя которой указано слева.

В команде присваивания имя переменной, записанной слева, может встретиться и в правой части, например,

A:= A + 7

В этой команде переменной A присваивается результат вычисления выражения A+7. Такая запись возможна именно потому, что переменные хранятся в памяти компьютера, а вычисления проводятся процессором. Значение переменной **A** суммируется с константой **7**, полученный результат записывается в переменную **A.**

Выражение – это формула, по которой вычисляется значение. Выражение может состоять из операндов, знаков операций и круглых скобок. В алгоритмических языках выражение записывается в строку.

Операндами являются константы, переменные и обращения к функциям.

Для обозначения операций в школьном АЯ используют символы

`+, –, *, /, div, mod, не, и, или, >, <, >=, <=, =, <>`

`div` – операция целочисленного деления, результатом является целая часть частного (ближайшее меньшее целое);

`mod` – операция получения остатка целочисленного деления;

`не, и, или` – логические операции, выполняются в соответствии с таблицами истинности;

`>, <, >=, <=, =, <>` – операции сравнения, или операции отношения.

Довольно часто школьники ошибаются при использовании операций целочисленного деления `div` и `mod` – вместо целочисленного остатка от деления используют десятичную часть частного. Справедливо равенство

$x = y \cdot \mathtt{div}(x, y) + \mathtt{mod}(x, y)$.

Приведем примеры выполнения операций целочисленного деления

x	***y***	**div(*x*, *y*)**	**mod(*x*, *y*)**
40	17	2	6
15	17	0	15

Важно отметить, что в разных системах программирования операции целочисленного деления при отрицательных аргументах (одном или двух) дают разный результат, как показано в таблице

x	***y***	**КуМир**		**ЭТ Excel**		**VisualBasic**	
		div(*x*, *y*)	**mod(*x*, *y*)**	**div(*x*, *y*)**	**mod(*x*, *y*)**	**a\b**	**a mod b**
–40	17	–3	11	–3	11	-2	-6
40	–17	–3	–11	–3	–11	-2	6
–40	–17	2	–6	2	–6	2	-6
15	–17	–1	–2	–1	–2	0	15
–15	–17	0	–15	0	–15	0	-15

Следует с осторожностью применять целочисленное деление при отрицательных значениях аргументов.

Операции в выражении выполняются в соответствии с приоритетами, определенными правилами языка. В школьном АЯ приняты следующие приоритеты выполнения операций

1	не
2	*, /, div, mod, и
3	+, –, или
4	>, <, >=, <=, =, <>

В первую очередь выполняются операции с меньшим приоритетом. Операции с одинаковым приоритетом выполняются слева направо, если порядок выполнения не указан явно круглыми скобками. Операции в скобках выполняются в первую очередь, с учетом приоритетов. Вычисление значения выражения с вложенными скобками начинается с внутренних скобок.

Примеры вычислений выражений

Команда присваивания	**Вычисление выражения**
Y:=8+3*5	8+3*5=8+15=23
A:=12/2*3	12/2*3=6*3=18
B:=32/4/2	32/4/2=8/2=4
C:=35/(3+12/3)	35/(3+12/3)=35/(3+4)=35/7=5
D:=((1+2)*5+2)*5+3	((1+2)*5+2)*5+3=(3*5+2)*5+3=17*5+3=88

Команды ввода и вывода

Для ввода и вывода данных в школьном АЯ используются команды

ввод <список_ввода>

вывод <список_вывода>

Списки ввода и вывода состоят из элементов, которые перечисляются через запятую.

В списке **ввода** могут присутствовать только имена переменных. При выполнении команды **ввод** алгоритм получает данные, которые записываются в соответствующие переменные.

В списке **вывода** могут быть перечислены константы, имена переменных и выражения.

Текстовые константы записываются в списке вывода в кавычках, а выводятся (например, на экран или принтер) без кавычек. Числовые константы выводятся без изменений. Если в списке вывода указана переменная, то выводится ее значение. Если в списке вывода указано выражение, выводится результат его вычисления.

Пример записи алгоритма

```
алг гипотенуза (арг вещ a, b, рез вещ c)
```

```
нач
      ввод a, b
      c:=sqrt(a*a+b*b)
      вывод "катеты ",a,",   ",b,"    гипотенуза ",c
кон
```

Если при выполнении команды **ввод** ввести, например, числа 3 и 4, они сохранятся соответственно в переменных `a` и `b`. При выполнении команды **вывод** будет выведено следующее:

```
катеты 3,   4    гипотенуза 5
```

Если вместо переменной `c` в список вывода записать `sqrt(a*a+b*b)`, будет сформирован тот же вывод.

Как бы ни был записан алгоритм: на языке формального исполнителя, на естественном языке или при помощи блок-схемы – он составляется с использованием базовых (основных) алгоритмических структур (конструкций).

Базовые алгоритмические структуры

Выделяют три базовые алгоритмические структуры (конструкции) – линейная (следование), ветвление и цикл – из которых можно построить любой алгоритм. **Каждая алгоритмическая структура имеет одну точку входа и одну точку выхода.**

Будем записывать структуры в виде блок-схем и школьного алгоритмического языка.

1. **Линейная структура** является самой простой организацией алгоритмов – команды выполняются последовательно одна за другой.

Блок-схема	Школьный АЯ
<Команда 1> <Команда 2>	… <Команда 1> <Команда 2> …

Рис. 1. Линейная структура

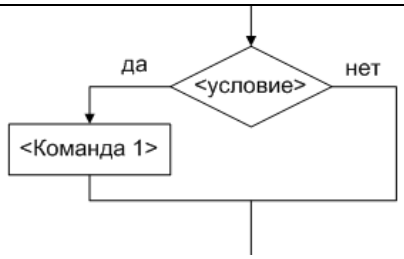
2. **Структура «ветвление»**. Решение некоторых задач требует различных действий в зависимости от выполнения некоторых условий. В таких случаях говорят о ветвлении алгоритма.

Для реализации структуры «ветвление» используются две структурированные команды[1] школьного АЯ – **если** и **выбор,** каждая из которых может быть полной и неполной. Полная команда **если** имеет вид **если…то…иначе…все,** неполная – **если…то…все.** Полная команда **выбор** имеет вид **выбор…при…иначе…все**.

В блок-схемах и школьном АЯ <условие> – это логическое выражение, результатом которого может быть одно из двух возможных значений – истина или ложь. В школьном АЯ эти значения записывают как `да` и `нет`. В языках программирования часто используются значения True и False. В компьютере эти значения хранятся как 1 и 0.

Полная форма команды **если** определяет две ветви команд: первая выполняется в случае истинности условия; вторая – в случае его ложности. В каждой ветви может выполняться не одна команда, а серия команд.

Неполная форма команды **если** предполагает, что при истинности условия выполняется команда1 (серия команд), в противном случае никакие действия не выполняются.

Блок-схема	**Школьный АЯ**
Неполная форма команды если	
	**если** <условие> **то** <Команда1> **все**
Полная форма команды если	

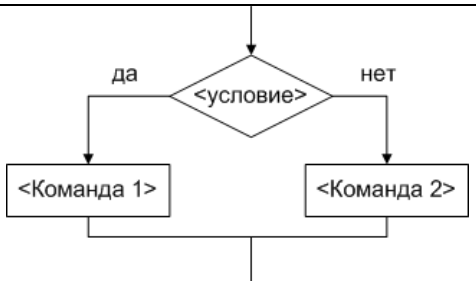
	**если** <условие>
	  то <Команда1>
	  иначе
	  <Команда2>
	все

Рис. 2. Структура ветвление. Команда если

[1] структурированная команда включает в себя несколько команд

Команда **выбор** используется для организации множественного ветвления.

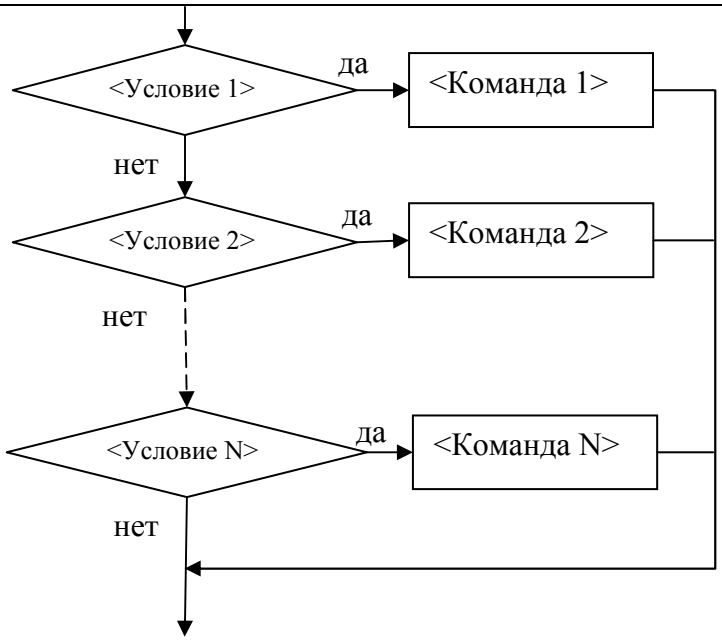
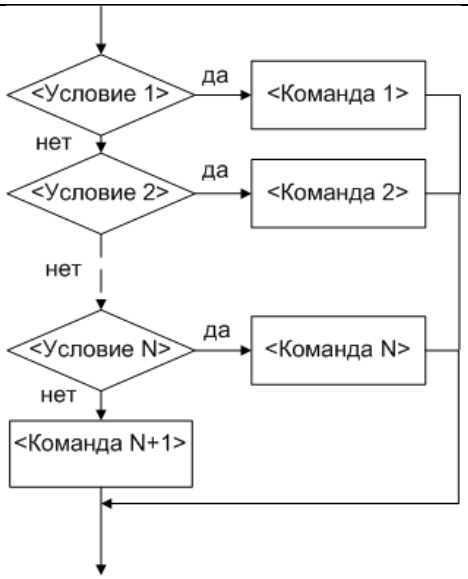
Блок-схема	Школьный АЯ
Неполная форма команды выбор	
	**выбор** **при** <условие1>:<Команда1> **при** <условие2>:<Команда2> **…** **при** <условиеN>:<КомандаN> **все**
Полная форма команды выбор	
	**выбор** **при** <условие1>:<Команда1> **при** <условие2>:<Команда2> **…** **при** <условиеN>:<КомандаN> **иначе** <КомандаN+1> **все**

Рис. 3. Структура ветвление. Команда выбор

3. **Циклическая структура (цикл)** обеспечивает многократное выполнение одних и тех же команд. Существует несколько разновидностей циклических структур. Рассмотрим две: циклическую структуру с предусловием, и циклическую структуру с параметром.

Любая циклическая структура состоит из двух частей – заголовка и тела цикла. Набор команд, повторяющихся при выполнении цикла, называют **телом цикла**. **Заголовок** определяет количество повторений тела цикла.

В цикле с предусловием заголовок имеет вид **пока** <условие>**.** Выполнение тела цикла (<Команды>) будет повторяться до тех пор, пока условие истинно.

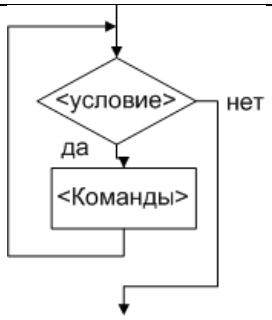
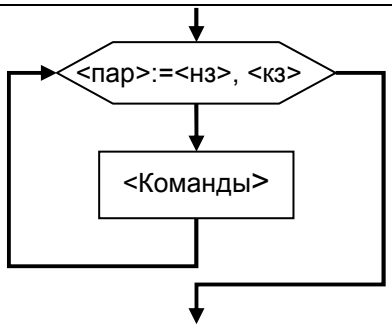
<table>
<tr><th>Блок-схема</th><th>Школьный АЯ</th></tr>
<tr><td colspan="2">Цикл с предусловием</td></tr>
<tr><td> 
</td><td>нц пока <условие>
  <Команды>
кц</td></tr>
<tr><td colspan="2">Цикл с параметром</td></tr>
<tr><td> 
</td><td>нц для <пар> от <нз> до <кз>
  <Команды>
кц</td></tr>
</table>

Рис. 4. Циклические структуры

В цикле с параметром заголовок цикла имеет вид:

для <пар> от <нз> до <кз>

Здесь

<пар> - параметр цикла, это может быть переменная целого типа;

<нз> – начальное значение параметра цикла,

<кз> – конечное значение параметра цикла.

Цикл с параметром выполняется следующим образом:

1. параметр цикла принимает начальное значение;
2. если параметр цикла не превышает конечного значения, выполняется тело цикла, иначе – выход из цикла, переход к следующей команде алгоритма.
3. параметр цикла изменяется на единицу или на заданный шаг;
4. переход к пункту 2.

Тела цикла с параметром при шаге 1 выполняется (<кз>-<нз>+1) раз. Если конечное значение меньше начального значения при шаге больше нуля, тело цикла не выполняется ни разу.

Выполнение алгоритмов. Трассировочные таблицы

Наиболее простым способом отслеживания действий, предписанных алгоритмом, является его пошаговое исполнение.

Для того чтобы наглядно представлять значения переменных, изменяющихся при выполнении алгоритма в сложных алгоритмических структурах, например, циклах, составляют **трассировочные таблицы,** которые называют также **таблицами значений**.

Трассировочные таблицы используются для анализа свойств алгоритма и проверки его соответствия решаемой задаче.

Трассировочные таблицы могут быть двух видов и отражать:

1) пошаговое выполнения каждого действия с записью результата в строку таблицы;
2) выполнение группы действий с записью результатов в одну строку для всей группы.

Рассмотрим составление трассировочной таблицы первого вида. В заголовке таблицы помещают имена всех переменных, используемых в алгоритме. В отдельном столбце записывают команды и логические выражения (условия), которые выполняются. Каждая строка таблицы соответствует одному шагу алгоритма, при котором изменяются значения переменных или выражений.

Как правило, в таблицу заносят только те значения, которые получены на очередном шаге. Если значение какой-либо переменной не меняется, его в таблицу не записывают, чтобы не загромождать ее. Подразумевается, что в переменной находится значение, записанное в ближайшей сверху строке соответствующего столбца.

Составление трассировочных таблиц для линейных структур

Попытаемся решить задачу: поменять значения двух переменных A и B. Запишем команды и результаты их выполнения в трассировочную таблицу.

Команда	A	B
A:=3	3	
B:=5		5
A:=B	5	
B:=A		5
Результат	**5**	**5**

Как видно из таблицы, задача не решена, значение 3, находившееся в переменной A, утеряно. Надо его сохранить, прежде чем присваивать переменной A значение переменной B. Введем дополнительную переменную C. Такую переменную, предназначенную для временного хранения значения, называют буферной.

Команда	A	B	C
A:=3	3		
B:=5		5	
C:=A			3
A:=B	5		
B:=C		3	
Результат	**5**	**3**	

Получен требуемый результат.

Решим эту же задачу без использования дополнительной переменной.

```
A:=3
B:=5
A:=A+B
B:=A-B
A:=A-B
```

№ шага	Команда	Вычисление правой части команды присваивания	A	B
1	A:=3	3	3	
2	B:=5	5		5
3	A:=A+B	A+B=3+5=8	8	
4	B:=A-B	A-B=8-5=3		3
5	A:=A-B	A-B=8-3=5	5	
Результат			**5**	**3**

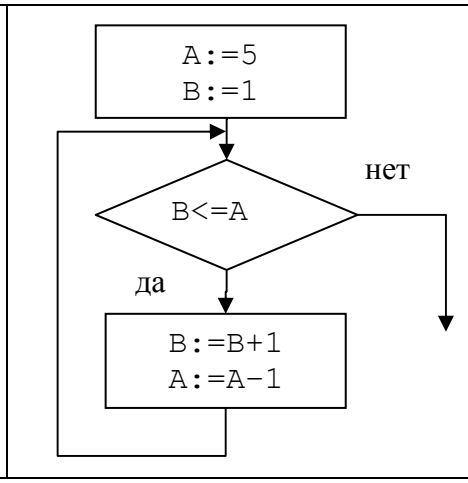
Задача решена. Из этого несложного примера видно, что для одной задачи можно составить несколько алгоритмов, которые отличаются количеством используемых переменных, команд и операций.

Составление трассировочных таблиц для циклических конструкций.

Задан фрагмент алгоритма, содержащего базовую структуру цикл с предусловием.

Школьный АЯ	Блок-схема

```
A:=5
B:=1
Пока B<=A
нц
      B:=B+1
      A:=A-1
кц
```



Определим,

- сколько раз выполняется тело цикла,
- значения переменных **A** и **B** после его завершения:

Построим трассировочную таблицу первого вида.

№	Команда или условие (логическое выражение)	Вычисление правой части команды присваивания или условия (логического выражения)	A	B
1	A:=5	5	5	
2	B:=1	1		1
3	B<=A	(1<=5)=да		
4	B:=B+1	1+1=2		2
5	A:=A-1	5-1=4	4	
6	B<=A	(2<=4)=да		
7	B:=B+1	2+1=3		3
8	A:=A-1	4-1=3	3	
9	B<=A	(3<=3)=да		
10	B:=B+1	3+1=4		4
11	A:=A-1	3-1=2	2	
12	B<=A	(4<=2)=нет		
Результат			**2**	**4**

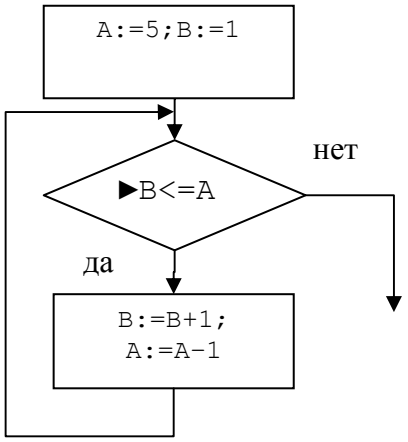
Из таблицы видно, что тело цикла выполнилось трижды (строки 4-5, 7-8, 10-11), переменные приняли значения A=2, B=4.

Трассировочная таблица первого вида может получиться довольно длинной. Более компактный вид имеют трассировочные таблицы второго вида, но для их построения требуется некоторый опыт трассировки алгоритмов, навык строгого соблюдения последовательности действий.

При составлении таблиц второго вида используются **контрольные точки**.

Контрольную точку устанавливают на выбранной пользователем строке алгоритма. Выполнение алгоритма продолжается до контрольной точки и приостанавливается на отмеченной строке. В трассировочную таблицу записываются текущие значения переменных и выражений. Если значение переменной не изменилось, его можно не вносить в таблицу. При необходимости можно поставить несколько контрольных точек.

Поставим **контрольную точку** на заголовке цикла **пока** B <= A.

Школьный АЯ	Блок-схема
A:=5 B:=1 ► **пока** B<=A **нц** B:=B+1 A:=A−1 **Кц**	

Строки составленной таблицы отражают состояние переменных и выражений программы в контрольных точках.

№ кт	B<=A	A	B
1	да	5	1
2	да	4	2
3	да	3	3
4	нет	2	4
Результат		**2**	**4**

Из таблицы видно, что цикл выполнится три раза (выражение, определяющее вход в цикл, три раза принимает значение да), и значения переменных равны **A**=2, **B**=4.

Пример разработки алгоритма

Постановка задачи: найти наибольший общий делитель D двух натуральных чисел X и Y.

Входные данные: натуральные числа X и Y.

Выходные данные: натуральное число D.

Методы решения.

Метод 1 – алгоритм Евклида. Пусть A и B одновременно не равные нулю целые неотрицательные числа, и пусть $A >= B$. Тогда если $B = 0$, то НОД(A, B) = A, а если $B \neq 0$, то для чисел A, B и R, где R - остаток от деления A на B выполняется равенство НОД(A, B) = НОД(B, R). Алгоритм выполняется по шагам следующим образом:

Шаг 1. Задать числа X и Y.

Шаг 2. Присвоить большее из этих чисел переменной A, меньшее – переменной B.

Шаг 3. Пока $B \neq 0$ выполнить

Найти остаток R от деления A на B;

Заменить A на B, B на R

Шаг 4. Присвоить переменной D значение переменной A.

Шаг 5. Останов.

Метод 2 – является также реализацией алгоритма Евклида. Операция определения остатка от деления заменяется последовательным вычитанием.

Шаг 1. Задать числа X и Y.

Шаг 2. Присвоить A:=X, B:=Y.

Шаг 3. Пока $A \neq B$ выполнить

если $A>B$ то присвоить A:=A–B

иначе присвоить B:=B–A

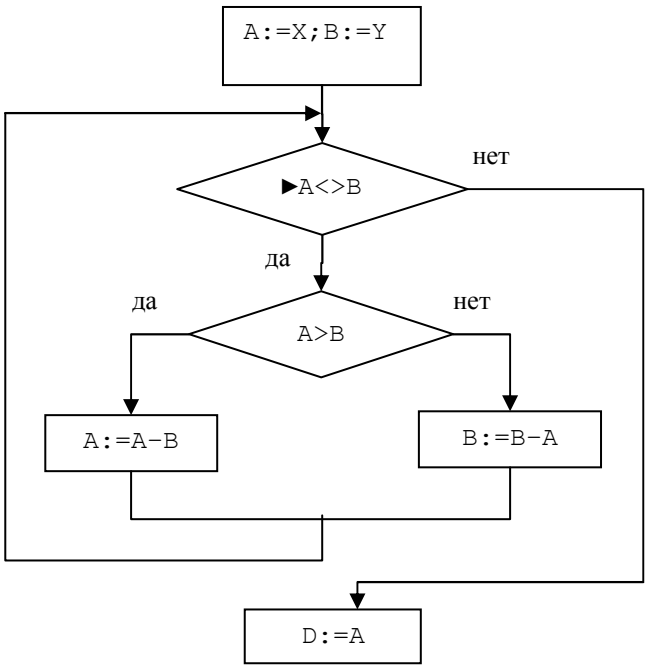
Шаг 4. Присвоить переменной D значение переменной A.

Шаг 5. Останов.

Составление алгоритма решения. Составим алгоритм решения задачи в виде блок-схемы и на школьном АЯ для третьего метода.

Школьный АЯ	**Блок-схема**

```
ввод X,Y
A:=X
B:=Y
►пока A<>B
      если A>B
      то A:=A-B
      иначе B:=B-A
      все
кц
D:=A
```



Проверка работоспособности программы, включающая ее тестирование

Пусть *X*=48, *Y*=18, НОД(48,18)=6.

Составим трассировочную таблицу первого вида и найдем наибольший общий делитель.

№	Команда или условие (логическое выражение)	Вычисление правой части команды присваивания или условия (логического выражения)	A	B
1	A:=X	48	48	
2	B:=Y	18		18
3	A<>B	(48<>18)=да		
4	A>B	(48>18)=да		
5	A:=A-B	48-18=30	30	
6	A<>B	(30<>18)=да		
7	A>B	(30>18)=да		
8	A:=A-B	30-18=12	12	
9	A<>B	(12<>18)=да		
10	A>B	(12>18)=нет		
11	B:=B-A	18-12=6		6
12	A<>B	(12<>6)=да		
13	A>B	(12>6)=да		
14	A:=A-B	12-6=6	6	
15	A<>B	(6<>6)=нет		
16	D:=A	6		
Результат: D = 6				

Составим трассировочную таблицу второго вида для этого примера. Положение контрольной точки показано в блок-схеме и в записи алгоритма на школьном АЯ.

№ кт	A<>B	A>B	A	B	D
1	да		48	18	
2	да	да	30		
3	да	да	12		
4	да	нет		6	
5	нет	да	6		
6					6

Примеры решения заданий

Формальный исполнитель РОБОТ

В среде формального исполнителя алгоритмы, составленные с использованием СКИ, исполняются по шагам.

Пример 1 задания с кратким ответом

Исполнитель Робот ходит по клеткам клетчатой доски, переходя по одной из команд **вверх, вниз, вправо, влево** в соседнюю клетку в указанном направлении. Робот выполнил следующую программу:

влево, вверх, вниз, влево, вверх, вниз, влево, влево, вверх, вправо.

Укажите наименьшее возможное число команд в программе, приводящей Робота из той же начальной клетки в ту же конечную.

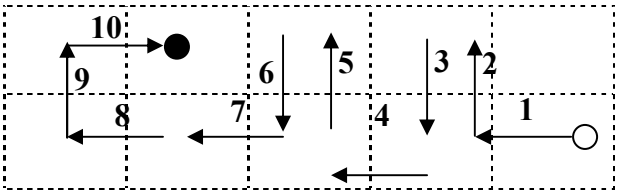
Решение:

Первый способ. Пронумеруем шаги алгоритма:

1 влево, 2 вверх, 3 вниз, 4 влево, 5 вверх, 6 вниз, 7 влево, 8 влево, 9 вверх, 10 вправо.

Пометим пустым кружком начальную клетку, закрашенным кружком – конечную.

Траектория движения робота, заданная алгоритмом:



Из рисунка видно, что требуется найти алгоритм, который переместит робота на три клетки влево и поднимет на одну клетку вверх. Таких алгоритмов четыре:

влево, **влево,** **влево,** **вверх.**	**вверх,** **влево,** **влево,** **влево.**	**влево,** **вверх,** **влево,** **влево.**	**влево,** **влево,** **вверх,** **влево**

Наименьшее количество команд – четыре.

Второй способ решения – «взаимное сокращение» команд. Будем вычеркивать из заданной последовательности пары взаимно обратных команд (**вверх-вниз** или **вправо-влево**).

влево, ~~вверх~~, ~~вниз~~, влево, ~~вверх~~, ~~вниз~~, влево, ~~влево~~, вверх, ~~вправо~~.

Осталось четыре команды: **влево, влево, влево, вверх.**

Ответ: 4

Пример 2 задания с выбором одного ответа

Система команд исполнителя РОБОТ, «живущего» в клетках прямоугольного лабиринта на плоскости:

вверх	**вниз**	**влево**	**вправо**

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре условия позволяют проверить отсутствие преград у каждой из сторон той клетки, где находится РОБОТ:

сверху свободно	**снизу свободно**	**справа свободно**	**слева свободно**

В цикле

ПОКА <*условие*> <*команда*>

команда выполняется, пока условие истинно, иначе происходит переход на следующую строку программы.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенный алгоритм, РОБОТ остановится в той же клетке, с которой он начал выполнение программы?

НАЧАЛО

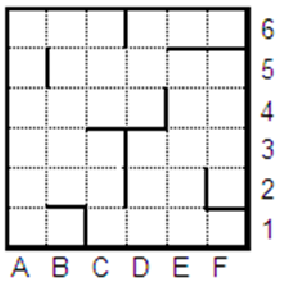
ПОКА < сверху свободно > вверх

ПОКА < слева свободно > влево

ПОКА < снизу свободно > вниз

ПОКА < справа свободно > вправо

КОНЕЦ



1) 1 2) 2 3) 3 4) 4

Решение:

РОБОТ, выполняющий такую программу, **не разрушится**, т.к. появление стены перед ним приводит к изменению направления его движения.

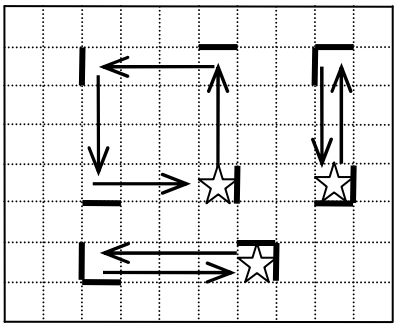
На первый взгляд, для решения задачи может потребоваться проверка всех клеток, то есть необходимо выполнить алгоритм для тридцати шести клеток (полный перебор). Однако это процесс трудоемкий, требующий внимания и времени.

Ограничим перебор. Для этого определим:

1. вид траектории движения РОБОТа, описанной заданным алгоритмом;
2. клетки, в которых РОБОТ может остановиться.

После чего проверим выполнение алгоритма только для клеток, в которых РОБОТ может остановиться.

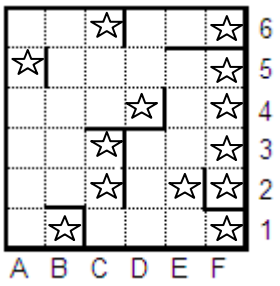
Шаг 1. Траектория движения РОБОТа – прямоугольник, ограниченный стенами, препятствующими движению, в предельных случаях – две прямые, как показано на рисунке:



Шаг 2. РОБОТ заканчивает движение при выполнении команды

ПОКА < справа свободно > вправо

Следовательно, он может остановиться только в клетках, имеющих правую стенку. Таких клеток на поле тринадцать, пометим их звездочками.



Шаг 3. Проверим, имеются ли траектории движения из этих клеток, определенные на первом шаге.

Начав движение из клеток A5, B1, C2, C3 и C6, РОБОТ остановится в клетке B1.

Начав движение из клеток D4, F6, РОБОТ остановится в клетке D4.

Начав движение из клеток F2, F3, F4, F5, E2, РОБОТ остановится в клетке C2.

Начав движение из клетки F1, РОБОТ остановится в клетке F1.

Таким образом, три клетки B1, D4 и F1 отвечают требованию условия задания.

Ответ: № 3

Пример 3 задания с выбором одного ответа

Система команд исполнителя РОБОТ, «живущего» в клетках прямоугольного лабиринта на плоскости:

вверх	**вниз**	**влево**	**вправо**

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре условия позволяют проверить отсутствие преград у каждой из сторон той клетки, где находится РОБОТ:

сверху свободно	**снизу свободно**	**справа свободно**	**слева свободно**

В цикле

ПОКА <*условие*> <*команда*>

команда выполняется, пока условие истинно, иначе происходит переход на следующую строку программы.

Если РОБОТ начнет движение в сторону стены, то он разрушится, и выполнение программы прервется.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал выполнение программы?

НАЧАЛО

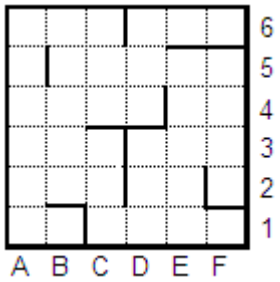
ПОКА < слева свободно > вверх

ПОКА < сверху свободно > вправо

ПОКА < справа свободно > вниз

ПОКА < снизу свободно > влево

КОНЕЦ



1) 1 2) 2 3) 3 4) 4

Решение:

РОБОТ, выполняющий такую программу, **может разрушиться**, если перед ним по ходу движения появится стена.

Для решения этой задачи также ограничим перебор, определим:

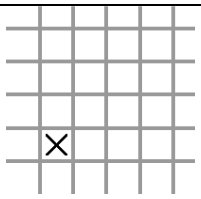
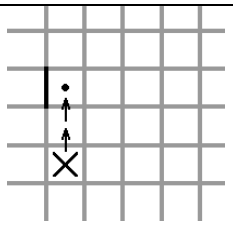
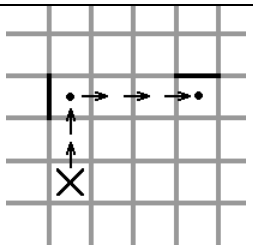
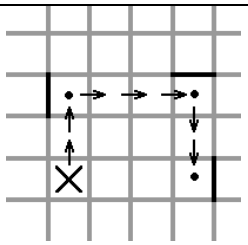
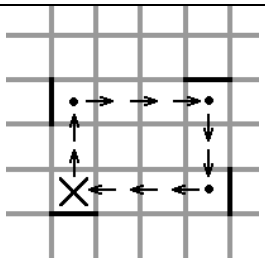
1. траекторию движения РОБОТа, описанную заданным алгоритмом;
2. клетки, в которых РОБОТ может остановиться,

и проверим выполнение алгоритма только для клеток, в которых РОБОТ может остановиться.

Будем называть успешным движение РОБОТа, при котором он не разрушится.

Шаг 1.

Проследим траекторию движения робота при безошибочном выполнении программы.

На клетчатом листе отметим крестом начальное положение Робота. Каждый шаг робота будем отмечать стрелкой из клетки в клетку.	
ПОКА < **слева свободно** > **вверх** Робот будет двигаться вверх по плоскости до тех пор, пока не встретит стену слева от себя. Как только слева от робота появилась стена, выполнение цикла, перемещающего его вверх, прерывается. Клетку остановки здесь и далее будем отмечать точкой.	
ПОКА < **сверху свободно** > **вправо** Робот будет двигаться вправо до тех пор, пока не стена не появится сверху от него. Появление стены, прерывает выполнение текущего цикла.	
ПОКА < **справа свободно** > **вниз** Робот будет двигаться вниз до тех пор, пока не стена не появится справа от него. Появление стены, прерывает выполнение текущего цикла.	
ПОКА < **снизу свободно** > **влево** Робот будет двигаться влево до тех пор, пока не стена не появится снизу от него. Появление стены, прерывает выполнение текущего цикла.	

Траектория движения робота в случае успеха – прямоугольник. Движение начинается снизу вверх, РОБОТ движется по часовой стрелке. «Тормоз» РОБОТа всегда слева по ходу его движения, т.е. если слева по ходу движения появляется стена, РОБОТ меняет направление движения.

Заметим, что в предельных случаях, стены могут смыкаться, образуя квадрат или очерчивать «уголки» прямоугольника, например, как это показано на рисунке 5.

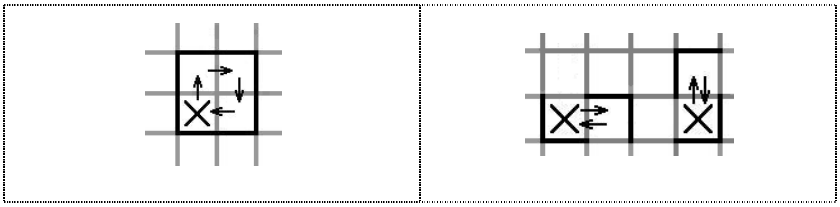


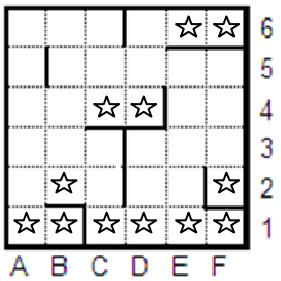
Рис. 5

Если РОБОТ начинает движение в клетках с тремя стенками (можно называть их стаканами), то он разрушится

при движении вправо
при движении вниз
при движении влево
при движении вверх

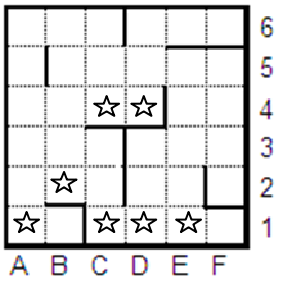
Заметим, что РОБОТ, выполняющий программу из предыдущего примера, выйдет из «стакана» целым и невредимым, сделав несколько поворотов (может, только голова закружится).

Шаг 2. Известно, что РОБОТ остановится, если снизу есть стена. Следовательно, будем рассматривать только клетки, в которых снизу есть стена. Они помечены на рисунке звездочками.



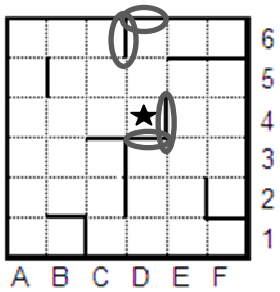
Шаг 3. Проверим помеченные клетки.

РОБОТ разрушится, если начнет движение из клеток B1, F1, F2, F6 (они ограничены тремя стенками). Если РОБОТ начнет движение из E6, он сразу разрушится. Удалим пометки в этих клетках.



Далее, начав движение:

- из A1 – остановится в B1;
- из C1 – остановится в F1;
- из D1 – разрушится при движении вправо из E2 (D1–D2–E2);
- из E1 – разрушится при движении вправо из F4 (E1–E2–E3–E4–F4);
- из B2 – разрушится при движении влево из D2 (B2–B5–E5–E2–D2);
- из C4 – разрушится при движении вверх из C6 (C4–C5–C6);
- из D4 – остановится в D4 (D4–D5–D6–D5–D4).



D4 – единственная клетка, удовлетворяющая условию задания.

Для нахождения количества «успешных» стартовых клеток можно найти количество стен, расставленных должным образом. В условиях нашей задачи такой случай – один:

Ответ: № 1

Формальный исполнитель Вычислитель

Пример 4 задания с кратким ответом

Исполнитель «Вычислитель» имеет следующую систему пронумерованных команд:

1. умножь на два

2. прибавь единицу

Первая умножает число на дисплее на два, а вторая прибавляет к числу на экране единицу. Алгоритм, преобразующий число 3 в число 26, записывается в виде последовательности команд **1121**, что соответствует:

1. умножь на два	3*2=6
1. умножь на два	6*2=12
2. прибавь единицу	12+1=13
1. умножь на два	13*2=26

Запишите порядок команд алгоритма, преобразующего число 3 в число 21, содержащего не более пяти команд, указывая лишь номера команд.

Решение.

Задача преобразования числа 3 в число 21 имеет не одно решение. Например, можно на каждом шаге добавлять по единице, тогда надо будет выполнить 18 команд **2. прибавь единицу**. Можно семь раз прибавить единицу, полученное число 10 умножить на два и прибавить единицу. Существуют и другие решения.

Для наглядного представления вариантов решений подобной задачи удобно использовать дерево.

Каждая вершина (узел) дерева может иметь потомков, с которыми она связана ветвями. По отношению к потомкам такой узел называется предком. Узел, не имеющий предка, называется **корнем**, или **вершиной дерева**. Узлы, не имеющие потомков, называют **листьями**. Ветви дерева могут соответствовать выполнению какой-либо операции, приводящей к результату, который сохраняется в узле-потомке.

Для поиска вариантов решений будем строить двоичное дерево, потому что система команд Вычислителя содержит только две команды. Если N – это номер уровня, то количество узлов на этом уровне двоичного дерева равно 2^N. Например, на пятом уровне двоичного дерева может находиться 32 узла.

Левые ветви дерева будут соответствовать выполнению команды **1. умножь на два**, правые – выполнению команды **2. прибавь единицу**. Пометим некоторые ветви для наглядности. В узлах дерева будем записывать полученный после выполнения соответствующей команды результат. В корень дерева запишем заданное число 3.

По условию задачи можно использовать не более пяти команд, поэтому дерево должно иметь не более пяти уровней. Дерево будем строить по уровням (в ширину) – от каждого узла верхнего уровня порождаются два узла следующего уровня.

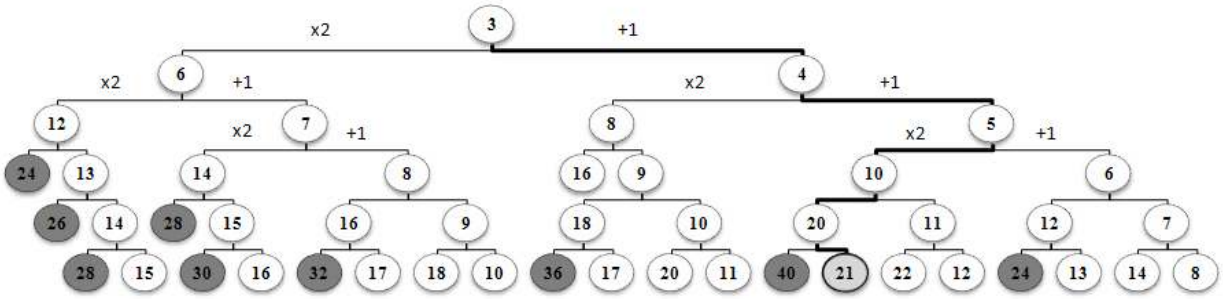


Рис. 6. Фрагмент двоичного дерева поиска вариантов решений

В некоторых узлах значение превышает заданное число 21, построение дерева от этих узлов прекращается. Результатом является путь от вершины дерева к узлу с заданным числом, состоящий из последовательности команд формального исполнителя. На рисунке этот путь выделен жирной линией. Последовательность ветвей решения – правая, правая, левая, левая, правая, что соответствует командам 22112.

Описанный метод решения приводит к верному результату, но является весьма трудоемким.

Решим обратную задачу: получить из числа 21 число 3. Команды, которые мы будем применять, также должны быть обратными командам исполнителя, заданным выше в условии задачи:

1. раздели на два

2. вычти единицу

Операция **раздели на два** может выполняться только для чисел, кратных двойке, иначе эта команда не будет обратной исходной команде **умножь на два**. Таким образом, не из каждого узла можно будет построить две ветви, следовательно, количество узлов дерева уменьшится.

Покажем, как можно использовать построение дерева для поиска решения обратной задачи. Дерево строим, как и в прямой задаче, до пятого уровня. В корень дерева запишем число 21. Построение дерева прекратим, как только на каком-либо уровне получим узел со значением 3. Отметим, что у некоторых узлов будет только один потомок, так как не ко всем числам можно применить операцию **раздели на два**. Правой ветви соответствует операция **раздели на два,** левой ветви и ветви, направленной вниз – операция **вычти единицу.** Решение выделено жирной линией.

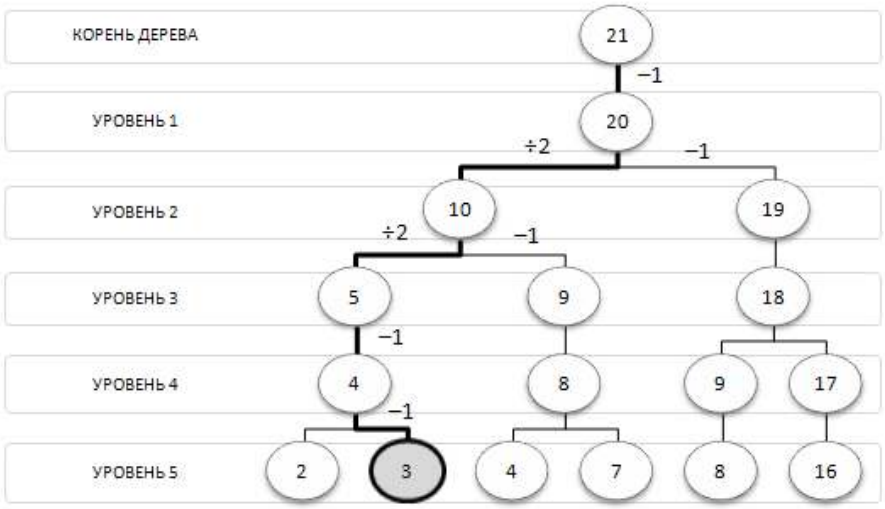


Рис. 7. Фрагмент дерева поиска вариантов решений обратной задачи

«Обратная» команда	Выполнение команды
2. вычти единицу	21– 1 = 20
1. дели на два	20 / 2 = 10
1. дели на два	10 / 2 = 5
2. вычти единицу	5 – 1 = 4
2. вычти единицу	4 – 1 = 3

В ответе номера операций следует записать в обратном порядке, от листа к корню.

Из рисунка видно, что решение обратной задачи менее трудоемко. Это объясняется тем, что не все числа кратны двум, следовательно, не всегда можно выполнить команду **раздели на два**, и количество узлов дерева резко сокращается.

Отметим, что в дереве решения есть два узла, содержащих значение 4. Следовательно. если потребуется из числа 4 получить число 21 не более чем за пять команд, то существуют два решения. Если же потребуется получить число 21 из числа 6 не более чем за пять команд, решения не существует.

При выборе способа решения подобных задач необходимо учитывать, что выполнение операции целочисленного деления без остатка не всегда возможно, а по условиям задач из целого числа получают целое. Поэтому если в системе команд есть операция деления, можно строить дерево решения прямой задачи; если в системе команд есть операция умножения и нет операции деления, удобнее решать обратную задачу.

***Ответ*:** 22112

Пример 5 задания с кратким ответом

У исполнителя Вычислитель две команды, которым присвоены номера:

1. возведи в квадрат

2. прибавь 1

Первая из них возводит число на экране в квадрат, вторая – увеличивает его на 1. Запишите порядок команд в программе получения из 4 числа 290, содержащей не более 4 команд, указывая лишь номера команд.

(Например, программа **12122** – это программа

возведи в квадрат

прибавь 1

возведи в квадрат

прибавь 1

прибавь 1

которая преобразует число 2 в 27.)

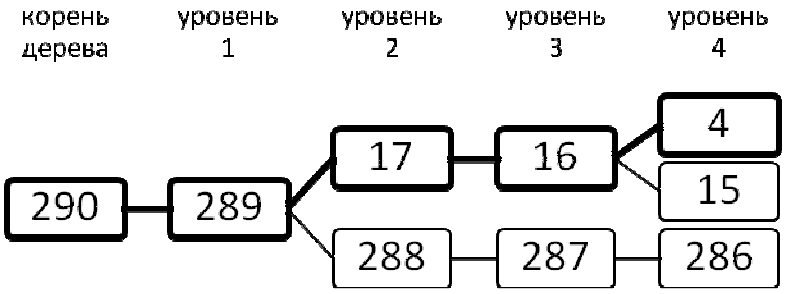
Решение.

Построим дерево решения обратной задачи, т.к. команда **извлеки квадратный корень**, обратная команде **1. возведи в квадрат**, не всегда дает целочисленный результат, необходимый для решения задачи. Вершины, содержащие не целое число, не будем строить, количество узлов дерева будет меньше, чем при решении прямой задачи. Дерево строим до четвертого уровня, используя обратные команды:

1. извлеки квадратный корень

2. вычти 1

Верхняя ветвь соответствует команде 1, нижняя – команде 2; если из узла выходит одна ветвь, она соответствует команде 2:



Решение выделено жирными линиями, от корня к листу выполняется последовательность команд 2121. В ответе следует записать ее в обратном порядке.

***Ответ*:** 1212

Пример 6 задания с кратким ответом

Специализированный процессор работает с положительными целыми однобайтовыми числами. Он может выполнять две команды:

1. сдвиг числа вправо на один двоичный разряд

2. прибавь 1

На вход процессору подается число 14, и процессор выполняет следующую последовательность команд 212112.

Укажите наименьшее возможное число команд, приводящее при том же исходном значении к тому же результату.

Решение.

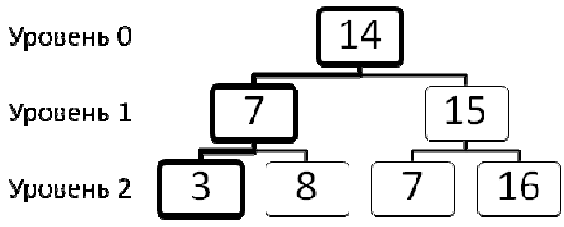
Процессор всегда обрабатывает двоичные числа. Команда **1. сдвиг числа вправо на один двоичный разряд** соответствует выполнению операции целочисленного деления двоичного числа на 2 с отбрасыванием остатка (см. тему «Системы счисления»), на школьном АЯ и в ряде языков программирования эта операция обозначается `div`. Например, для исходного двоичного числа 11101_2 результатом выполнения команды будет число 1110_2, для исходного десятичного числа 29 результат равен 14.

Определим результат выполнения последовательности команд 212112 для исходного числа $14 = 1110_2$:

Команда	CC_2	CC_{10}
2. прибавь 1	$1110_2+1_2=1111_2$	14+1=15
1. сдвиг числа вправо на один двоичный разряд	111_2	div(15,2)=7
2. прибавь 1	$111_2+1_2=1000_2$	7+1=8
1. сдвиг числа вправо на один двоичный разряд	100_2	div(8,2)= 4
1. сдвиг числа вправо на один двоичный разряд	10_2	div(4,2)= 2
2. прибавь 1	$10_2+1_2=11_2$	2+1=3

Таким образом, в результате выполнения заданной последовательности команд получено число 3.

Построим дерево решений, левая ветвь будет соответствовать первой команде, правая ветвь и ветвь вниз – второй команде **прибавь 1**. Дерево будем строить «в ширину» до первого получения результата.



На уровне 2 получен требуемый результат (видно, что тройка появится и на следующем уровне). Следовательно, достаточно выполнения последовательности двух команд 11.

***Ответ*:** 2

Пример 7 задания с кратким ответом

У исполнителя Калькулятор две команды, которым присвоены номера:

1. умножь на 2

2. прибавь 5

Первая команда умножает число на 2, вторая увеличивает его на 5.

Программа для Калькулятора – это последовательность команд.

Сколько есть программ, которые число 3 преобразуют в число 47?

Решение:

В задаче нет ограничения на количество команд в программе. Отметим, что обе команды увеличивают число. Если в результате выполнения какой-либо программы будет получено число больше 47, дальнейшее выполнение команд не приведет к необходимому результату.

Задание можно решать разными способами. Рассмотрим два из них: построение дерева решения и вычисление результата по рекуррентной формуле[2].

Способ 1. Построение дерева выполнения команд.

В отличие от предыдущей задачи, где мы искали самую короткую программу получения одного числа из другого, в этом задании необходимо найти количество вершин, содержащих результат. Если строить дерево прямого решения (получение числа 47 из числа 3), оно будет полным, т.к. из каждой вершины можно выполнить обе команды. Если использовать обратные команды

1. раздели на 2

2. вычти 5

дерево будет меньшего размера, т.к. не все числа кратны двум, и команда 1 не может быть выполнена для нечетных чисел (из условия задачи очевидно, что все числа, получаемые в результате выполнения команд – целые).

Будем строить дерево решений получения числа 3 из числа 47 с использованием обратных команд. Корнем дерева является вершина со значением требуемого результата 47, т.к. команды только увеличивают значение числа. Задача – найти количество вершин со значением 3. К каждой из них ведет свой путь, т.е. своя программа. Так как неизвестно количество команд в программах (количество уровней дерева), будем строить дерево в глубину.

---

[2] Рекуррентной называется формула, выражающая n-й член числовой последовательности через несколько предыдущих её членов, например, числа Фибоначчи задаются формулами: $a_0 = 0$, $a_1 = 1$, $a_{n+2} = a_{n+1} + a_n$, $(n \geq 0)$.

Будем считать, что левая ветвь соответствует команде 1, правая ветвь – команде 2. Если из вершины может выйти только одна ветвь (команда 2), направим ее вниз. Листьями дерева будут вершины, значения которых равно или меньше 3, они обозначены прямоугольниками. Выделим цветом листья, значение которых равно 3.

Сначала из корня дерева построим левые ветви до листьев. Затем вернемся выше на ближайший уровень, из которого еще не построена правая ветвь, и продолжим построение.

Проиллюстрируем построение дерева. Первая программа состоит из последовательности команд 212111 и получает из числа 47 число

((47 – 5) / 2 – 5) / 2 / 2 / 2 = 2<3

(см. рис. 8). Этот лист не выделен на рисунке цветом, т.к. не получено требуемое число 3.

Возвращаемся вверх до первой вершины, из которой еще не была выполнена вторая команду – это вершина 4, выполняем команду 2 (Вычти 5). Получим лист со значением – 1 < 3.

Еще раз возвращаемся вверх, на этот раз до листа 8, выполним команду 2, получим число 3 – лист дерева с нужным результатом. Этот лист выделим цветом (рис. 9).

Затем выполним команду 2 из листа 16, получим 11, продолжим построение до получения листьев 3 (выделен цветом) и 1 (рис. 10).

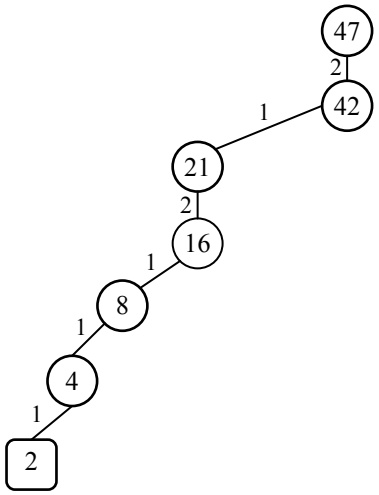


Рис. 8

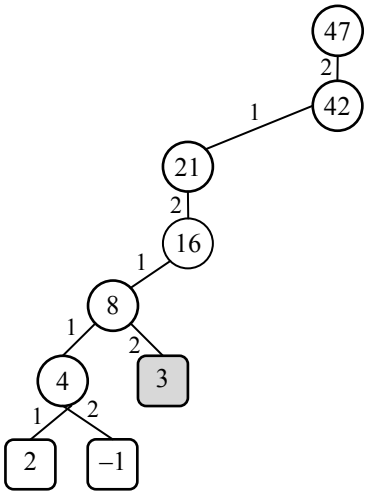


Рис. 9

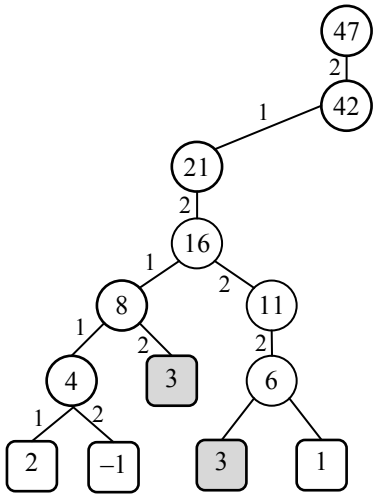


Рис. 10

Следующая команда выполняется от вершины 42. При построении получим последовательность вершин 37, 32, 16. Обратим внимание на то, что вершина 16 нам уже встречалась (обе вершины выделены жирным контуром), из нее можно получить два листа с интересующим нас результатом – числом 3. Поэтому можно не продолжать построение от второй вершины 16, а сразу записать, что будет получено два листа (рис. 11).

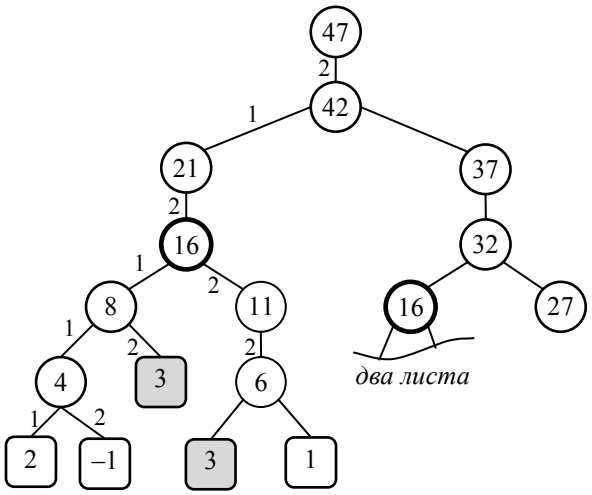


Рис. 11

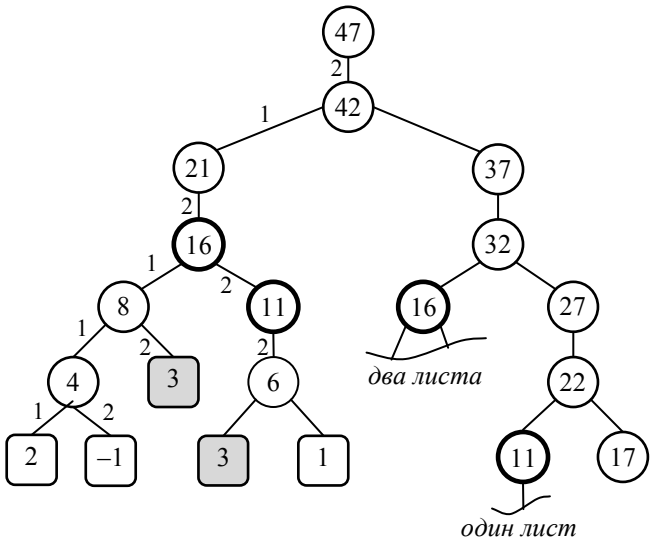


Рис. 12

Аналогично не продолжаем построения из вершин 11 (рис.12) и 6 (рис. 13), которые уже встречались.

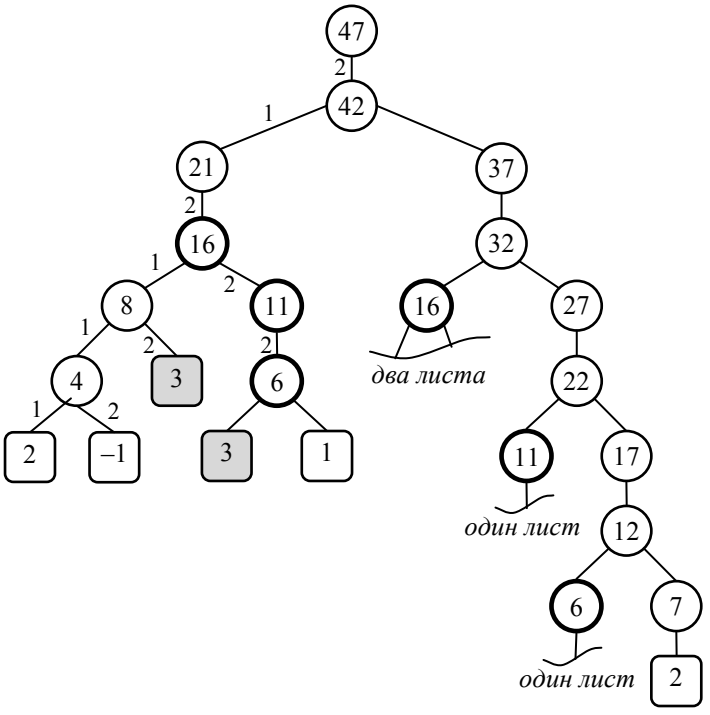


Рис. 13

Для получения результата следует подсчитать количество листьев со значением 3: 2 + 2 + 1 + 1 = 6.

Способ 2. Использование рекуррентных соотношений

Введем функцию $F(n)$, значением которой является количество программ Калькулятора, преобразующих заданное число 3 к аргументу функции n. Следует определить, каким образом можно вычислить значение функции по ее предыдущим значениям, т.е. составить

рекуррентные соотношения. Кроме того, для рекуррентных соотношений необходимо задать некоторые начальные значения функции.

Шаг 1. Определим начальные значения функции $F(n)$. После выполнения первой команды программы получим либо $3 \cdot 2 = 6$, либо $3 + 5 = 8$. Ни 6, ни 8 невозможно получить другой программой (последовательностью команд). Тогда

$F(6) = F(8) = 1$.

Обе команды увеличивают обрабатываемое число, следовательно, невозможно получить из числа 3 числа меньше 6 и число 7. Можно записать:

$F(n) = 0$, при $n < 6$, $n = 7$.

Шаг 2. Рекуррентная формула для нечетных аргументов. Нечетные числа $n = 2k + 1$ можно получить только после выполнения команды «прибавь 5». Эта команда применяется к предыдущему числу $(n – 5)$: $2k + 1 – 5 = 2k – 4$. Для нечетных аргументов функции можно записать:

$F(2k + 1) = F(2k – 4)$, при $k >= 4$;

Шаг 3. Рекуррентная формула для четных аргументов. Четные числа $n = 2k$ можно получить после выполнения любой из двух команд. При этом команда «умножь на 2» применяется к предшествующему числу k, а команда «прибавь 5» – к предшествующему числу $2k – 5$. Для четных чисел можно записать:

$F(2k) = F(2k – 5) + F(k)$, при $k > 4$.

Шаг. 4. Вычисления будем проводить по рекуррентным формулам

$F(2k + 1) = F(2k – 4)$, при $k >= 4$;

$F(2k) = F(2k – 5) + F(k)$, при $k > 4$;

$F(6) = 1$; $F(8) = 1$

$F(k) = 0$, при $k < 6$, $k = 7$.

Для записанных соотношений

$F(9) = F(4) = 0$,

$F(10) = F(5) + F(5) = 0$,

$F(11) = F(6) = 1$, и т.д.

Шаг 5. Вычислим по рекуррентным формулам количество программ получения числа 47 из числа 3:

$F(47) = F(42) = F(37) + F(21) =$

$= F(32) + F(21) = F(27) + F(16) + F(16) =$

$= F(22) + 2 \cdot F(16) = F(17) + 2 \cdot F(16) + F(11) =$

$= 2 \cdot F(16) + F(12) + F(11) =$

$= F(12) + 3 \cdot F(11) + 2 \cdot F(8) =$

$= 2 + F(7) + 4 \cdot F(6) = 2 + 4 = 6$

Ответ: 6

Цепочки (конечные последовательности)

Пример 8 задания с кратким ответом

Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа – латинской буквы «A». Каждая из последующих цепочек создается такими действиями: в очередную строку сначала записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на **i**-м шаге пишется **i**-я буква алфавита), к ней слева дважды подряд приписывается предыдущая строка. Вот первые 4 строки, созданные по этому правилу:

(1) A

(2) AAB

(3) AABAABC

(4) AABAABCAABAABCD

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Имеется задание:

«Определить четыре символа, стоящих подряд в n-й строке, начиная с позиции $\mathbf{2^{n-1}-1}$, считая от левого края цепочки».

Выполните это задание для n = 9

Решение:

Определим количество символов N_i в i-той строке: $N_i = N_{i-1}*2+1$; $N_1 = 1$.

№ строки	Количество символов в строке		символы в конце строки
1	1	2^1-1	
2	1·2+1=3	2^2-1	B
3	3·2+1=7	2^3-1	C
4	7·2+1=15	2^4-1	D
5	15·2+1=31	2^5-1	E
6	31·2+1=63	2^6-1	F
7	63·2+1=127	2^7-1	G
8	127·2+1=255	2^8-1	H
9	255·2+1=511	2^9-1	I

Вычислим номер символа, стоящего на позиции $\mathbf{2^{n-1} - 1}$, для n=9 номер символа равен $2^8 - 1 = 255$. Нам необходимо определить четыре символа, стоящих на позициях 255, 256, 257 и 258 соответственно.

Первые 255 символов девятой строки есть восьмая строка. На 255-ой позиции в восьмой строке стоит символ H. В девятой строке сразу за 255-м символом начинается повтор восьмой строки с начала, то есть AABAAB…...

Покажем эти позиции в строке:

255	256	257	258	259	260
H	A	A	B	A	A

Ответ: HAAB

Пример 9 задания с кратким ответом

Строки (цепочки символов латинских букв) составляются по следующему правилу. Первая строка состоит из одного символа – латинской буквы «A». Каждая из последующих двенадцати цепочек создается такими действиями:

Сначала дважды записывается предыдущая строка;

Затем к ней приписываются две следующие по алфавиту буквы;

Пример:

(1) A

(2) AABC

(3) AABCAABCDE

(4) AABCAABCDEAABCAABCDEFG

Латинский алфавит (для справки)

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Запишите шесть символов подряд, стоящие в седьмой строке со 93-го по 98-е место (считая слева направо).

Решение:

Определим количество символов N_i в i-той строке: $N_i = N_{i-1} * 2 + 2$; $N_1 = 1$.

№ строки	Количество символов в строке	символы в конце строки
1	1	
2	$1 \cdot 2 + 2 = 4$	BC
3	$4 \cdot 2 + 2 = 10$	DE
4	$10 \cdot 2 + 2 = 22$	FG
5	$22 \cdot 2 + 2 = 46$	HI
6	$46 \cdot 2 + 2 = 94$	JK
7	$94 \cdot 2 + 2 = 190$	LM

Первые 94 символа седьмой строки – это шестая строка. С позиции 95 повторяется шестая строка сначала.

Покажем эти позиции в строке:

93	94	95	96	97	98	99	100
J	K	A	A	B	C	A	A

Ответ: JKAABC

Алгоритмы, заданные на естественном языке

Пример 10 задания с выбором одного ответа

Зачет по информатике проводится в форме компьютерного тестирования. Каждый студент имеет доступ к системе тестирования по паролю.

Преподаватель огласил студентам алгоритм получения пароля по заданной последовательности чисел:

если в последовательности все цифры, кратные трем, разделить на три, после чего в полученной последовательности все четные цифры увеличить на 1, а затем из полученной последовательности удалить все цифры меньшие трех, то полученная последовательность будет паролем:

Студент получил у преподавателя последовательность чисел 3791633 непосредственно перед началом зачета. Какой пароль он должен ввести для входа в систему тестирования?

1) 7 2) 73 3) 733 4) 833

Решение:

Выполним алгоритм получения пароля по шагам.

1) все цифры, кратные трем, разделить на три: 1731211
2) все четные цифры увеличить на 1: 1731311
3) удалить все цифры меньшие трех: 733

Ответ: № 3

Пример 11 задания с выбором одного ответа

Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам.

1. Записывается результат сложения значений старших разрядов двух заданных чисел.

2. К нему дописывается результат сложения значений средних разрядов этих чисел по такому правилу: если он меньше первой суммы, то второе полученное число приписывается к первому слева, иначе – справа.

3. Итоговое число получают приписыванием справа к полученному после второго шага числу суммы значений младших разрядов исходных чисел.

Пример. Исходные трехзначные числа 138, 212. Поразрядные суммы 3, 4, 10. Результат:3410.

Определите, какое из предложенных чисел может быть результатом работы автомата.

1) 3507 2) 13412 3) 91216 4) 13197

Решение:

Решим задачу с помощью рассуждений. Итоговое число состоит из трех частей, каждая из которых представляет собой сумму двух однозначных чисел.

- Сумма цифр старших разрядов принадлежит отрезку [2, 18]. Минимальные цифры в старших разрядах трехзначного числа равны 1, минимальная сумма старших разрядов равна 2. Максимальные цифры в разрядах равны 9, максимальная сумма равна 18.
- Суммы цифр остальных разрядов принадлежат отрезку [0, 18].

По заданию можно определить, что вторая часть числа должна быть больше первой. Рассмотрим каждое из предложенных чисел:

1) 3507 – варианты представления числа: 35 0 7, 3 50 7 и 3 5 07. В первых двух вариантах представления значения 35 и 50 больше 18. В третьем варианте десятичная запись числа начинается с 0 (07). Ответ неверный.

2) 13412 – разобьем число на части, учитывая, что каждая часть числа должна быть меньше 19: 13, 4, 12. 13 > 4, ответ неверный.

3) 91216 – можно разбить число на 9, 12 и 16. 9 < 12. Следовательно, вариант ответа может быть верным.

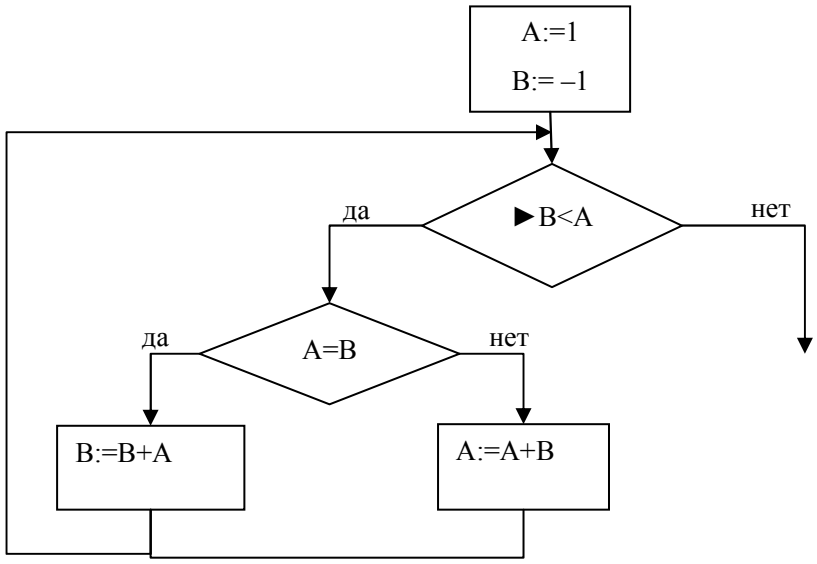
4) 13197 – каким бы образом мы не разбили это число на три части, одна из них всегда будет больше 18, значит вариант ответа неверный.

Ответ: 3

Алгоритмы, заданные блок-схемой

Пример 12 задания с кратким ответом

Определите значение переменной **A** после выполнения фрагмента алгоритма, представленного блок-схемой:



В ответе укажите одно число – значение переменной **A**.

Решение:

При решении задач такого типа могут применяться трассировочные таблицы как первого, так и второго вида. Однако, поскольку алгоритм содержит циклическую конструкцию в данной задаче удобнее построить трассировочную таблицу второго вида.

Устанавливаем контрольную точку в заголовке цикла с условием B < A.

Для данного алгоритма получим следующую трассировочную таблицу:

№ кт	B<A	A	B
1	да	1	–1
2	да	0	
3	да	–1	
4	нет		
Итог:		–1	–1

Ответ: –1

Построение трассировочной таблицы является надежным способом получения ответа на вопрос о результатах выполнения алгоритма, однако этот способ зачастую сопряжен со значительными затратами времени. Поэтому желательно попытаться выяснить смысл задачи, для решения которой разработан алгоритм. Если это сделать не удается, попытайтесь отследить закономерность изменения анализируемых данных в процессе построения трассировочной таблицы. В ряде случаев выявленная закономерность раскрывает суть решаемой задачи, что позволит получить ответ без построения полной трассировочной таблицы.

Алгоритмы, заданные на алгоритмическом языке

Пример 13 задания с выбором одного ответа

Определите значение переменной c после выполнения следующего фрагмента программы, в котором a, b и c – целочисленные переменные.

```
a:=90
b:=25
b:=a-b*2
a:=a-b*2
если a>2*b
     то a:=div(a,b)
     иначе b:=div(b,a)
все
c:=a+b
```

1) c=14 2) c=41 3) c=93 4) c=115

Решение:

Построим трассировочную таблицу первого вида:

№	Команда или условие (логическое выражение)	Вычисление правой части команды присваивания или условия (логического выражения)	a	b	c
1	a:=90		90		
2	b:=25			25	
3	b:=a-b*2	90-25*2=40		40	
4	a:=a-b*2	90-40*2=10	10		
5	a>2*b	10>40*2=нет			
6	b:=div(b,a)	div(40,10)=4		4	
7	c:=a+b	10+4=14			14
Результат			10	4	14

Ответ: № 1

Решения заданий демоварианта 2012

Задание А5

Характеристики задания

Проверяемые элементы содержания	Формальное исполнение алгоритма, записанного на естественном языке
Контролируемый элемент содержания (по кодификатору)	1.6.1. Формализация понятия алгоритма
Требования к уровню подготовки (по кодификатору)	1.1.3. Строить модели объектов, систем и процессов. Записывать алгоритмы на естественном языке и в виде блок - схем
Вид деятельности	Применение знаний и умений в стандартной ситуации
Уровень	базовый
Максимальный первичный балл	1
Время выполнения	2 мин.

Задание

А5 Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам.

1. Вычисляются три числа – сумма старших разрядов заданных чисел, сумма средних разрядов этих чисел, сумма младших разрядов.

2. полученные три числа записываются друг за другом в порядке убывания (без разделителей)

Пример. Исходные трехзначные числа 835, 196. Поразрядные суммы 9, 12, 11. Результат:12119

Определите, какое из следуюших чисел может быть результатом работы автомата.

1) 151303 2) 161410 3) 191615 4) 121613

Решение:

Решим задачу с помощью рассуждений. Итоговое число состоит из трех частей, каждая из которых представляет собой сумму двух однозначных чисел.

- Сумма цифр старших разрядов принадлежит отрезку [2, 18]. Минимальные цифры в старших разрядах трехзначного числа равны 1, минимальная сумма старших разрядов равна 2. Максимальные цифры в разрядах равны 9, максимальная сумма равна 18.
- Суммы цифр остальных разрядов принадлежат отрезку [0, 18].

Рассмотрим каждое из предложенных чисел:

1) 151303 – есть один вариант разбиения числа на три части 15, 13 и 03. Десятичная запись числа начинается с 0 (03). Ответ неверный.

2) 161410 – разобьем число на части, учитывая, что каждая часть числа должна быть меньше 19, получим 16, 14, 10. Числа расположены по убыванию, ответ верный.

3) 191615 – каким бы образом мы не разбили это число на три части, одна из них всегда будет больше 18, значит ответ неверный. (Можно разбить число на 19, 16 и 15. Упорядоченность по убыванию есть, но сумма двух цифр не может быть больше 18.)

4) 121613 – число разбивается на части 12, 16, 13, но последовательность чисел не упорядочена, ответ неверный

Ответ: 2

Задание А13

Характеристики задания

Проверяемые элементы содержания	Умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд
Контролируемый элемент содержания (по кодификатору)	1.6.3. Построение алгоритмов и практические вычисления
Требования к уровню подготовки (по кодификатору)	1.1.4. Читать и отлаживать программы на языке программирования
Вид деятельности	Применение знаний и умений в стандартной ситуации
Уровень	повышенный
Максимальный первичный балл	1
Время выполнения	2 мин.

Задание

А13 Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	**вниз**	**влево**	**вправо**

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно	**снизу свободно**	**слева свободно**	**справа свободно**

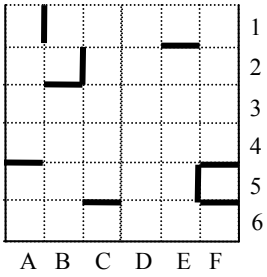
Цикл

ПОКА <*условие*> команда

выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Если РОБОТ начнет движение в сторону стены, то он разрушится и программа прервется.

Сколько клеток лабиринта соответствует требованию, что, выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал движение?

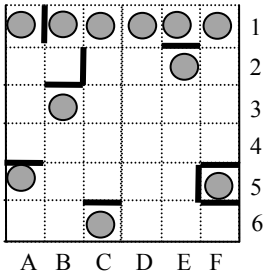
НАЧАЛО ПОКА <**справа свободно**> **вниз** ПОКА <**снизу свободно**> **влево** ПОКА <**слева свободно**> **вверх** ПОКА <**сверху свободно**> **вправо** КОНЕЦ	

1) 1 2) 3 3) 5 4) 7

Решение:

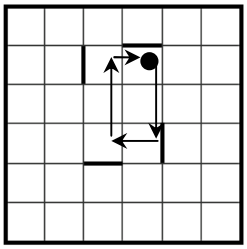
Как и в предыдущих подобных заданиях начнем с определения клеток, которые надо проверить, т.е. сократим перебор.

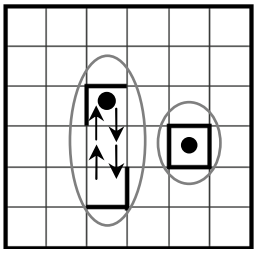
Заметим, что РОБОТ заканчивает выполнение алгоритма в клетке, ограниченной стеной сверху, следовательно, надо проверить только клетки, имеющие стенку сверху. Отметим эти клетки точками.



На рисунке отмечено 11 клеток – это максимальное количество клеток, которые потребуется проверять.

Определим траекторию движения РОБОТа: в общем случае это прямоугольник, движение начинается вниз и идет по часовой стрелке. «Тормоз» РОБОТа (стена, заставляющая РОБОТа изменить направление движения) всегда слева по ходу его движения. Возможные виды траекторий показаны на рисунке ниже.

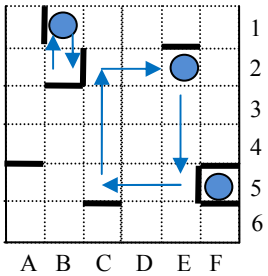




Отметим, что при выполнении заданного алгоритма РОБОТ всегда разрушается, если клетка ограничена тремя стенами («стакан»). Таких клеток в этом примере нет, если бы они были, можно было бы сразу удалим отметки с таких клеток, т.е. не проверять такие клетки.

Для оставшихся отмеченных клеток проверим выполнение алгоритма:

A1	Разрушится при движении влево
B1	B1- B2 - B1
C1	C1-C2-C3-C4-C5-разрушился при движении вниз
D1	D1-D2-D3-D4-D5-D6- разрушился при движении вниз
E1	E1-D1-C1-B1-разрушился при движении влево
F1	F1-E1-разрушился при движении вверх
A5	A5-A6- разрушился при движении вниз
B3	B3-B4-B5-B6- разрушился при движении вниз
C6	разрушился при движении вниз
E2	E2-E3-E4-E5-поворот-D5-C5-поворот – C4-C3-C2-поворот-D2-E2
F5	РОБОТ повернется на 360 градусов и останется на месте



Есть три клетки, выходя из которых РОБОТ выполнит программу, не разрушится и вернется в исходную клетку. Это – B1 E2, F5.

Ответ: 2

Задание B2

Характеристики задания

Проверяемые элементы содержания	Умение создавать линейный алгоритм для формального исполнителя
Контролируемый элемент содержания (по кодификатору)	1.6.1. Формализация понятия алгоритма
Требования к уровню подготовки (по кодификатору)	1.1.4. Читать и отлаживать программы на языке программирования
Вид деятельности	Применение знаний и умений в стандартной ситуации
Уровень	базовый
Максимальный первичный балл	1
Время выполнения	4 мин.

Задание

B2 У исполнителя Утроитель две команды, которым присвоены номера:

1. **прибавь 1**
2. **умножь на 3**

Первая из них увеличивает число на экране на 1, вторая - утраивает его.

Запишите порядок команд в программе преобразования числа 1 в число 22, содержащей не более 5 команд, указывая лишь номера команд (например, 21211 – это программа:

умножь на 3

прибавь 1

умножь на 3

прибавь 1

прибавь 1

которая преобразует число 1 в 14.)

(Если таких программ более одной, запишите любую из них.)

Решение:

Построим дерево выполнения команд. Заметим, что в результате выполнения любой программы, составленной из предложенных команд, получим целое число.

Команда 1 и обратная ей могут применяться ко всем числам.

Команду 2 можно применить ко всем числам. Но обратную к ней команду «раздели на 3» можно применить только к числам, кратным 3 (учитывая, что работаем с целыми числами). Таких чисел в три раза меньше, и обратная команда может применяться в 3 раза реже.

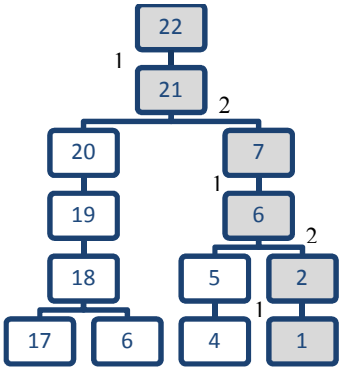
Вывод: количество ветвей дерева, построенного от результата 22 к исходному числу 1 с использованием команд, обратных заданным, меньше, чем при построении дерева от заданного числа 1 к результату 22.

Обратные команды:

1. вычесть 1

2. разделить на 3

Из каждого узла дерева могут выходить две ветви: левая соответствует выполнению обратной команды 1, правая – выполнению обратной команды 2. Если команда 2 не может выполниться, из узла выходит одна ветвь, соответствующая команде 1.



Запишем последовательность команд получения из числа 1 числа 22 (от листа к корню дерева).

Ответ: 12121

Задание B13

Характеристики задания

Проверяемые элементы содержания	Умение анализировать результат исполнения алгоритма
Контролируемый элемент содержания (по кодификатору)	1.6.1 Формализация понятия алгоритма
Требования к уровню подготовки (по кодификатору)	1.1.3. Строить модели объектов, систем и процессов. Записывать алгоритмы на естественном языке и в виде блок - схем
Вид деятельности	Применение знаний и умений в новой ситуации
Уровень	Повышенный
Максимальный первичный балл	1
Время выполнения	7 мин.

Задание

B13 У исполнителя Кузнечик две команды:

1. прибавь 3,

2. вычти 2.

Первая из них увеличивает число на экране на 3, вторая – уменьшает его на 2 (отрицательные числа допускаются).

Программа для Кузнечика – это последовательность команд. Сколько различных чисел можно получить из числа 1 с помощью программы, которая содержит ровно 5 команд?

Ответ: _____.

Решение

В общем случае в дереве решений, содержащем 5 уровней, может быть $2^5 = 32$ листа. Построение такого дерева требует времени и внимания.

Поскольку обе команды исполнителя аддитивные (прибавляют положительное или отрицательное число), последовательность выполнения команд не имеет значения (сложение ассоциативно и коммутативно). Разные программы могут выдавать один и тот же результат, например, в результате выполнения трех разных программ:

Программа 1	***Программа 2***	***Программа 3***
прибавь 3	**прибавь 3**	**вычти 2**
вычти 2	**вычти 2**	**вычти 2**
вычти 2	**прибавь 3**	**прибавь 3**
вычти 2	**вычти 2**	**прибавь 3**
прибавь 3	**вычти 2**	**вычти 2**

будет получено число $1 + 2 \cdot 3 + 3 \cdot (–2) = 1$

Таким образом, в случае, когда в командах указаны конкретные слагаемые, важно лишь количество команд 1 и 2 в программе.

Пусть x – количество команд 1 в программе, $0 \le x \le 5$, тогда количество команд 2 равно $(5 – x)$. В результате выполнения какой-либо программы из 5 команд могут быть получены числа

$K_x = 1 + x \cdot 3 + (5 – x) \cdot (–2) = 5 \cdot x – 9.$

где x может принимать 6 разных значений (0, 1, 2, 3, 4, 5). Числа K_x линейно зависят от x, следовательно, в результате выполнения программ, состоящих из 5 команд, будет получено 6 разных чисел.

Заметим, что если бы команды имели вид

1. прибавь a

2. вычти b

то потребовались бы дополнительные рассуждения, т.к. при некоторых сочетаниях значений a и b можно получить одно и то же число.

Ответ: 6.

Задание С3

Характеристики задания

Проверяемые элементы содержания	Умение построить дерево по заданному алгоритму и обосновать результат построения
Контролируемый элемент содержания (по кодификатору)	1.6.3. Построение алгоритмов и практические вычисления
Требования к уровню подготовки (по кодификатору)	1.1.5. создавать программы на языке программирования по их описанию
Вид деятельности	Применение знаний и умений в новой ситуации
Уровень	высокий
Максимальный первичный балл	3
Время выполнения	30 мин.

Условие и решение задания С3 приводится в демоварианте.

Задания для самостоятельного решения

Задания с выбором одного ответа

№ 1

Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	**вниз**	**влево**	**вправо**

При выполнении этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

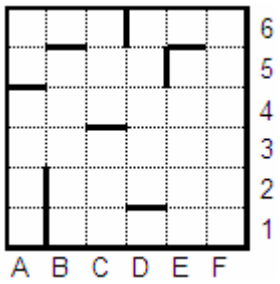
Четыре команды служат для проверки истинности условия отсутствия соответствующей стены у той клетки, где находится РОБОТ:

сверху свободно	**снизу свободно**	**справа свободно**	**слева свободно**

Цикл ПОКА < *условие* > *команда* выполняется, пока условие истинно, иначе происходит переход на следующую строку. Если РОБОТ начнет движение в сторону стены, то он разрушится, и выполнение программы прервется.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал выполнение программы?

НАЧАЛО
ПОКА < слева свободно > вниз
ПОКА < снизу свободно > вправо
ПОКА < справа свободно > вверх
ПОКА < сверху свободно > влево
КОНЕЦ



1) 1 2) 2 3) 3 4) 4

№ 2

Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх	**вниз**	**влево**	**вправо**

При выполнении этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды служат для проверки истинности условия отсутствия соответствующей стены у той клетки, где находится РОБОТ:

сверху свободно	**снизу свободно**	**справа свободно**	**слева свободно**

Цикл ПОКА < *условие* > *команда* выполняется, пока условие истинно, иначе происходит переход на следующую строку. Если РОБОТ начнет движение в сторону стены, то он разрушится, и выполнение программы прервется.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ уцелеет и остановится в той же клетке, с которой он начал выполнение программы?

НАЧАЛО

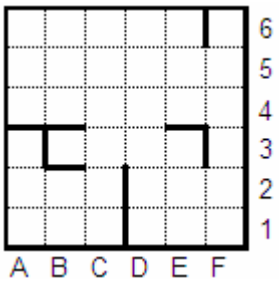
ПОКА < слева свободно > влево

ПОКА < снизу свободно > вниз

ПОКА < справа свободно > вправо

ПОКА < сверху свободно > вверх

КОНЕЦ



1) 1 2) 2 3) 3 4) 4

№ 3

Петя пригласил своего друга Сережу в гости, но не сказал ему код от цифрового замка своего подъезда, а послал следующее SMS-сообщение: “в последовательности цифр 4, 7, 2, 1, 9 все числа меньше пяти увеличить на 1, а из всех больших пяти вычесть 1, затем удалить из полученной последовательности все нечетные числа”. Выполнив действия, указанные в сообщении, Сережа получил следующий код для цифрового замка:

1) 628 2) 42 3) 68 4) 428

№ 4

Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам.

1. Записывается результат сложения значений старших разрядов заданных чисел.

2. К нему дописывается результат сложения значений средних разрядов этих чисел по такому правилу: если он меньше первой суммы, то второе полученное число приписывается к первому слева, иначе – справа.

3. Итоговое число получают приписыванием справа к полученному после второго шага числу суммы значений младших разрядов исходных чисел.

Пример. Исходные трехзначные числа: 328, 207. Поразрядные суммы: 5,2,15. Результат:2515

Определите, какое из предложенных чисел может быть результатом такой операции.

1) 15117 2) 19213 3) 181717 4) 11146

№ 5

Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам.

1. Записывается результат сложения значений старших разрядов заданных чисел.

2. К нему дописывается результат сложения значений средних разрядов этих чисел по такому правилу: если он меньше первой суммы, то второе полученное число приписывается к первому справа, иначе – слева.

3. Итоговое число получают приписыванием справа к полученному после второго шага числу суммы значений младших разрядов исходных чисел.

Пример. Исходные трехзначные числа 721, 195. Поразрядные суммы: 8, 11, 6. Результат: 1186.

Определите, какое из предложенных чисел может быть результатом такой операции.

1) 1112 2) 151612 3) 19118 4) 12045

Задания с кратким ответом

№ 6

Исполнитель Робот ходит по клеткам клетчатой доски, переходя по одной из команд **вверх, вниз, вправо, влево** в соседнюю клетку в указанном направлении. Робот выполнил следующую программу:

вправо, вправо, вверх, влево, вправо, влево, влево, вверх, вправо.

Укажите наименьшее возможное число команд в программе, приводящей Робота из той же начальной клетки в ту же конечную.

№ 7

У исполнителя Вычислитель две команды, которым присвоены номера:

1. прибавь 3

2. умножь на 4

Выполняя первую из них, Вычислитель прибавляет к числу на экране 1, а выполняя вторую, увеличивает его в четыре раза. Запишите порядок команд в программе получения из числа 2 числа 152, содержащей не более пяти команд, указывая лишь номера команд.

(Например, программа **12111** – это программа

прибавь 3

умножь на 4

прибавь 3

прибавь 3

прибавь 3

которая преобразует число 1 в число 25).

№ 8

Специализированный процессор работает с двоичными положительными целыми однобайтовыми числами. Он может выполнять две команды:

1. сдвиг числа влево на один двоичный разряд

2. вычти 1

Для заданного десятичного числа 14 процессором выполнена последовательность команд 11222. Запишите полученный результат в десятичной системе счисления.

№ 9

Строки (цепочки символов латинских букв) создаются по следующему правилу. Первая строка состоит из одного символа – латинской буквы «А». Каждая из последующих цепочек создается такими действиями: в очередную строку сначала записывается буква, чей порядковый номер в алфавите соответствует номеру строки (на **i**-м шаге пишется **i**-я буква алфавита), к ней слева дважды подряд приписывается предыдущая строка.

Вот первые 4 строки, созданные по этому правилу:

(1) A

(2) AAB

(3) AABAABC

(4) AABAABCAABAABCD

Латинский алфавит (для справки):

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Имеется задание:

«Определить пять символов, стоящих в n-й строке начиная с позиции $\mathbf{2^{n-1}-2}$, считая от левого края цепочки».

Выполните это задание для n = 9

№ 10

Строки (цепочки символов латинских букв) составляются по следующему правилу. Первая строка состоит из одного символа – латинской буквы «А». Каждая из последующих цепочек создается такими действиями:

- Сначала дважды записывается предыдущая строка;

- Затем к ней приписывается следующая по алфавиту буква;

Пример:

(1) A

(2) AAB

(3) AABAABC

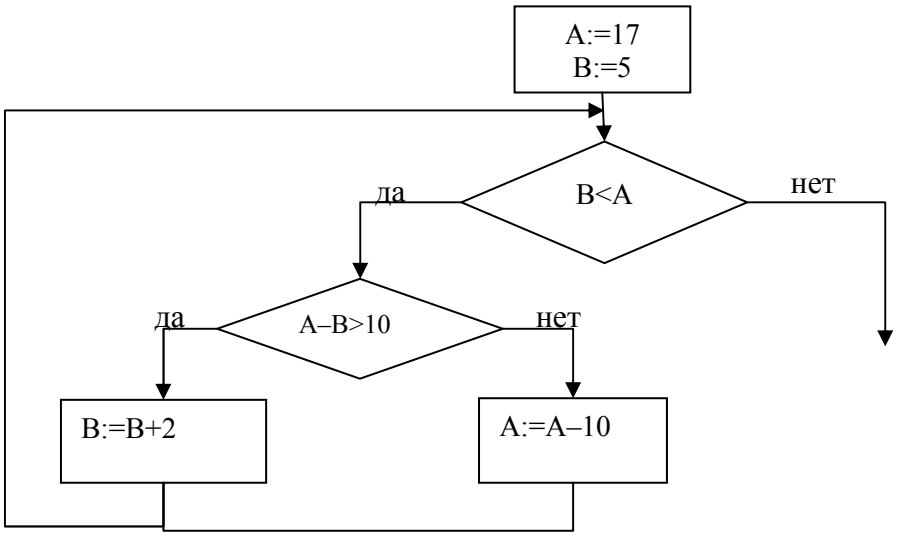
(4) AABAABCAABAABCD

Латинский алфавит (для справки) ABCDEFGHIJKLMNOPQRSTUVWXYZ

Запишите четыре символа подряд, стоящие в седьмой строке со 123-го по 126-е место (считая слева направо).

№ 11

Определите значение переменной **B** после выполнения фрагмента алгоритма, представленного блок-схемой:



Знаком **:=** обозначена операция присваивания.

В ответе укажите одно число – значение переменной **B**.