

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

С.В. Сидоров
Г.С. Крынецкая

Методическое пособие
по подготовке к олимпиадам
школьников инженерной
направленности

**Информационно-технологическое
направление
«МАТЕМАТИКА
В ИНФОРМАТИКЕ. РЕШЕНИЕ
ОЛИМПИАДНЫХ ЗАДАЧ»**

9–10 классы



Москва 2017

УДК 681.3
К85

Крынецкая Г.С.

К85 Методическое пособие по подготовке к олимпиадам школьников инженерной направленности направленности : Информационно-технологическое направление «Математика в информатике. Решение олимпиадных задач» : 9–10-й классы / Г.С. Крынецкая, С.В. Сидоров. – М. : Изд. Дом НИТУ «МИСиС», 2017. – 67 с.

Рассматриваются олимпиадные задачи информационно-технологического направления предыдущих лет для школьников старших классов. По каждой задаче приводится теоретическое сопровождение темы задачи, различные способы ее решения, типовые ошибки, допущенные в решении подобных задач.

Темы задач – кодирование и представление различных видов информации, различные системы счисления, различные типы данных, алгоритмизация (следование алгоритму, составление алгоритма, анализ алгоритма), программирование и математика.

В задачах программирования коды программ приводятся на языках C#, PascalABC и VBA.

Предназначено для подготовки к решению олимпиадных задач, а также для углубленного изучения информатики и математики школьниками старших классов школы.

УДК 681.3

© С.В. Сидоров,
Г.С. Крынецкая, 2017
© НИТУ «МИСиС», 2017

ОГЛАВЛЕНИЕ

Введение	4
Задача 1 (Задача контрольной МФТИ «Выходи решать» 2017)	5
Задача 2 (Олимпиада школьников МИСиС, 2011 год)	8
Задача 3 (Олимпиада школьников МИСиС, 2011 год)	11
Задача 4 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 10 класс)	13
Задача 5 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 10 класс)	15
Задача 6 (Заключительный этап олимпиады школьников «Ломоносов-2015» по информатике (5–9 классы))	18
Задача 7 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 9 класс)	19
Задача 8 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 11 класс, финальный тур)	23
Задача 9 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 10 класс)	27
Задача 10 (Задача контрольной МФТИ «Выходи решать» 2017)	32
Задача 11 (10 класс, Роботрон, отборочный тур 2016 год)	36
Задача 12 (Всероссийская олимпиада школьников по информатике, 2014–15 уч. год Первый (школьный) этап, г. Москва Задания для 9–11 классов)	46
Задача 13 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 11 класс)	55
Заключение	66

ВВЕДЕНИЕ

Дисциплина «Информатика» и связанные с ней информационные технологии развиваются настолько стремительно, что за ними не поспевает разработка учебных программ. Поэтому так важна роль олимпиад по информатике, которые мотивируют учащихся к самостоятельному и более углубленному изучению предмета. С другой стороны это подтверждается увеличением интереса учащихся к данным олимпиадам.

Цель данного пособия – пробудить интерес к углубленному изучению информатики, решению неординарных задач, к программированию. Авторы надеются, что изучение данного пособия будет способствовать развитию навыков создания и отладки коды программ, более глубокому изучению языков программирования, умению работать с символьными и с текстовыми типами данных.

Олимпиада по информатике «МИСиС зажигает звезды» проводится для 7, 8, 9, 10 и 11 классов в два этапа: отборочный и заключительный. Варианты для всех классов различаются по уровню сложности предлагаемых задач.

В пособие приведены задания олимпиад предыдущих лет. Все задачи снабжены подробными решениями, анализом ошибок и рекомендациями по оптимизации алгоритмов. Задачи имеют разный уровень сложности, что позволяет использовать пособие для учащихся с различной степенью подготовки.

Программные реализации задач приведены на трех наиболее популярных в настоящее время языках программирования:

- Pascal ABC, как один из популярных языков, изучаемых в школах;
- C# по причине углубленного изучения в МИСиС;
- VBA (Visual Basic for Application) как наиболее распространенная версия Бейсика, входящая в состав пакета Microsoft Office.

Это позволяет наиболее заинтересованным читателям потренироваться при решении задач на всех этих языках программирования, сравнить их возможности, почувствовать их преимущества и недостатки, убедиться, что все языки структурного программирования в целом похожи.

Пособие предназначено для школьников 9–10 классов. Основное внимание в нем уделено представлению информации в памяти компьютера, алгоритмизации задач, различным типам данных, и написанию программных кодов поставленных задач. Также пособие может быть полезно студентам первых курсов и учителям информатики.

В данном пособии ко всем заданиям приведены решения, в частности детально разобраны программные коды. Читателю рекомендуют не только теоретически изучить приведенные задачи, но и самостоятельно повторить программную реализацию задач.

Для практического закрепления навыков авторы рекомендуют самостоятельно реализовать программные коды задачи 6 и 7, а также – программно реализовать логику автомата задачи 8. Несмотря на то, что в приведенных заданиях программная реализация не предусмотрена, эти задачи дают возможность попрактиковаться в реализации логики алгоритма, в работе с массивами и с тестовыми данными.

Задача 1 (Задача контрольной МФТИ «Выходи решать» 2017)

Космические разведчики обнаружили загадочный прибор с клавиатурой и дисплеем. При включении прибора на дисплее появилось предложение ввести числовую последовательность. По окончании ввода последовательности на дисплее появлялся текст.

Результаты исследования приведены в таблице:

«Введите последовательность»	«Ваш результат»
0001001000	олово
010010	лов
11001001000	слово
100001	вол
101110110	вес
10000100110	волос
1100001001011101111	словей

Что получится, если в прибор ввести следующую информацию?

1011101100100	?
---------------	---

Проверяется

Навыки кодирования информации.

Обсуждается

Анализ кода. Кодирование и декодирование информации с использованием кода переменной длины.

Теоретическое сопровождение темы. Решение

В данной задаче вполне логично сделать вывод о том, что каждая двоичная последовательность перекодируется в текст: все приведенные варианты тестирования прибора соответствуют этому наблюдению. Каждый символ текста соответствует своей последовательности двоичного кода. Следовательно, задача сводится к определению соответствия между двоичным кодом и символами текста согласно имеющимся вариантам.

Внимательно рассмотрим имеющиеся данные; определим количество символов в исходной последовательности, в результирующих данных:

Последовательность	Длина последовательности	Результат	Длина результата
0001001000	10	олово	5
010010	6	лов	3
11001001000	11	слово	5
100001	6	вол	3
101110110	9	вес	3
10000100110	11	волос	5
1100001001011101111	19	соловей	7

Исходная последовательность формируется только двоичным кодом (используются символы «0» и «1»), результирующая последовательность формируется шестью символами «о», «л», «в», «с», «е», «й».

Можно предположить, что каждый символ текста кодируется двоичным кодом заданной, фиксированной длины; этому предположению соответствуют слово «олово», кодируемое 10 двоичными символами и слово «лов», кодируемое 6 двоичными символами. Но данное предположение не находит подтверждения в словах «слово», кодируемое 11 двоичными символами и «вес», кодируемое 9 двоичными символами.

Следовательно, данное предположение неверно. Более того, из приведенных наблюдений стоит сделать вывод о том, что каждый текстовый символ кодируется двоичной последовательностью нефиксированной длины.

Как правило, код переменной длины применяется в случаях, когда некоторые символы встречаются существенно чаще, чем прочие. Для наиболее часто используемых символов используются короткие двоичные цепочки, для редких символов – длинные. В целом подобный подход должен сократить общую длину кода.

Рассмотрим частоту появления текстовых символов:

Символ	Количество повторений символа							Итого
	олово	лов	слово	вол	вес	волос	соловей	
«о»	3	1	2	1		2	2	11
«л»	1	1	1	1		1	1	6
«в»	1	1	1	1	1	1	1	7
«с»			1		1	1	1	4
«е»					1		1	2
«й»							1	1

Из таблицы видно, что наиболее часто используется символ «о», несколько реже – символы «л» и «в»; менее популярен – «с»; самыми редкими символами являются «е» и «й».

Из проведенного частотного анализа можно сделать выводы, что для символов «о», «л», «в» выделяется код минимальной длины. Это предположение подтверждается в словах «олово», «вол» и «лов», которые состоят соответственно из 5, 3, 3 текстовых символов, и кодируются соответственно двоичными цепочками длиной 10, 5, 5.

Если рассмотреть слова «олово» и «лов», то можно увидеть, что они имеют совпадения, которые ниже выделены жирным шрифтом, как в двоичном коде, так и текстовых символах:

0001001000	олово
010010	лов

Из этого наблюдения можно сделать предположение, что двоичный код «00» соответствует символу «о».

Дальнейшее рассмотрение слов «лов» и «олово» подводит к версии, что символ «л» кодируется последовательностью «01», а символ «в» – «10».

л о в
010010

Слово «вол» («100001») подтверждает эту версию.

Сравнение слов «олово» и «слово», различающихся на одну букву, позволяет выделить символ «с», который кодируется «110».

0001001000	олово
11001001000	слово

Подтверждение можно найти в слове «волос».

Далее рассмотрим слово «вес»: кодирующие цепочки символов для «в» и «с» известны, следовательно, для «е» выделяется цепочка «1110».

10 1110 110
в е с

Последним словом, является «соловей», которое дает цепочку кодирования для «й»:

110 00 01 00 10 1110 1111
с о л о в е й

В результате получаем следующую схему кодирования:

Символ	Кодирующая цепочка символов двоичной системы
«о»	00
«л»	01

«в»	10
«с»	110
«е»	1110
«й»	1111

Стоит обратить внимание, что исходное предположение о зависимости длины кодирующих цепочек от частоты использования символов подтвердилось.

Согласно этой схеме кодирования раскодируем требуемую двоичную цепочку: 1011101100100

10 1110 110 01 00
в е с л о

Ответ: весло

Типичные ошибки, допущенные при реализации задачи

1. Решение задачи использованием кода постоянной длины
2. Ошибки невнимательности

Задача 2 (Олимпиада школьников МИСиС, 2011 год)

Как представляется число -3_{10} в одном байте памяти компьютера?

Проверяется

Знание и понимание представления информации в памяти ЭВМ.

Обсуждается

Прямой, обратный и дополнительный коды, их применения в логике работы процессора.

Теоретическое сопровождение темы. Решение

Известный факт, что в одном байте (8 бит) размещается 255 чисел (комбинации от 0000 0000 до 1111 1111). Если под представление (хранение) числа, согласно условию задачи выделяется один байт, то будут задействованы все 8 бит.

Например, для представления чисел 16 и 32 в памяти ЭВМ, несмотря на то, что число 16 требует 5 разрядов, а число 32 – 6 разрядов, отводится одинаковое количество разрядов.

Естественно, имеется ограничение в количестве представляемых чисел.

Особенности организации ЭВМ таковы, что для представления каждого числа отводится заранее определенное количество байтов в зависимости от его типа.

Например, в Паскаль под число, заявленное как тип Byte, отводится только 1 байт, и, как результат, переменная типа Byte не может хранить значение большее 255. Переменная типа Word размером 2 байта может хранить число от 0 до 65535 (2 в 16 степени). Аналогичные типы данных присутствуют в Бейсике.

Название числового типа данных	Длина, байт числового типа данных	Диапазон значений числового типа данных
Byte	1	0...255
Word	2	0...65535

В Си Шарп имеется бОльшее разнообразие аналогичных типов данных:

Название числового типа данных	Длина, байт числового типа данных	Диапазон значений числового типа данных
byte	1	0...255
ushort	2	0...65535
uint	4	0...4294967295
Ulong	8	0...2 ⁶⁴ -1

Возвращаясь к исходному условию задачи, можем кодировать число без знака «3» как «0000 0011» (то есть как двоичный код этого числа). Но в данной задаче надо закодировать число со знаком «-3».

Для возможности представления отрицательных типов один бит выделенных байт (старший) отводится под знак числа: «0» – число положительно; «1» – число отрицательно.

Следовательно, на кодирование абсолютного значения числа отводится на один разряд меньше, что уменьшает максимально возможное кодируемое значение, но сохраняет общее количество кодируемых чисел.

В Паскале и Бейсике это:

Название числового типа данных	Длина, байт числового типа данных	Диапазон значений числового типа данных
ShortInt	1	-128..+127
Integer	2	-32768..+32767
LongInt	4	-2 147 483 648..+2 147 483 647

В Си Шарп это:

Название числового типа данных	Длина, байт числового типа данных	Диапазон значений числового типа данных
sbyte	1	-128..+127
short	2	-32768..+32767
int	4	-2147483648..+ 2147483647
long	8	-2 ⁶³ ..+ 2 ⁶³ -1

Таким образом, число «-3» должно представляться типами данных ShortInt или sbyte. Но запись числа со знаком не подразумевает, что «-3» будет представлено как «1 000 0011» (единица в старшем разряде указывает на знак, далее семь бит абсолютного значения).

Для представления отрицательных чисел, помимо прямого кода разработаны обратный и дополнительный коды.

Обратный код получается инвертированием числовых разрядов прямого кода числа. В частности, число «-3» (код абсолютного значения «0000 0011») в обратном коде будет представлено как «1111 1100» (разряд, указывающий на знак, получается единица).

Дополнительный код получается инвертированием и добавлением единицы в младший разряд. В дополнительном коде число «-3» будет представлено как «1111 1101» ($1111\ 1100 + 0000\ 00001$).

Причина введения обратного и дополнительного кода – при выполнении арифметических операций над числами заменить вычитание сложением. Основным арифметическим узлом в АЛУ процессора является сумматор. Разработав логическую схему сумматора, разработчики пришли к выводу, что проще произвести математические преобразования вычитаемых чисел и заменить вычитание сложением числа в обратном или дополнительном код, чем разрабатывать схему «вычитателя».

Продемонстрируем замену вычитания сложением на примере $9-5 = 4$, предполагая, что под каждое число выделяется один байт.

Прямой код числа 9	0 0 0 0 1 0 0 1
Прямой код числа 5	0 0 0 0 0 1 0 1

Преобразуем вычитаемое число 5 как суммируемое -5

Прямой код числа 5	0 0 0 0 0 1 0 1
--------------------	-----------------

Для операции сложения двух чисел представим -5 в обратном коде

Обратный код числа -5	1 1 1 1 1 0 1 0
-----------------------	-----------------

Сложим числа 9 и $-5_{ок}$, используя классические правила сложения чисел

Прямой код числа 9	+	0 0 0 0 1 0 0 1
Обратный код числа -5		1 1 1 1 1 0 1 0
		1 0 0 0 0 0 1 1

Получается, что произошел перенос в непредусмотренный 9ый разряд. По правилам работы с обратным кодом, единицу переноса следует суммировать с младшим разрядом результата.

Прямой код числа 9	+	0	0	0	0	1	0	0	1
Обратный код числа -5		1	1	1	1	1	0	1	0
	+	0	0	0	0	0	0	1	1
									1
		0	0	0	0	0	1	0	0

Получившийся результат 0000 0100 соответствует числу 4 в десятичной системе счисления.

Как видно из примера, недостатком обратного кода является двойная операция суммирования. Этот недостаток устраняется введением дополнительного кода.

Прямой код числа 9	+	0	0	0	0	1	0	0	1	
Дополнительный код числа -5		1	1	1	1	1	0	1	1	
		1	0	0	0	0	0	1	0	0

Как видно, по результатам суммирования $9 + -5_{\text{дк}}$ также получается единица переноса в 9-ый разряд. По правилам суммирования в дополнительном коде эта единица отбрасывается.

Прямой код числа 9	+	0	0	0	0	1	0	0	1
Дополнительный код числа -5		1	1	1	1	1	0	1	1
		0	0	0	0	0	1	0	0

Результат также получается также 0000 0100.

Ответом данной задачи будет: -3_{10} в одном байте памяти компьютера представляется в обратном коде как «1111 1100_{2ок}» или в дополнительном коде как «1111 1101_{2дк}».

Типичные ошибки, допущенные при реализации задачи

1. Отсутствие представления о способе представления и хранения чисел.
2. Ответ -11 ошибочен и содержит сразу две ошибки: неверное представление числа в байте (опущены 6 символов) и неверное представление знака числа.

Задача 3 (Олимпиада школьников МИСиС, 2011 год)

Какие из перечисленных чисел больше 200_{10} , какие меньше:

- 0101 0111₂,
- 1111 1111₂,
- 1110 0111₂,
- 0001 1000₂,
- 1100 1100₂,

0011 0011₂?

Проверяется

Знание двоичной системы счисления, понимание веса разряда в любой системе счисления.

Обсуждается

Двоичные системы счисления, веса разрядов

Теоретическое сопровождение темы. Решение

Разряды двоичного числа соотносятся с десятичными числами следующим образом (иначе говоря, в таблице указан вес каждого разряда двоичной системы):

Номер разряда	8	7	6	5	4	3	2	1
Вес разряда	128	64	32	16	8	4	2	1

Чтобы перевести 200_{10} в двоичную систему, можно пойти стандартным путем последовательного деления 200 на 2. Но проще использовать веса разрядов (саму таблицу весов полезно выучить наизусть). Продемонстрируем перевод десятичного числа в двоичный код с использованием веса разрядов.

Для представления 200_{10} необходимо и достаточно 8 разрядов, так как 200_{10} больше 128_{10} и меньше 255_{10} .

Начнем формировать двоичное число. Первый разряд из восьми будет единица (если не поставит единицу в старший разряд, то, даже заполнив все младшие разряды единицами, получим максимум 127_{10} , то есть сформировать число 200_{10} не получится):



Записанное двоичное число соответствует 128_{10} . Следовательно, осталось заполнить разряды, обеспечивающие числовое значение, соответствующее $200_{10} - 128_{10} = 72_{10}$.

72_{10} больше, чем 64, следовательно, следующий разряд, имеющие вес 64 также равен единице:



Записанное двоичное число соответствует $(128 + 64)_{10} = 192_{10}$. Следовательно, осталось заполнить разряды, обеспечивающие числовое значение, соответствующее $200_{10} - 192_{10} = 8_{10}$.

8_{10} меньше, чем 32 и меньше, чем 16, следовательно, следующие два разряда не должны вносить свой вес в запись числа и равны нулю:



Следующие разряд имеет вес 8, следовательно, устанавливаем в нем значение единицы:

72_{10} больше, чем 64_{10} , следовательно, следующий разряд, имеющие вес 64 также равен единице:



Записанное число равно $128_{10} + 64_{10} + 8_{10} = 200_{10}$

Получив значение $1100\ 1000_2$, можно, не производя перевод из двоичной системы счисления в десятичную, увидеть, что варианты а) ($0101\ 0111_2$), d) ($0001\ 1000_2$) и f) ($0011\ 0011_2$) заведомо меньше, чем 200_{10} .

Очевидно, что вариант b) ($1111\ 1111_2$, соответствующий 255_{10}) больше, чем 200_{10} .

Проведем для оставшихся вариантов поразрядное сравнение.

Число c) ($1110\ 0111_2 = 128 + 64 + 32 + 0 + 0 + 4 + 2 + 1$) имеет вес в третьем разряде, который не имеет исходное число $200_{10} = 1100\ 1000_2$. Даже не вычисляя полностью его десятичное значение, видно, что $128 + 64 + 32 = 224$ уже больше, чем 200, следовательно, вариант c) – больше, чем число 200_{10} .

Аналогичное сравнения можно провести для варианта e) ($1100\ 1100_2 = 128 + 64 + 0 + 0 + 8 + 4 + 0 + 0$), разряды которого совпадают с разрядами число $200_{10} = 1100\ 1000_2$. в первых пяти позициях. Шестой разряд варианта e) увеличивает вес числа на 4, следовательно, вариант e) также больше, чем $1100\ 1000_2$.

Ответ данной задачи:

Больше 200_{10} : $1111\ 1111_2$, $1110\ 0111_2$, $1100\ 1100_2$

Меньше 200_{10} : $0101\ 0111_2$, $0001\ 1000_2$, $0011\ 0011_2$

Альтернативным вариантом решения может быть перевод всех чисел в десятичную систему счисления. При этом имеет смысл также воспользоваться таблицей весов разрядов. Но данный вариант требует больше временных затрат на его реализацию.

Типичные ошибки, допущенные при реализации задачи

1. Математические ошибки
2. Преобразование всех чисел из двоичной системы счисления в десятичную. При этом задача может быть решена правильно, но с потерей большого количества времени.

Задача 4 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 10 класс)

Определите, какой минимальный объем потребуется для хранения закодированного изображения приведенного рисунка в двоичном коде в битах.

Проверяется

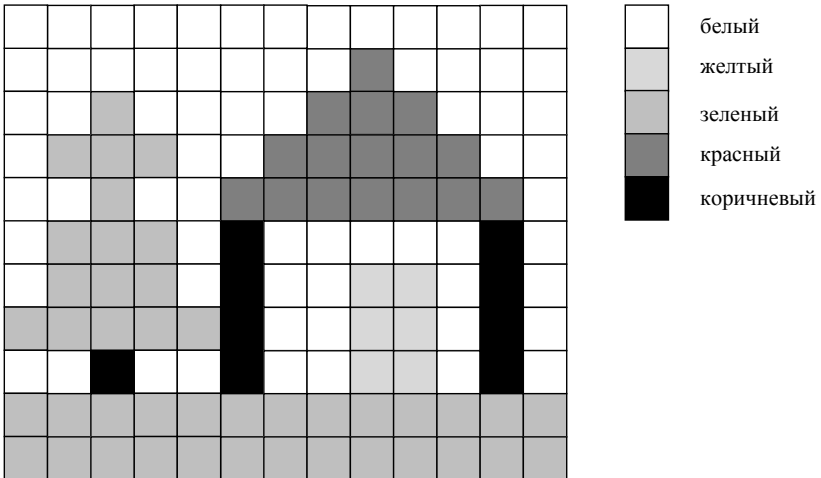
Навыки кодирования информации.

Обсуждается

Кодирование растровой графической информации.

Теоретическое сопровождение темы. Решение

В данной задаче приведено растровое изображение, в котором рисунок состоит из множества точек – пикселей. Каждая точка имеет свой цвет.



Соответственно, для кодирования такого изображения требуется описать цвет каждой точки.

Общее количество точек рисунка (13 по ширине, 11 по высоте) равно $13 \cdot 11 = 143$.

В картинке используется 5 цветов. Надо определить, какое количество разрядов двоичной системы потребуется для кодирования пяти цветов.

Для кодирования k различных объектов (точек, символов) требуется найти такую минимальную степень двойки n , при которой $2^n \geq k$. В данной задаче для кодирования пяти символов имеем 2^3 больше пяти. То есть требуется три разряда.

Для наглядности сопоставим каждый цвет с числом:

Число	Цвет рисунка	Двоичный код числа
0	Белый	000
1	Желтый	001
2	Зеленый	010
3	Красный	011
4	Коричневый	100

6	Не используется	101
7	Не используется	110
8	Не используется	111

Отведя три разряда для записи о цвете каждой точке, получаем три неиспользуемых комбинации, что является нормальной ситуацией в подобных задачах: если отвести всего два разряда, то для одного цвета не хватит числовой комбинации.

При выбранной схеме кодирования кодированное изображение будет представляться в памяти следующей цепочкой двоичных символов:

000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	011	000	000	000	000	000
000	000	010	000	000	000	000	011	011	011	000	000	000	000
000	010	010	010	000	000	011	011	011	011	011	000	000	000
000	000	010	000	000	011	011	011	011	011	011	011	000	000
000	010	010	010	000	100	000	000	000	000	000	000	100	000
000	010	010	010	000	100	000	000	001	001	000	000	100	000
010	010	010	010	010	100	000	000	001	001	000	000	100	000
000	000	100	000	000	100	000	000	001	001	000	000	100	000
010	010	010	010	010	010	010	010	010	010	010	010	010	010
010	010	010	010	010	010	010	010	010	010	010	010	010	010

	белый	000
	желтый	001
	зеленый	010
	красный	011
	коричневый	100

Минимальный объем памяти в битах рассчитывается как $13 \cdot 11 \cdot 3 = 429$ бит.

Типичные ошибки, допущенные при реализации задачи

1. Неправильное определение количество разрядов для кодирования пяти цветов как результат незнания темы кодирование графической растровой информации.
2. Математические ошибки.

Задача 5 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 10 класс)

Вычислите и получите результат в двоичной системе счисления $10011000_2 / 4_{10}$

Проверяется

Знание и понимание систем счисления. Математические навыки.

Обсуждается

Системы счисления. Основание систем счисления.

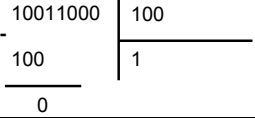
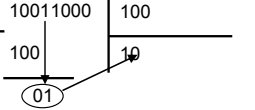
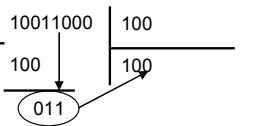
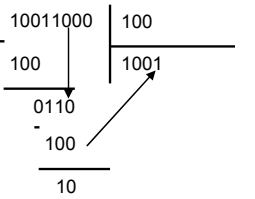
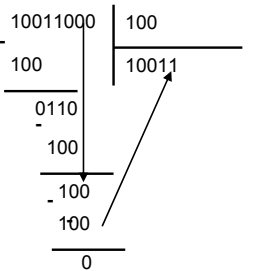
Теоретическое сопровождение темы. Решение

Наиболее очевидным и наименее удачным вариантом решения данной задачи является преобразование делимого из двоичной системы в десятичную (получается 152_{10}), деление его на 4 (получается 38_{10}) и преобразование результата в двоичную систему (получается 100110_2). Детальное описание данного решения не приводится в силу его очевидности.

Минусом данного решения является значительная потеря времени на математические расчеты и большая вероятность ошибки.

Другим вариантом является преобразование делителя в двоичную систему $4_{10} = 100_2$.

Естественно, далее можно решить данную задачу, применяя классическое деление «в столбик»:

<p>При делении 10011000_2 на 100_2 (далее указание на двоичную систему опускаем) выделяем первые три символа 10011000, из которых вычитаем делитель; первым разрядом результата записывается 1</p>	
<p>Результат вычитания – ноль, сносим вниз следующий разряд 10011000; 01 меньше делителя, записываем следующим разрядом результата 0</p>	
<p>Сносим вниз следующий разряд 10011000; 011 меньше делителя, записываем следующим разрядом результата 0</p>	
<p>Сносим вниз следующий разряд 10011000; 0110 можно делить далее: записываем следующим разрядом результата 1, вычитаем 100 из 0110; получаем в остатке 10</p>	
<p>Сносим вниз следующий разряд 10011000; 100 можно делить далее: записываем следующим разрядом результата 1, вычитаем 100 из 100; получаем в остатке 0</p>	

Оставшийся ноль исходного числа 10011000_2 переносим как последний разряд результата. Ответ 100110_2

$$\begin{array}{r}
 10011000 \quad | \quad 100 \\
 - 100 \quad \quad | \quad 100110 \\
 \hline
 0110 \\
 - 100 \\
 \hline
 100 \\
 - 100 \\
 \hline
 0
 \end{array}$$

Данное решение требует меньше времени на преобразования чисел из одной системы счисления в другую систему, и обратно. Но также имеет вероятность ошибки при делении в столбик в двоичной системе.

Самое краткое и лаконичное решение реализуется при анализе результатов перевода делителя в двоичную систему счисления: $10011000_2 / 100_2 = ?$

Хорошее знание позиционных систем счисления подсказывает, что умножение на основание системы счисления производится переносом запятой (отделяющую целую часть от дробной) на одну позицию вправо. Например, в десятичной системе счисления

$$98 * 10 = 980$$

$$100 * 10 = 1000.$$

Последний пример работает и в двоичной (и во всех прочих) системе счисления:

$$4 * 2 = 8 \quad (4_{10} = 100_2, \quad 2_{10} = 10_2, \quad 8_{10} = 1000_2, \quad 100_2 * 10_2 = 1000_2)$$

А при делении на основание системы счисления требуется перенести запятую (отделяющую целую часть от дробной) на одну позицию влево. Это правило работает как в десятичной системе (например, $12300/10 = 1230$), так и во всех прочих системах счисления.

Таким образом, для деления 10011000_2 на 100_2 достаточно убрать два нуля справа у делимого. Результат 10011000 .

Ответ: 100110_2

Типичные ошибки, допущенные при реализации задачи

1. Поиск решения переводом чисел в десятичную систему счисления, деления в десятичной системе счисления и переводом результата в двоичную систему счисления. При подобном подходе возрастает вероятность математических ошибок и тратится слишком много времени.

2. Арифметические ошибки при выполнении деления в двоичной системе счисления. $10011000_2 / 100_2$. Данный вариант решения зада-

чи возможен и не является ошибкой при соблюдении общепринятых правил деления «в столбик».

Задача 6 (Заключительный этап олимпиады школьников «Ломоносов-2015» по информатике (5–9 классы))

Переведите число 2211201122110201 из троичной системы счисления в систему счисления с основанием 27. В качестве цифр используйте десятичные цифры и заглавные латинские буквы.

Проверяется

Понимание сути любой позиционной системы счисления.

Обсуждается

Соответствие между системами счисления в случае, если основание одной системы счисления является степенью другой

Теоретическое сопровождение темы. Решение

Для быстрого и простого решения данной задачи достаточно увидеть, что 27 (основание новой системы счисления) является третьей степенью тройки (основание текущей системы счисления представления числа): $3^3 = 27$. Поэтому, аналогично предыдущей (четвертой) задаче, преобразование числа из одной системы счисления в другую можно выполнить без арифметических расчетов: каждые три разряда исходного числа троичной системы счисления кодируются одним разрядом числа 27-ричной системы счисления в соответствии со следующей таблицей.

Запись числа в троичной системе счисления			Запись числа в 27-ричной системе счисления	Запись числа в 10-тичной системе счисления
3-ий разряд	2-ой разряд	1-ый разряд		
0	0	0	0	0
0	0	1	1	1
0	0	2	2	2
0	1	0	3	3
0	1	1	4	4
0	1	2	5	5
0	2	0	6	6
0	2	1	7	7
0	2	2	8	8
1	0	0	9	9
1	0	1	A	10
1	0	2	B	11
1	1	0	C	12
1	1	1	D	13
1	1	2	E	14
1	2	0	F	15
1	2	1	G	16

Запись числа в троичной системе счисления			Запись числа в 27-ричной системе счисления	Запись числа в 10-тичной системе счисления
3-ий разряд	2-ой разряд	1-ый разряд		
1	2	2	H	17
2	0	0	I	18
2	0	1	J	19
2	0	2	K	20
2	1	0	L	21
2	1	1	M	22
2	1	2	N	23
2	2	0	O	24
2	2	1	P	25
2	2	2	Q	26

Запишем исходное число 2211201122110201, разбив по три разряда, начиная с младшего:

002 211 201 122 110 201

Теперь каждую группу символов перекодируем одним символом 27-ричной системы счисления (в таблице эти группы выделены цветом):

2 M J H C J

Это и будет ответ $2MJHCS_{27}$

Типичные ошибки, допущенные при реализации задачи

1. Математические ошибки
2. Преобразование заданного числа с десятичную систему счисления с последующим переводом в 27-ричную: задача может быть решена правильно, но с потерей большого количества времени.

Задача 7 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 9 класс)

Напишите результат работы программы



Проверяется

Навыки алгоритмизации, умения работать с блок-схемами, анализировать алгоритмы, знание логических типов данных, умение работать с логическими переменными.

Обсуждается

Типы данных. Операции, допустимые и недопустимые для различных типов данных. Логические типы данных. Логические переменные.

Теоретическое сопровождение темы. Решение

Наиболее часто используемыми типами данных являются типы, работающие с целыми (integer или long) или дробными (single или double) числами. Но, наряду с перечисленными типами, существуют типы, работающие с символами (char), со строками (string), с логическими значениями (bool или boolean) и многие другими.

Каждый тип данных определяет, сколько памяти требуется отвести под переменную данного типа, а также – какие операции допустимы над данными заявленного типа, и как эти операции будут выполняться.

Например, операция сложения допустима как для данных числового, так и для данных строкового типа, но не допустима для сим-

вольного типа. Но для различных типов данных она будет выполняться по-разному.

Сложение чисел **13 + 42** в результате даст число **55**. Числа складываются по классическим правилам математики.

Сложение строк «**тринадцать**» + «**сорок два**» в результате даст строку «**тринадцать****сорок два**». Отсутствие пробела не является опечаткой: для строковых типов данных операция сложения выполняется как формирование новой строки последовательной записью первого и второго слагаемого.

Стоит отметить, что для строковых типов данных не выполняется правило «от перемены мест слагаемых сумма не меняется». Сложение строк «**сорок два**» + «**тринадцать**» в результате даст строку «**сорок дватринадцать**».

Но операция вычитания, допустимая для чисел (например, **13–42 = –29**) логически недопустима, а потому – не определена для строковых данных.

В данной задаче X и Y описываются как логические переменные. Логический тип данных работает с переменными, которые принимают только значения «ЛОЖЬ» или «ИСТИНА», что позволяет отводить под эти переменные небольшое количество памяти (по сравнению с прочими типами данных). Иногда значение «ИСТИНА» записывается как “True” (английский вариант слова «истина») или как «1». А значение «ЛОЖЬ» записывается как “False” (английский вариант слова «ложь») или как «0».

Для логических переменных допустимы операции присваивания значения («ЛОЖЬ» или «ИСТИНА»), логического сложения (иначе называемого дизъюнкцией), логического умножения (иначе называемого конъюнкцией), и отрицания.

Рассмотрим выполнение операций над следующими логическими переменными с заданными исходными значениями X = ИСТИНА и Y=ИСТИНА.

Операция отрицания записывается как **НЕ X**, или как **no X**, или как **¬X**, или \bar{X} . Результатом команды **Y= ¬X** будет значение Y=ЛОЖЬ.

Операция логического сложения записывается как **X ИЛИ Y**, или **X or Y**, или **X ∨ Y**, или **X + Y**. Результатом команды **X and Y** определяется (в соответствии с логикой) по таблице истинности

X	Y	X and Y
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ИСТИНА

ИСТИНА	ЛОЖЬ	ИСТИНА
ИСТИНА	ИСТИНА	ИСТИНА

В данном случае, при $X = \text{ИСТИНА}$ и $Y = \text{ЛОЖЬ}$, $X \text{ ИЛИ } Y = \text{ИСТИНА}$.

Операция логического умножения записывается как $X \text{ И } Y$, или $X \text{ and } Y$, или $X \wedge Y$, или $X * Y$. Результатом команды $X \text{ or } Y$ определяется (в соответствии с логикой) по таблице истинности

X	Y	X or Y
ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
ЛОЖЬ	ИСТИНА	ЛОЖЬ
ИСТИНА	ЛОЖЬ	ЛОЖЬ
ИСТИНА	ИСТИНА	ИСТИНА

В данном случае, при $X = \text{ИСТИНА}$ и $Y = \text{ЛОЖЬ}$, $X \text{ И } Y = \text{ЛОЖЬ}$.

Допустимые операции имеют следующий приоритет выполнения:

1. Отрицание
2. Умножение
3. Сложение

Например, определение значения выражения $X \text{ ИЛИ НЕ } Y \text{ И НЕ } X$ будет выполняться в следующем порядке

1. Вычислить НЕ Y (ИСТИНА)
2. Вычислить НЕ X (ЛОЖЬ)
3. Вычислить НЕ Y И НЕ X (ЛОЖЬ)
4. Вычислить X ИЛИ НЕ Y И НЕ X (ИСТИНА)

Использование скобок меняет порядок выполнения операций. Например, $(X \text{ ИЛИ НЕ } Y) \text{ И НЕ } X$, будет вычисляться в следующем порядке

1. Вычислить НЕ Y (ИСТИНА)
2. Вычислить НЕ X (ЛОЖЬ)
3. Вычислить $(X \text{ ИЛИ НЕ } Y)$ (ИСТИНА)
4. Вычислить $(X \text{ ИЛИ НЕ } Y) \text{ И НЕ } X$ (ЛОЖЬ)

Разберем блок-схему, предложенную в задаче.

Оператор $X=3=4$ – оператор присваивания логической переменной X значения. Призываемое значение ($3=4$) предварительно должно быть вычислено. Естественно, число 3 не равно числу 4, следовательно, значение логической переменной X равно ЛОЖЬ.

Оператор $Y=7>6$ – оператор присваивания логической переменной Y значения. Призываемое значение ($7 > 6$) предварительно должно быть вычислено. Число 7 больше, чем 6, следовательно, значение логической переменной Y равно ИСТИНА.

Далее, согласно блок-схеме, последовательно требуется вывести четыре значения:

- X (значение которого ЛОЖЬ),

- Y (значение которого ИСТИНА),
- $(X \text{ or } Y) \text{ and } Y$; для вычисления данного выражение сначала определяется $(X \text{ or } Y) = \text{ЛОЖЬ ИЛИ ИСТИНА} = \text{ИСТИНА}$; далее ИСТИНА and $Y = \text{ИСТИНА and ИСТИНА} = \text{ИСТИНА}$
- $(Y \text{ and } X) \text{ or } Y$; для вычисления данного выражение сначала определяется $(Y \text{ and } X) = \text{ИСТИНА И ЛОЖЬ} = \text{ЛОЖЬ}$; далее ЛОЖЬ or $Y = \text{ЛОЖЬ or ИСТИНА} = \text{ИСТИНА}$

Ответ: результатом работы программы является последовательный вывод на экран «ложь, истина, истина, истина».

Альтернативный вариант ответа «0, 1, 1, 1».

Типичные ошибки, допущенные при реализации задачи

1. Отсутствие опыта работы с логическими типами данных.
2. Неумение работать с блок-схемами, и/или незнание графического представления оператора вывода.
3. Как результат первых двух пунктов, нахождение единственного варианта ответа, например, «ложь».

Задача 8 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 11 класс, финальный тур)

В таблице приведены входные и выходные данные некоторого автомата.

На входе	На выходе
925784, 2, 5	9
925784, 2, 4	94
925784, 3, 3	924
925784, 7, 1	error
1329710, 2, 5	10
925784, 1, 5	4
1329710, 2, 8	error
1329710, 2, 4	110

Укажите, что будет на выходе автомата, если на вход подать

На входе	На выходе
1329710, 4, 3	
925784, 8, 5	
925784, 4, 1	

Проверяется

Умение выявлять закономерности, представление о функциях и параметрах функций, понимание строковых и символьных типов данных.

Обсуждается

Анализ данных. Типы данных. Способы решения задач на выявление закономерностей.

Теоретическое сопровождение темы. Решение

Данная задача является усложненным продолжением предыдущей задачи. В ней также имеется некоторое устройство, которое принимает данные на вход и для каждого набора входных данных дает некий результат.

Для некоторых входных наборов данных работа автомата является невозможной и вместо ответа автомат дает сообщение об ошибке «error».

Для решения данной и подобной ей задач рекомендуется руководствоваться следующими правилами:

- В исходных данных задачи присутствует информация, необходимая для решения задачи. Иными словами – в задаче приведено достаточное количество входных данных, чтобы подметить закономерность, по которой работает автомат.

- Важно понять, какие входные параметры и почему, приводят к ошибкам. В ряде случаев закономерность проще определить, отталкиваясь от ошибочных вариантов входных данных.

- Помнить, что не всегда ряд символов-цифр надо рассматривать как число. Запись «123456» может являться как числом «сто двадцать три тысячи четыреста пятьдесят шесть», так и строкой из шести символов-цифр «123456».

Параллельно анализировать результаты при сходных входных параметрах.

Рассмотрим входные параметры и их результаты, чтобы определить недопустимые значения параметров. Первая ошибка возникает при «исправном» первом параметре (подобное значение первого параметра не давало сбоя ранее), но при значениях второго и третьего параметра «7, 1». Аналогично, вторая ошибка не зависит от первого параметра, но значения второго и третьего параметров «2, 8» невозможны. Последняя строка входных данных показывает, что второй параметр «2» разрешен. Можно сделать допущение, что третий параметр «8» приводит к ошибке.

На входе	На выходе
925784, 2, 5	9
925784, 2, 4	94
925784, 3, 3	924
925784, 7, 1	error
1329710, 2, 5	10

925784, 1, 5	4
1329710, 2, 8	error
1329710, 2, 4	110

Но в первой ошибке третий параметр «1», из чего можно сделать допущение, что причина не только в значении третьего параметра, но в комбинации второго и третьего параметров. Также можно заметить, что автомат не может выполнить свою работу при относительно «больших» значениях второго или третьего параметров.

Далее стоит проанализировать сходные комбинации и их результаты. В первом и втором варианте входных заданий входные параметры почти совпадают: отличие только в третьем параметре.

Также интересно рассмотреть первый и шестой варианты входных данных, имеющие отличие во втором параметре.

На входе	На выходе
925784, 2, 5	9
925784, 2, 4	94
925784, 3, 3	924
925784, 7, 1	error
1329710, 2, 5	10
925784, 1, 5	4
1329710, 2, 8	error
1329710, 2, 4	110

Аналогично стоит рассмотреть пятый и восьмой варианты, имеющие отличие в третьем параметре.

На входе	На выходе
925784, 2, 5	9
925784, 2, 4	94
925784, 3, 3	924
925784, 7, 1	error
1329710, 2, 5	10
925784, 1, 5	4
1329710, 2, 8	error
1329710, 2, 4	110

Внимательное наблюдение позволит заметить *текстовое*, а не арифметическое, сходство выходных результатов при сходных входных. Более того, в записи результата используются только символы первого параметра. В этом случае дальнейшие умозаключения стоит выводить исходя из вывода, что первый параметр – текстовая строка из символов-цифр.

Сделанное наблюдение находит подтверждение во всех прочих вариантах: результат состоит только из символов первого параметра.

Анализ ошибочных вариантов также косвенно указывают, что второй и третий параметр связаны с позициями символов в строке: если там находятся слишком большие значения, то работа со строкой невозможна.

Повторное попарное сравнение первого со вторым варианта, и пятого с восьмым, показывает, что, чем меньше третий параметр, тем длиннее результат. Это закономерность косвенно подтверждается третьим и шестым вариантом входных данных.

На входе	На выходе
925784, 2, 5	9
925784, 2, 4	94
925784, 3, 3	924
925784, 7, 1	Error
1329710, 2, 5	10
925784, 1, 5	4
1329710, 2, 8	Error
1329710, 2, 4	110

Из этого факта можно сделать вывод, что третий параметр указывает, сколько символов первого параметра надо *удалить* для получения результата.

Итак, выстраивается логическая цепочка: результат формируется из символов первого параметра, удалением некоторого количества из него символов. Напрашивается вопрос: какие символы удалять?

Если вернуться к первому варианту, то видим, что из строки «925784» на выходе остается только «9», то есть только один, первый символ. Исходно в строке было 6 символов, удалено 5 символов, остался один. Значение второго параметра – «2», удаление произведено, начиная со второго символа. Следовательно, второй параметр – позиция, начиная с которой надо удалять количество символов, указанных третьим параметром.

Ошибка возникает, если второй параметр указывает на несуществующую позицию, как в случае «925784, 7, 1»: в строке «925784» всего 6 символов, удалять, начиная с седьмого невозможно. Или, если надо удалять символов больше, чем существует, как в случае «1329710, 2, 8»: в строке «1329710», длиной семь символов, начиная со второй позиции удалить более, чем шесть символов, невозможно.

Любую подмеченную закономерность обязательно *надо проверить* на всех входных вариантах. Если какой-либо вариант не соответствует подмеченной закономерности, то она неверна.

Ответ: Автомат выводит оставшиеся символы в исходной строке (первый параметр), если из исходной убрать указанные символы, на-

чиная с заданной позиции (второй параметр) указанным количеством (третий параметр).

На входе	На выходе
1329710, 4, 3	1320
925784, 8, 5	error
925784, 4, 1	92584

Типичные ошибки, допущенные при реализации задачи

1. Поиск только арифметических закономерностей для определения работы автомата.

2. Отсутствие объяснения работы автомата. В некоторых случаях задача бывает частично решена, но содержит ошибки в ответе, по которым невозможно определить ход мыслей автора работы. Рекомендуется включать в ответ объяснения работы автомата.

Задача 9 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 10 класс)

Робот-исполнитель снабжен камерой наблюдения и умеет шагать по плоскости; при перемещении на плоскости остается след. Робот-исполнитель имеет следующий набор команд:

Вперед (s) – по этой команде он перемещается на s шагов вперед, «куда камера смотрит»

Поворот (m) – по этой команде камера поворачивается на m градусов по часовой стрелке

Повтори (k)

(

Команда 1

Команда 2

...

Команда n

) – обеспечивает повторение команд с первой по n-ую k раз.

Роботу задали следующую программу:

a=1

k=1

Повтори (2)

(

Повтори (3)

(

Вперед (a)

Повтори (2)

(

Поворот (120)

Вперед (a)

)

a=a+k

)
k=-k
)

Изобразите, что нарисует робот

Проверяется

Навыки алгоритмизации, умение следовать алгоритму, учитывать циклическое изменение параметра.

Обсуждается

Логика работы алгоритма.

Теоретическое сопровождение темы. Решение

При решении данной задачи требуется продемонстрировать умение следовать алгоритму в соответствии с предложенной системой команд вне зависимости от используемого языка программирования.

Одним из вариантов выполнения подобных задач является пошаговое выполнение всех инструкций от первой до последней команды. Но подобный подход осложнен тем, что при большом количестве команд (более семи) человеческий мозг, в отличие от компьютера, может упустить какую-либо команду или значение какого-либо параметра, что приведет к неверному ответу.

Альтернативным вариантом решения является изучение структуры предложенного алгоритма для выявления закономерностей (циклов) и дальнейшего использования выявленных закономерностей для получения ответа.

Оценим структуру задачи:

a=1

k=1

Повтори (2)

(
Повтори (3)

(
Вперед (a)

Повтори (2)

(
Поворот (120)

Вперед (a)

)

a=a+k

)

k=-k



)

Как видно, в задаче используется три цикла; причем имеет место вложение циклов.


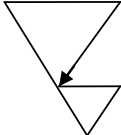
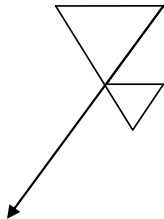
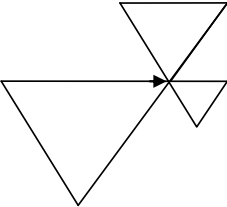
Рассмотрим, что выполняет каждый из циклов, учитывая, что значение параметров a и k в начале программы устанавливается равным 1. Начнем с внутреннего цикла (стрелка на рисунке указывает направление камеры):

Фрагмент кода	Результат выполнения	Описание результата
Поворот (120)		Поворот камеры на 120 градусов по часовой стрелке
Вперед (a)		Продвижение робота вперед на один шаг
Поворот (120)		Поворот камеры на 120 градусов по часовой стрелке
Вперед (a)		Продвижение робота вперед на один шаг

Результатом работы внутреннего цикла является рисование «галочки» с длинной стороны a и смена направления камеры на 240 градусов относительно исходного направления:

До работы внутреннего цикла	По завершению работы внутреннего цикла
	

Рассмотрим следующий цикл, повторяющийся три раза:

Фрагмент кода	Результат выполнения	Описание результата
Вперед (a)		Продвижение робота вперед на один шаг
Повтори (2) (Поворот (120) Вперед (a))		Рисование «галочки» с длиной стороны в один шаг и со сменой направления камеры в 240 градусов
$a=a+k$	$a=1+1=2$	Вычисление и присвоение нового значения переменной a, увеличением старого значения a на величину k
Вперед (a)		Продвижение робота вперед на a шагов (a увеличено на предыдущем шаге)
Повтори (2) (Поворот (120) Вперед (a))		Рисование «галочки» с длиной стороны 2 шага и со сменой направления камеры в 240 градусов
$a=a+k$	$a=2+1=3$	Вычисление и присвоение нового значения переменной a, увеличением старого значения a на величину k
Вперед (a)		Продвижение робота вперед на a шагов
Повтори (2) (Поворот (120) Вперед (a))		Рисование «галочки» с длиной стороны 3 шага и со сменой направления камеры в 240 градусов
$a=a+k$	$a=3+1=4$	Вычисление и присвоение нового значения переменной a, увеличением старого значения a на величину k

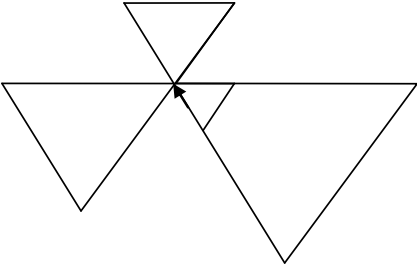
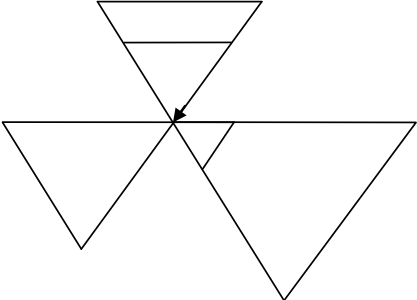
Видно, что на каждом шаге цикла прорисовывается равносторонний треугольник со стороной a , после чего значение a изменяется на k , таким образом, размер следующего треугольника отличается от предыдущего.

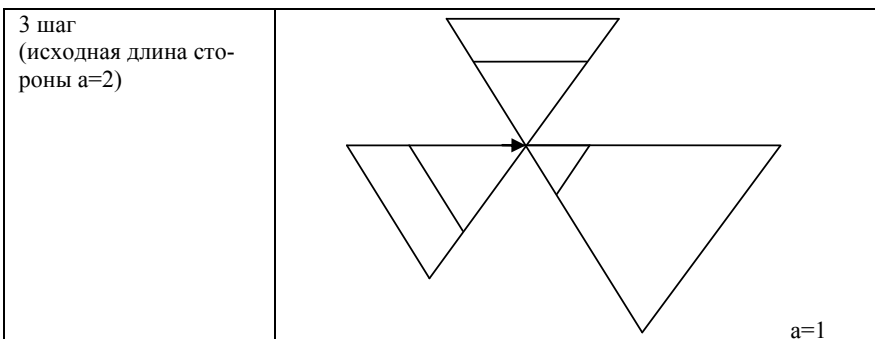
По окончании работы цикла направление камеры совпадает с исходным направлением. Значение параметра a равно 4.

Рассмотрим внешний цикл: он повторяется дважды. Следовательно, уже разобранный последовательность действий будет выполнена дважды. Но, по завершению каждого шага внешнего цикла меняется значение параметра k : $k = -k$. Исходное значение $k = 1$; следовательно, при каждом шаге внешнего цикла значение k будет менять знак.

Параметр k (как уже было показано) влияет на изменение размера стороны треугольника.

При первом шаге внешнего цикла сторона каждого следующего треугольника становилась на один шаг больше предыдущего. При втором (и последнем) шаге внешнего цикла размер каждого следующего треугольника будет уменьшаться на один шаг ($a = a + k$, то есть $a = a - 1$). На втором шаге первый рисуемый треугольник будет иметь длину стороны 4 шага; второй – 3 шага, третий – 2 шага:

	Результат выполнения
1 шаг (исходная длина стороны $a=4$)	
2 шаг (исходная длина стороны $a=3$)	

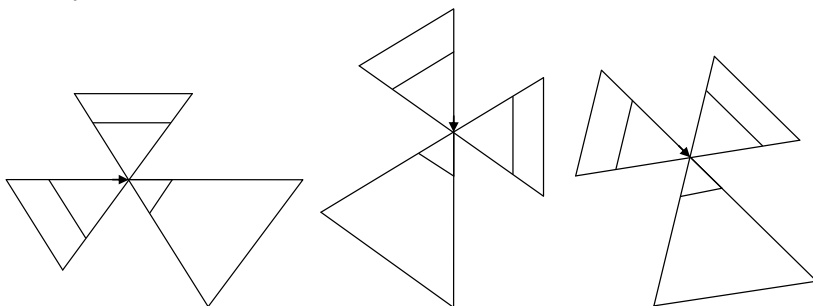


Последняя нарисованная фигура является решением данной задачи. Детальное пошаговое выполнение всех шагов программы приведено для наглядности решения.

Типичные ошибки, допущенные при реализации задачи

1. Ответ о невозможности определения результата работы робота из-за отсутствия информации о исходном направлении камеры. В задании действительно не указано исходное направление камеры, но исходное направление камеры влияет только на угол размещения фигуры конечного результата, но не оказывает никакого влияния на рисуемую фигуру.

Например, принципиально, какое изображение из нижеприведенных будет дано в качестве ответа:



2. Непонимание алгоритма задачи.

3. Ошибки невнимательности, возникающие при ручном пошаговом выполнении алгоритма, которых можно избежать, проведя предварительный анализ кода.

Задача 10 (Задача контрольной МФТИ «Выходи решать» 2017)

Последовательности символов создаются по следующему правилу. Нулевая строка: «ABCDEFGH». Каждая из последующих строк создается следующим действием: дважды подряд записывается пре-

дыдущая строка и в конец дописывается очередная буква английского алфавита, начиная с А. Вот строки с номерами 1–3, созданные по этому правилу:

1. ABCDEFGHABCDEFGHА
2. ABCDEFGHABCDEFGHААBCDEFGHABCDEFGHАВ
3. ABCDEFGHABCDEFGHААBCDEFGHABCDEFGHАВАВСDE
FGHABCDEFGHААBCDEFGHABCDEFGHАВС

Сколько раз в строке с номером 8 встретится символ «D»? В ответе укажите целое число.

Проверяется

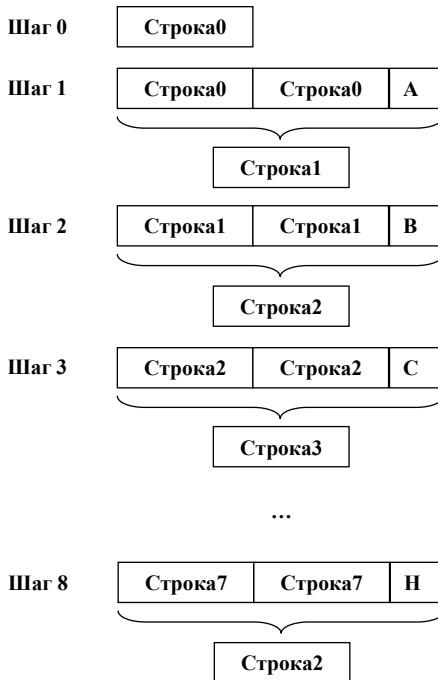
Математические навыки, умение анализировать имеющуюся информацию, выявлять зависимости, алгоритмизировать найденную зависимость и получать результаты согласно найденному алгоритму.

Обсуждается

Анализ данных, математическая реализация алгоритма.

Теоретическое сопровождение темы. Решение

Заданный алгоритм формирования строки сам по себе не сложен: на каждом *i*-ом шаге исходная строка записывается дважды и в конец к ней дописывается *i*-ый символ алфавита:



Но при этом надо учитывать, что при такой схеме размер строки увеличивается практически в арифметической прогрессии. Если длина нулевой строки равна восьми, то длина первой строки будет равна $8+8+1=17$; длина второй – $17+17+1=35$; и т.д.

Номер строки	Длина строки
0	8
1	17
2	35
3	71
4	143
5	287
6	575
7	1151
8	2303

То есть длина каждой последующей строки определяется как удвоенная длина предыдущей плюс один.

Как видно из приведенной таблицы, длина восьмой строки равна 2303, что делает практически невозможным решение задачи ручным формированием восьмой строки и подсчет в ней количества повторения символа «D». Даже при аккуратном и безошибочном выполнении этого задания на решение данной задачи подобным способом понадобится все время, отведенное на олимпиаду.

Следовательно, надо определить схему подсчета количества вхождения требуемого символа в требуемой строке.

Внимательно рассмотрим механизм формирования строки с учетом количества входящих в строку символов:

Шаг	Добавляемый символ	Результирующая строка
0		ABCDEFGH
1	A	ABCDEFGH ABCDEFGH ABCDEFGH A
2	B	ABCDEFGHABCDEFGH ABCDEFGHABCDEFGH ABCDEFGHABCDEFGH AB
3	C	ABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGH ABC
4	D	ABCDEFGHABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGHABCDEFGH ABCDEFGHABCDEFGHABCDEFGHABCDEFGH ABCD
...		

Результаты анализа будем заносить с нижерасположенную таблицу.

В нулевой строке все восемь исходных символов встречаются по одному разу. Первая строка формируется дублированием нулевой и в конец ей приписывается первый символ «А». Следовательно, все восемь символов должны быть удвоены, а первый символ – еще и увеличен на единицу.

Вторая строка формируется дублированием первой строки, следовательно, во второй строке все символы удваиваются, но второй символ также увеличивается на единицу. Третья строка формируется удваиванием второй, а третий символ третьей строки еще и увеличивается на единицу и т.д.

Наблюдается закономерность: чтобы подсчитать количество вхождений i -го символа в j -ую строку исходное количество символов (равное единице) надо удваивать j раз; но, если $j > i$, то в момент $i = j$, удвоенное количество символов надо увеличить на единицу.

строка символ	0	1	2	3	4	...
A	1	$1*2+1=3$	$3*2=6$	$6*2=12$	$12*2=24$...
B	1	$1*2=2$	$2*2+1=5$	$5*2=10$	$10*2=20$...
C	1	$1*2=2$	$2*2=4$	$4*2+1=9$	$9*2=18$...
D	1	$1*2=2$	$2*2=4$	$4*2=8$	$8*2+1=17$...
E	1	$1*2=2$	$2*2=4$	$4*2=8$	$8*2=16$...
F	1	$1*2=2$	$2*2=4$	$4*2=8$	$8*2=16$...
G	1	$1*2=2$	$2*2=4$	$4*2=8$	$8*2=16$...
H	1	$1*2=2$	$2*2=4$	$4*2=8$	$8*2=16$...

Например, символ «F» (его порядковый номер – 6) входит в третью строку $1*2*2*2=8$ раз. Для символа «F» порядковый номер больше номера строки; в формуле используется только трехкратное удвоение.

А символ «B» (порядковый номер – 2) входит в третью строку $(1*2*2+1)*2=10$ раз. Для символа «B» порядковый номер меньше номера строки, поэтому при втором удвоении к полученному числу добавляется единица.

Таким образом, чтобы определить количество вхождений символа «D» в восьмую строку, надо удваивать исходное количество символа «D» (равное единице); при четвертом удвоении добавить единицу:

Строка	Количество вхождений символа «D»
0	1
1	$1*2=2$
2	$2*2=4$
3	$4*2=8$
4	$8*2+1=17$

5	$17*2=34$
6	$34*2=68$
7	$68*2=136$
8	$136*2=272$

Ответ: 272

Типичные ошибки, допущенные при реализации задачи

1. Непонимание алгоритма формирования строки
2. Решение методом составления строки и пересчёта символов «вручную».
3. Арифметические ошибки

Задача 11 (10 класс, Роботрон, отборочный тур 2016 год)

Сколько значащих нулей в двоичной записи шестнадцатеричного числа 12AC0? (Ноль называется значащим, если удаление его из записи числа ведёт к изменению значения числа). Приведите решение задачи.

Напишите алгоритм (возможно в виде блок-схемы) определения количества значащих нулей в двоичной записи числа.

Проверяется

- При решении данной задачи требуется продемонстрировать
- знание систем счисления, в частности, сопоставления двоичной системы с шестнадцатеричной;
 - математические навыки,
 - знание типов данных,
 - способность определения типа обрабатываемых данных,
 - умение работать со строковыми переменными
 - и навыки грамотного составления алгоритма для решения задачи.

Обсуждается

Шестнадцатеричная система счисления, соответствие между системами счисления в случае, если основание одной системы счисления является степенью другой.

Кодирование текстовой информации. Простейшие операции с текстовыми переменными.

Приемы программирования; целесообразность использования оператора goto и способы написания альтернативного кода без использования goto.

Теоретическое сопровождение темы. Решение.

Первая часть задачи проверяет знание соответствия шестнадцатеричной и двоичной систем счисления. Естественно, можно перевести заданное шестнадцатеричное число в десятичное, а далее – в двоичное, но этот способ решения потребует слишком много времени и сопряжен с большим количеством ошибок расчета.

Для быстрого и легкого решения первой части задачи надо знать, что двоичная система счисления в настоящее время представляет особую важность по причине того, что все данные в современных вычислительных системах представляются в двоичном коде. Двоичный код, в свою очередь, обусловлен физической реализацией вычислительных систем (заряд есть/ заряда нет; транзистор открыт/ транзистор закрыт). Но двоичное представление информации неудобно для человеческого восприятия. Для удобства двоичные данные преобразуются в десятичные.

Классический пример: IP-адрес 4-ой версии протокола TCP/IP. IP-адрес представляет собой 32-х битное число, которое для представления пользователю разбивается на 4 октета, и каждый октет записывается в десятичном виде. Например: IP-адрес 11000000 10101000 00001100 00010011 пользователю представляется как 192.168.12.19.

Но преобразование двоичного кода в десятичный и обратное ему не всегда очевидно. В приведенном примере октет 10101000 преобразуется в 168, но ментально это не вычисляется. Для чисел большей разрядности это утверждение еще более справедливо.

Таким образом, отображение двоичной системы удачнее производить в шестнадцатеричную систему, используя тот факт, что $16 = 2^4$.

Шестнадцатеричный символ	Соответствующее двоичное число	Соответствующее десятичное число
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Продолжая пример IP-адресации вспомним, что IP-адрес 6-ой версии протокола TCP/IP представляет 128-ми битное число, которое отображается пользователю уже в шестнадцатеричной системе счисления.

ления. Например, 2A00:1370:811B:19D2:D213:8EC3:A456:8648. То есть 32 шестнадцатеричных символа, каждый из которых кодирует 4 символа двоичной системы. $32 \cdot 4 = 128$.

Очевидно, что для 6-ой версии TCP/IP представлять IP/адрес в десятичной системе было бы крайне неудобно.

Таким образом, возвращаясь к решению данной задачи, самым простым способом является прямой перевод заданного числа согласно указанной выше таблице из шестнадцатеричной системы в двоичную.

$$12AC0_{16} = 0001\ 0010\ 1010\ 1100\ 0000_2$$

Далее требуется продемонстрировать понятие значащего нуля: все нули, расположенные левее первого ненулевого символа, не меняют значения числа и не являются значащими.

$$0001\ 0010\ 1010\ 1100\ 0000_2 = 1\ 0010\ 1010\ 1100\ 0000_2.$$

Количество нулей справа подсчитывается вручную: данное число содержит 11 значащих нулей.

Вторая часть задания предполагает алгоритмизацию подсчета значащих нулей введенного в программу двоичного кода.

При реализации алгоритма важно понимать, что вводимая информация (то есть двоичный код числа) вводится как строка. То есть переменная, содержащая исходные данные будет иметь тип string. Причин тому несколько:

- В противном случае (при вводе исходных данных как числа) задача частично теряет свой смысл: все незначащие нули отбрасываются автоматически.
- Часть языков программирования высокого уровня (например, C#) воспринимает вводимый с клавиатуры набор символов именно как строку.
- Вводимое число автоматически воспринимается как число десятичной системы счисления, и дальнейшие математические действия будут алгоритмически не вполне корректными по отношению к введенной информации.

Таким образом, в данной задаче требуется демонстрация понимания типов данных и работы со строковыми данными.

Теоретические аспекты работы с тестом следующие.

Для ввода, хранения и обработки текстовой информации предусмотрен свой способ представления информации в двоичном коде: любой символ, вводимый с клавиатуры, представляется соответствующим этому символу двоичным кодом в соответствии с таблицей кодирования.

Первой и наиболее известной таблицей кодирования была ASCII. Исходная таблица ASCII позволяла кодировать всего 128 символов, среди которых были буквы английского алфавита, цифры и все прочие клавиатурные символы. Преимуществом ASCII является малый размер кода для хранения символа. Недостатком – малое количество представляемых символов, не позволяющее охватывать все алфавиты всех стран.

Наиболее распространенная в настоящее время таблица кодировок UNICODE. UNICODE имеет несколько стандартов, согласно которым для хранения одного символа отводится 2 или 4 байта.

Приведем примеры кодирования некоторых символов в ASCII и UNICODE.

Символ	В кодировке ASCII	В кодировке UNICODE двоичный код	В кодировке UNICODE шестнадцатеричный код
A	0100 0001	0000 0000 0100 0001	0041
B	0100 0010	0000 0000 0100 0010	0042
C	0100 0011	0000 0000 0100 0011	0043
a	0110 0001	0000 0000 0110 0001	0061
b	0110 0010	0000 0000 0110 0010	0062
c	0110 0011	0000 0000 0110 0011	0063
А	1100 0000	0000 0100 0001 0000	0410
Б	1100 0001	0000 0100 0001 0001	0411
В	1100 0010	0000 0100 0001 0010	0412
а	1110 0000	0000 0100 0011 0000	0430
б	1110 0001	0000 0100 0011 0001	0431
в	1110 0010	0000 0100 0011 0011	0432
4	0011 0100	0000 0000 0011 0100	0034
5	0011 0101	0000 0000 0011 0101	0035
6	0011 0110	0000 0000 0011 0110	0036

Из детального рассмотрения фрагментов таблиц можно сделать следующие выводы:

- Имеет место частичное совпадение в кодах ASCII и UNICODE (в младших разрядах). В приведенной таблице это хорошо видно для символов латинского алфавита и для чисел. Но это не обеспечивает совместимости кодировок: для русского алфавита кодировки имеют различия.
 - Символы представления заглавных и прописных букв различаются.
 - В обеих кодировках двоичный код соответствует алфавитному порядку буквы.
 - Символы, отображающие цифры, также представлены в таблицах кодировки. И хотя имеется некоторое соответствие между двоичной записью числа и его кодом в таблицах ASCII и UNICODE, важно пони-

мать, что представление числа как числа и числа как символа различаются. Например, число «6», представленное в памяти вычислительной системы как число, будет иметь вид «0000 0110» (если предположить, что на число отводится 1 байт). Символ «6» будет записан в кодировке ASCII как «0011 0110» и в кодировке UNICODE «0000 0000 0011 0110».

Продолжая тему представления чисел как строкой информации, надо понимать, что число «66», записанное как число, будет иметь вид «0100 0010» (если предположить, что на число отводится 1 байт). Записанное как текст, число «66» будет представлено как набор (массив) символов «6» и «6». В кодировке ASCII будет записано как «0011 0110 0011 0110»; в кодировке UNICODE - «0000 0000 0011 0110 0000 0000 0011 0110».

В языках программирования для работы с текстом вводятся специальные переменные типов `char` и `string`. Переменная типа `char` содержит всегда только один символ. Переменная типа `string` содержит последовательность символов.

С каждым типом переменных предусмотрены свои допустимые операции.

Если для переменных числового типа допустимы операции сложения, вычитания, умножения, деления и проч. (набор математических операций различается в различных языках программирования), то для строковых переменных возможна операция сложения, определения длины строки, поиска подстроки в строке.

При этом операция сложения будет выполняться иначе, нежели для числовых данных. Например, для числовых переменных «123» и «654» операция сложения даст «777». Для строковых переменных «123» и «654» операция сложения даст «123654».

Операция определения длины строки для «123654» даст «6», операция поиска подстроки «6» в строке «123654» вернет позицию «4» (если нумерация символов в строке начинается с 1).

Возвращаясь непосредственно к решению задачи, исходим из того, что на вход программы поступает строка 0001 0010 1010 1100 0000, представляющая двоичный код шестнадцатеричного числа. (Естественно, программу можно алгоритмизировать полностью, введя исходно шестнадцатеричное число, и, преобразовав его в дальнейшем в двоичное представление. Но этот вариант значительно усложнит задачу.)

Основные шаги по решению задачи следующие:

1. Ввод данных
2. Анализ длины строки: если строка нулевая, то дальнейшее выполнение программы не требуется, ответом будет «0».

3. Определение позиции первой единицы (все нули до нее не подсчитываются)

4. Если единица существует и не находится на последней позиции строки, то начинать подсчет значащих нулей. В противном случае, значащих нулей не существует, ответ «0».

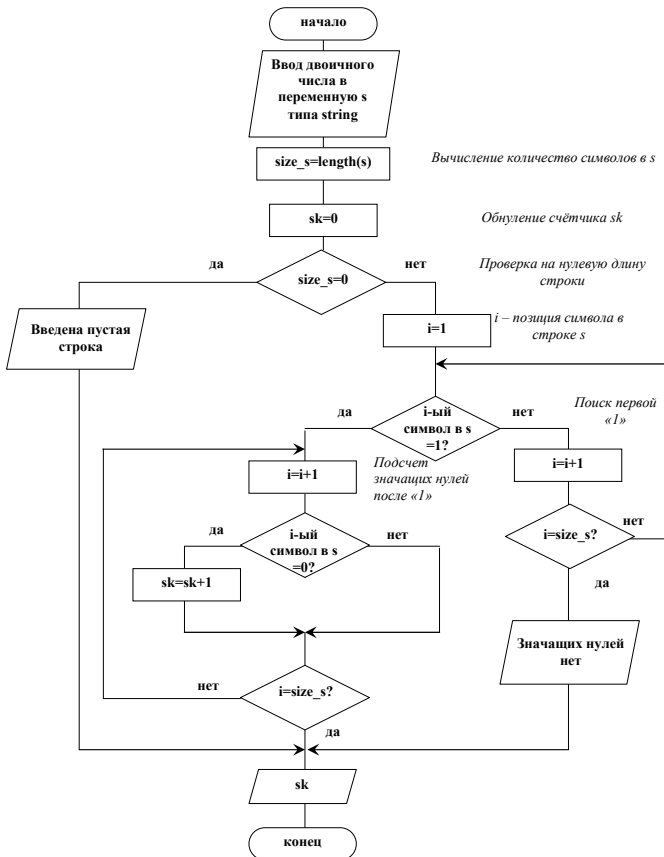
5. Для подсчета значащих нулей вводится и обнуляется переменная-счетчик.

6. Перебирая строку с найденной позиции первой единицы до конца строки, подсчитывается количество нулей

7. Вывод результата

Ниже приведены решения задачи как в виде блок-схемы, реализующей алгоритм поставленной задачи, так и в виде программных кодов (языки программирования VBA, Pascal ABC, C#).

Блок-схема алгоритма задачи



Решение задачи на VBA.

```
Sub task()
Dim s As String
s = InputBox("Введите число")
size_s = Len(s)
sk = 0
If size_s <> 0 Then
For i = 1 To size_s
If Mid(s, i, 1) = "1" Then Exit For
Next i

If i < size_s Then
For j = i To size_s
If Mid(s, j, 1) = "0" Then sk = sk + 1
Next j
End If
End If
MsgBox (sk)
End Sub
```

'определение длины строки
'обнулении счетчика
'если введенная строка нулевая, то окончание работы
'поиск от первой до последней позиции символа "1"
'при нахождении символа "1" – досрочный выход из цикла

'если "1" не нашлась или оказалась последней, то окончание работы
'поиск от позиции первой единицы до конца строки
'если есть "0", увеличить счетчик

Решение на Pascal ABC

```
label 1;
var s:string;
size_s,sk,i,j:integer;
begin
readln(s);
size_s:=Length(s);
sk:=0;
if (size_s<>0) then
begin
for i:=1 to size_s do
if Copy(s,i,1)='1' then goto 1;

1:if i<size_s then
for j:=i to size_s do
if Copy(s,j,1)='0' then sk:=sk+1;
end;
writeln(sk);
end.
```

По поводу алгоритма, реализованного на Паскале, стоит дать некоторое уточнение: в алгоритме используется оператор goto, который не рекомендуется к активному использованию в программировании. Этому можно привести следующие объяснения:

- Оператор goto позволяет совершать переходы из любой точки программы в любую другую, что при некорректном его использовании может привести к значительным сложностям отладки программы
- Современные средства программирования позволяют писать программы без использования goto.

Так, в VBA, для досрочного выхода из цикла For next используется оператор ExitFor. В Pascal ABC и в C# для аналогичных целей используется break.

Таким образом, последний алгоритм можно переписать как

```
var s:string;
size_s,sk,i,j:integer;
```

```

begin
readln(s);
size_s:=Length(s);
sk:=0;
if (size_s<>0) then
begin
  for i:=1 to size_s do
    if Copy(s,i,1)='1' then break;

    if i<size_s then
      for j:=i to size_s do
        if Copy(s,j,1)='0' then sk:=sk+1;
      end;
    writeln(sk);
  end.
end.

```

Оба алгоритма реализуют абсолютно одинаковую логику. Применение оператора goto не усложняет алгоритм и в данном случае его применение не считается ошибкой. Оба решения засчитываются как верные. Но стоит помнить, что избыточное и необоснованное применение оператора goto может быть воспринято при проверке задачи как демонстрация незнания основных операторов языка.

Другим способом решить ту же задачу является использование операторов языка, автоматически реализующих поставленную задачу: ниже приведен алгоритм, в котором позиция первой единицы определяется оператором Pos('1',s) (найти в строке s позицию строки '1'). Естественно, демонстрация хорошего знания текстовых (или иных операторов) учитывается при проверке. Но в подобных ситуациях рекомендуется либо детальное знание операторов, либо использования более простого способа решения задачи.

```

var s:string;
size_s,sk,i,j:integer;
begin
readln(s);
size_s:=Length(s);
sk:=0;
if (size_s<>0) then
begin
  i:=Pos('1',s);

  if i<size_s then
    for j:=i to size_s do
      if Copy(s,j,1)='0' then sk:=sk+1;
    end;
  writeln(sk);
end.
end.

```

<

Окно вывода

00000
6

Как видно из приведенного скриншота, при введенной строке пять нулей «00000», программа в качестве ответа выводит число 6, которое не является верным. Причиной тому является специфика оператора `Pos('1', s)`, который при отсутствии искомой строки (а при указанных введенных данных единица отсутствует) в качестве результата работы дает число «0». Что приводит к дальнейшему некорректному выполнению алгоритма.

В данной ситуации требуется ввести дополнительную проверку: `if`

```

(i<size_s) and (i>0)
var s:string;
size_s,sk,i,j:integer;
begin
  readln(s);
  size_s:=Length(s);
  sk:=0;
  if (size_s<>0) then
  begin
    i:=Pos('1',s);

    if (i<size_s) and (i>0) then
      for j:=i to size_s do
        if Copy(s,j,1)='0' then sk:=sk+1;
      end;
  writeln(sk);
  end.

```

Ниже приводится решение задачи на C# двумя способами: первый – описанный в блок-схеме, с поиском первой единицы перебором строки. Второй – с использованием `i=s.IndexOf("1");`, работающим аналогично `Pos('1', s)` в Pascal ABC.

```

static void Main(string[] args)
{
  string s;
  int i,j,size_s,sk;
  s = Console.ReadLine();
  size_s = s.Length;
  sk = 0;
  if (size_s != 0)
  {
    for (i = 0; i < size_s; i++)
      if (s.Substring(i,1)=="1") break;

    for (j = i; j < size_s; j++)
      if (s.Substring(j, 1) == "0") sk++;
  }
  Console.WriteLine(sk);
}

```

```

    }
static void Main(string[] args)
{
    string s;
    int i,j,size_s,sk;
    s = Console.ReadLine();
    size_s = s.Length;
    sk = 0;
    if (size_s != 0)
    {
        i=s.IndexOf("1");

        for (j = i; j < size_s; j++)
            if (s.Substring(j, 1) == "0") sk++;

    }
    Console.WriteLine(sk);
}

```

Таким образом, на детальном разборе данной задачи продемонстрировано, что решения задач могут быть оформлены различными способами, при реализации программного кода может быть выбран любой существующий язык программирования, алгоритмы могут различаться. Но обязательно

- должны учитываться система счисления вводимого в программу числа
- и тип переменных, содержащих введенные значения
- должны отсутствовать математические и логические ошибки
- рекомендуется демонстрация четкого алгоритмического мышления.

Типичные ошибки, допущенные при реализации задачи

1. Неверное определение типов данных и вытекающие ошибки обработки данных
2. Пропуск пункта 3 алгоритма «Определение позиции первой единицы» и подсчет всех нулей в записи числа
3. Пропуск пункта 2 алгоритма «Анализ длины строки: если строка нулевая, то дальнейшее выполнение программы не требуется» не является существенной ошибкой, но наличие в алгоритме данного пункта означает, что автор программы умеет анализировать вводимые значения и оптимизировать алгоритм. Аналогичная ситуация с пунктом 4 алгоритма «Если единица существует и не находится на последней позиции строки.»

Задача 12 (Всероссийская олимпиада школьников по информатике, 2014–15 уч. год Первый (школьный) этап, г. Москва Задания для 9–11 классов

«Автомобильные номера». В Российской Федерации на разных видах транспортных средств устанавливаются разные по формату регистрационные знаки («автомобильные номера»). Вот пример нескольких возможных форматов регистрационных знаков.

№	Пример	Описание формата	Тип транспортного средства
1	Y019KM	Буква, три цифры, две буквы	Частные транспортные средства
2	AB179	Две буквы, три цифры	Общественный транспорт и такси
3	OH2645	Две буквы, четыре цифры	Прицепы
4	3384CT	Четыре цифры, две буквы	Мотоциклы

В этой задаче «буквой» может быть любая заглавная буква латинского алфавита. Напишите программу, которая по регистрационному знаку определяет его тип или определяет, что регистрационный знак некорректен. Программа получает на вход три строки текста, каждая строка содержит один образец регистрационного знака (возможно, некорректный). Каждый образец содержит от 1 до 10 символов, являющихся цифрами и заглавными латинскими буквами (других символов во входных данных быть не может).

Программа должна вывести для каждого образца число, соответствующее типу транспортного средства, как в приведенной таблице, то есть 1 – для частных транспортных средств, 2 – для общественного транспорта, 3 – для прицепов, 4 – для мотоциклов. Если номерной знак некорректен (не подходит ни к одному из указанных типов), то необходимо вывести число 0.

Каждое число необходимо выводить в отдельной строке. Пример входных и выходных данных

Ввод	Вывод
Y019KM	1
A9999	0
OH2645	3

Проверяется

Умение создавать и отлаживать коды программ на каком-либо языке программирования; навыки алгоритмизации, умение работать с символьными и с текстовыми типами данных.

Естественно, для выполнения данной программы требуется наличие аппаратных и программных средств. Независимо от условий проводимой олимпиады (проводится она в дисплейном классе или в

письменном виде) представляется полезным разобрать программную реализацию данной задачи.

Обсуждается

Символьные и строковые типы данных.

Применение функций.

Оптимальность алгоритма.

Теоретическое сопровождение темы. Решение

Алгоритм решения данной задачи не сложен: требуется ввести номер транспортного средства и проанализировать введенный набор символов на соответствие существующим форматам. Фактически задача сводится к посимвольному сравнению заданного номера. Для ее успешного решения надо понимать, что такое символьный тип данных.

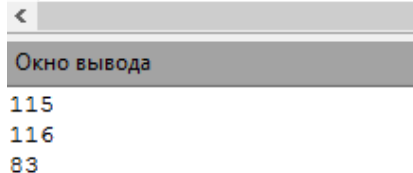
Переменная типа `string` представляет собой последовательность отдельных символов. Причем, переменная типа `string` может не содержать ни одного символа. В Pascal ABC и в C# строка – это массив символов. Для строк в любых языках программирования введены операции сложения, извлечения части подстроки (слева, справа, из любой позиции), определения позиции искомой подстроки в строке, определения длины строки.

Переменная типа `char` содержит только один символ. Преимущество символьных данных заключается в том, что для символьных переменных доступны операции получения соответствующего символу UNICODE и операция обратная ей – по UNICODE вернуть символ.

На приведенном рисунке видно, что UNICODE символа 's' равен 115, а UNICODE следующего за ним символа 't' равен 116. Таким образом, для символьных переменных есть операция сравнения. Сравнение производится по UNICODE. Получается, что 's' < 't'.

А 'S' < 's'.

```
begin
ch:='s';
writeln(ordUnicode(ch));
ch:='t';
writeln(ordUnicode(ch));
ch:='S';
writeln(ordUnicode(ch));
end.
```



Окно вывода

```
115
116
83
```

Именно это свойство символьных переменных (сравнение символов по UNICODE) и применяют для проверки соответствия какого-либо символа заданной группе символов:

- Если (согласно условию задачи) надо проверить, не является ли символ цифрой, то данный символ должен быть больше или равен '0' и одновременно меньше или равен '9'.
- Если символ проверяется на принадлежность к латинскому алфавиту (согласно условию задачи – только заглавные буквы), то он должен быть больше или равен 'A' и меньше или равен 'Z'.

Вывод: для проведения сравнения введенного номера с существующими шаблонами надо будет сравнивать каждый символ введенного номера. Но, согласно условию, вводится номер целиком (не по одному символу), то есть строка. Для разрешения данного вопроса в Pascal ABC и в C# надо вспомнить, что строка – это массив символов. На рисунке ниже приведена программа работы с строковой переменной s.

```
var s:string;
begin
s:='abcdefg';           //в переменную s записывается строка 'abcdefg'
writeln(s[2]);         //на экран выводится второй элемент строки
writeln(s[5]);         //на экран выводится пятый элемент строки
writeln(s[s.Length]); //на экран выводится последний элемент строки
end.
```

Окно вывода

```
b
e
g
```

Из приведенного рисунка видно, что s, объявленная как строковая переменная позволяет обращаться к себе, как к массиву:

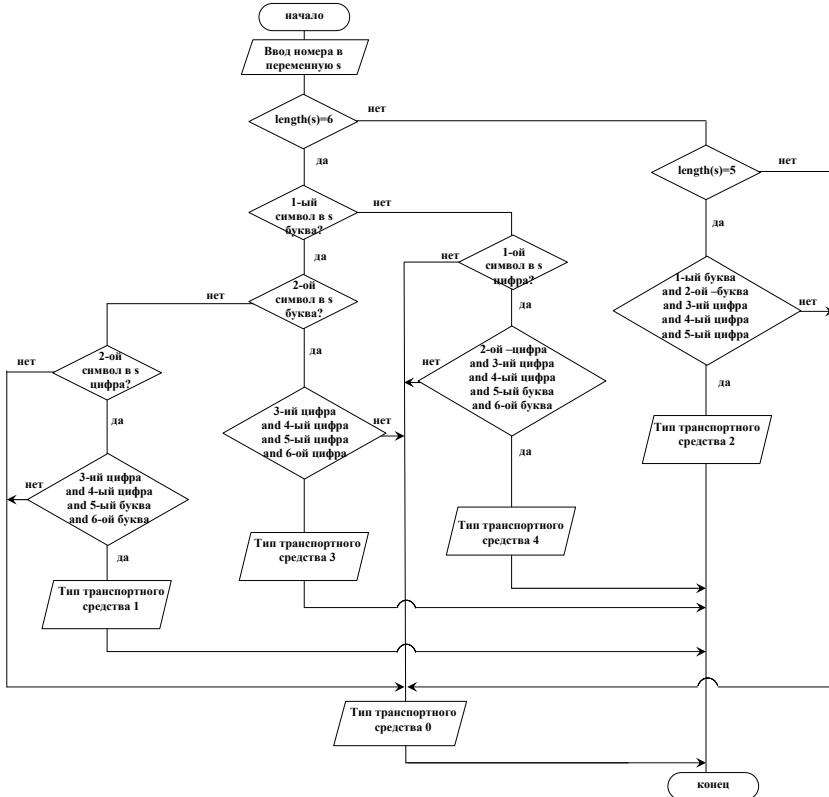
- s[2] – обращение ко второму элементу массива s, то есть символ 'b';
- s[5] – обращение к пятому элементу массива s, то есть символ 'e'.
- s[s.Length] – обращение к элементу массива s, с номером s.Length. s.Length – это определение длины строки s. Следовательно, s[s.Length] – символ 'g'.

В VBA каждый символ строки извлекается при помощи команды Mid(s, 1), которая позволяет извлечь подстроку из строки s длиной (в данном случае) 1 символ.

Сравнивая символы введенного номера важно помнить, что введенный номер – строка, которая может содержать от одного до десяти символов. А существующие форматы описывают номера в 5 и 6

символов. Таким образом, представляется целесообразным предварительно проверить длину введенной строки, не проводя дальнейших проверок, если длина не соответствует 5 или 6.

Таким образом, определив способы сравнения номера с шаблонами, можно представить алгоритм решения задачи в виде блок-схемы.

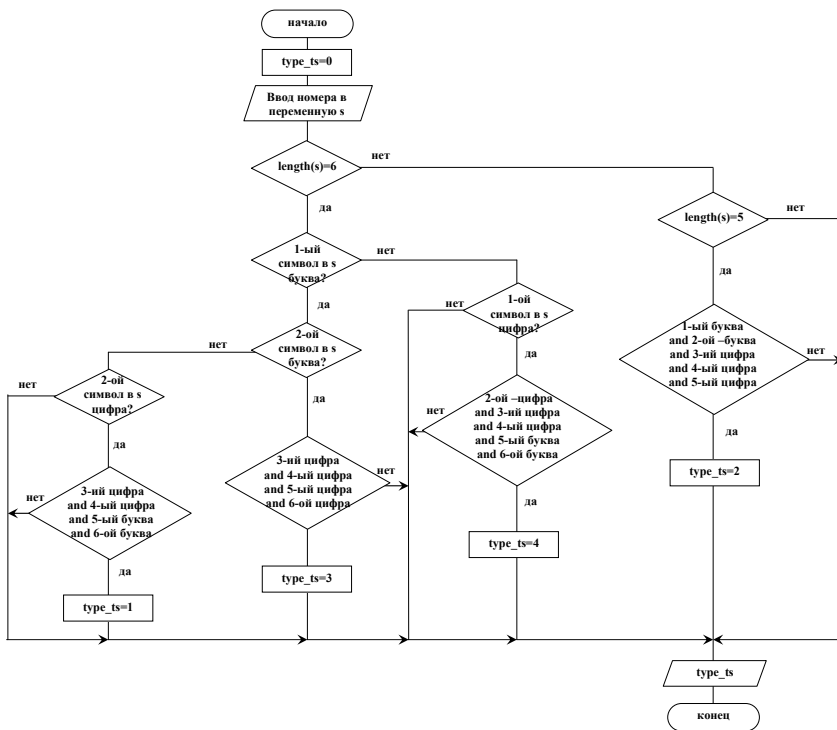


Анализ блок-схемы показывает, что

- Практическая реализация задачи будет строиться на слишком большом количестве последовательно примененных операторов условного перехода (обратите внимание, что каждая проверка каждого символа – это также двойное условие).
- Логика приведенного алгоритма затруднена и грозит применением непопулярного оператора goto для реализации выхода из задачи по несовпадению ни с одним из указанных форматов.

Для исправления указанных недостатков:

- Рекомендуется ввести функции булевского типа для проверки, является ли символ цифрой (`test_num`) или заглавной буквой (`test_char`) латинского алфавита. Обе функции принимают символ в качестве параметра; анализируют его на принадлежность требуемому диапазону и возвращают `true`, если принадлежит; `false`, если нет.
- Ввести переменную `type_ts` (тип транспортного средства) и определить ее значение равным нулю до начала работы алгоритма. Только в случае совпадения с одним из указанных форматов значение переменной будет переопределено.



Далее приводятся коды программ, реализующих данный алгоритм. Естественно, можно привести и иные, более удачные варианты решения данной задачи. При выборе алгоритма авторы руководствовались:

- соображениями наибольшей очевидности алгоритма и программного кода;

- возможностью однотипной реализации алгоритма на языках PascalABC, C#, VBA.

Некоторым отклонением от исходной задачи является организация ввода номеров до тех пор, пока не будет введена пустая строка (нулевой длины), что позволяет проверить все ветви алгоритма однократным запуском программы. Для этого применен оператор цикла по условию `while` с условием продолжения цикла ненулевой длиной введенной строки.

Для приведения приведенных кодов в полное соответствие с исходной задачей достаточно заменить цикл по условию на цикл по счетчику с тремя повторениями.

При анализе кодов программ обратите внимание, что имеются частичные несовпадения кодов, что связано со спецификой каждого языка:

- В C# нумерация любого массива начинается с нуля. Строка – это массив символов, следовательно, первым символом является `s[0]`.

- В Pascal ABC строка – это также массив символов. Но нумерация символов в массиве строки начинается с 1, следовательно, первым символом является `s[1]`. Ввод данных в программе заканчивается вводом символа '0'.

- В VBA не используется тип `char`, но есть возможность производить сравнение строк согласно UNICODE их символов: «большей» является та строка, в которой UNICODE соответствующего символа выше. Кроме того строка не представляется массивом символов, поэтому в процедуру передается не символ, а подстрока, выделенная из проверяемой строки.

C#:

```
class Program
{
    static bool test_char(char tc)
    {
        if ((tc >= 'A') && (tc <= 'Z')) return true;
        else return false;
    }
    static bool test_num(char tc)
    {
        if ((tc >= '0') && (tc <= '9')) return true;
        else return false;
    }
    static void Main(string[] args)
    {
        string s = Console.ReadLine();
        while (s.Length != 0)
```

```

    {
        int type_ts = 0;
        if (s.Length == 6)
        {
            if (test_char(s[0]))
            {
                if (test_char(s[1]))
                    if (test_num(s[2]) && test_num(s[3])
&& test_num(s[4]) && test_num(s[5])) type_ts = 3;
                if (test_num(s[1]))
                    if (test_num(s[2]) && test_num(s[3])
&& test_char(s[4]) && test_char(s[5])) type_ts = 1;
            }
            else
                if (test_num(s[0]))
                    if (test_num(s[1]) && test_num(s[2]) &&
test_num(s[3]) && test_char(s[4]) && test_char(s[5])) type_ts = 4;
            }
            else
            {
                if (s.Length == 5)
                {
                    if (test_char(s[0]) && test_char(s[1]) &&
test_num(s[2]) && test_num(s[3]) && test_num(s[4])) type_ts = 2;
                }
            }
            Console.WriteLine("тип транспортного средства
{0}", type_ts);
            s = Console.ReadLine();
        }
    }
}

```

Pascal ABC:

```

function test_char(tc:char):boolean;
begin
if (tc>='A') and (tc<='Z') then test_char:=true else
test_char:=false;
end;
function test_num(tc:char):boolean;
begin
if (tc>='0') and (tc<='9') then test_num:=true else
test_num:=false;
end;
var s:string;
type_ts:integer;
begin
readln(s);

```

```

while s<>'0' do
begin
type_ts:=0;
if (s.Length=6) then
begin
if test_char(s[1]) then
begin
if test_char(s[2]) then
if
(test_num(s[3]))and(test_num(s[4]))and(test_num(s[5]))and(test
st_num(s[6])) then type_ts:=3;
if test_num(s[2]) then
if
(test_num(s[3]))and(test_num(s[4]))and(test_char(s[5]))and(t
est_char(s[6])) then type_ts:=1;
end
else
if test_num(s[1]) then
if
(test_num(s[2]))and(test_num(s[3]))and(test_num(s[4]))and(te
st_char(s[5]))and(test_char(s[6]))then type_ts:=4;
end
else
if (s.Length=5) then
if
(test_char(s[1]))and(test_char(s[2]))and(test_num(s[3]))and(
test_num(s[4]))and(test_num(s[5])) then type_ts:=2;
writeln(type_ts);
readln(s);
end;
end.

```

VBA:

```

Function test_char(tc As String)
If tc >= "A" And tc <= "Z" Then
test_char = True
Else
test_char = False
End If
End Function
Function test_num(tc As String)
If tc >= "0" And tc <= "9" Then
test_num = True
Else
test_num = False
End If
End Function
Sub task()
Dim s As String
Dim type_ts As Integer

```

```

s = InputBox("Введите номер")
While Len(s) <> 0
type_ts = 0
  If Len(s) = 6 Then
    If test_char(Mid(s, 1)) Then
      If test_char(Mid(s, 2)) Then
        If test_num(Mid(s, 3)) And _
          test_num(Mid(s, 4)) And _
          test_num(Mid(s, 5)) And _
          test_num(Mid(s, 6)) Then type_ts = 3
        Else
          If test_num(Mid(s, 2)) Then
            If test_num(Mid(s, 3)) And _
              test_num(Mid(s, 4)) And _
              test_char(Mid(s, 5)) And _
              test_char(Mid(s, 6)) Then type_ts = 1
            End If
          End If
        End If
      If test_num(Mid(s, 1)) Then
        If test_num(Mid(s, 2)) And _
          test_num(Mid(s, 3)) And _
          test_num(Mid(s, 4)) And _
          test_char(Mid(s, 5)) And _
          test_char(Mid(s, 6)) Then type_ts = 4
        End If
      Else
        If Len(s) = 5 Then
          If test_char(Mid(s, 1)) And _
            test_char(Mid(s, 2)) And _
            test_num(Mid(s, 3)) And _
            test_num(Mid(s, 4)) And _
            test_num(Mid(s, 5)) Then type_ts = 2
          End If
        End If
      MsgBox (type_ts)
      s = InputBox("Введите номер")
    Wend
  End Sub

```

Типичные ошибки, допущенные при реализации задачи

1. Отсутствие функций проверки символа на число или на заглавную латинскую букву позволяет реализовать данный алгоритм, но значительно его затрудняет. В подобных случаях рекомендуется применение функций проверки.

2. Ошибки реализации логики алгоритма, как правило – в применении оператора if.

3. Избыточные проверки уже проверенных символов. Например, после успешной проверки первого символа на число остается проверить оставшиеся 3 символа на число и 2 символа на букву латинского алфавита. Указанный недочет не является фатальной ошибкой, но увеличивает время выполнения программы и ... не лучшим образом свидетельствует о логике автора программы

4. Применение goto для выхода в случае неудачного прохождения проверки. В данной задаче наиболее удобным вариантом является исходное определение значения типа транспортного средства равным нулю. В случае удачного прохождения проверки значение типа переопределяется в соответствии со стандартом.

Задача 13 (Олимпиада по информатике «МИСиС зажигает звезды», 2016 год, 11 класс)

Дошкольник научился писать и написал свое первое предложение. Но, подчеркивая важность некоторых слов, некоторые слова в предложении написал с большой буквы, а некоторые – с маленькой. Имен собственных в предложении нет. Напишите программу, которая исправит текст дошкольника: первая буква в предложении должна быть большая, все остальные слова должны начинаться с маленькой буквы.

Символ	В кодировке ASCII	В кодировке UNICODE двоичный код	В кодировке UNICODE шестнадцатеричный код
А	1100 0000	0000 0100 0001 0000	0410
Б	1100 0001	0000 0100 0001 0001	0411
В	1100 0010	0000 0100 0001 0010	0412
а	1110 0000	0000 0100 0011 0000	0430
б	1110 0001	0000 0100 0011 0001	0431
в	1110 0010	0000 0100 0011 0011	0432

Проверяется

Навыки алгоритмизации и программирования, знание особенностей типов данных и умение с ними работать; в частности – с символьными и строковыми данными.

Обсуждается

Особенности работы с символьными данными, создание и реализация алгоритма, программирование алгоритма, применение функций и процедур, передача параметра по ссылке и передача параметра по значению. Тестирование программ.

Теоретическое сопровождение темы. Решение

Исходя из условия задачи, очевидно, что тип обрабатываемых данных – это строка. Программа должна принимать строку как вход-

ные данные и выводить строку в качестве выходных данных. Строка-результат формируется коррекцией исходной строки:

- Надо рассмотреть в исходной строке каждый символ;
- Первый символ в строке должен быть заглавной буквой;
- Далее надо проверять прочие символы. По условию некоторые слова в предложении дошкольник может написать с большой буквы (по ошибке). Чтобы выделить следующее слово, можно проверять каждый символ строки: если перед ним стоит символ пробела, то рассматриваемый символ является первым символом слова.

- Если это слово является началом предложения, значит, рассматриваемый символ должен быть заглавной буквой.
- Если это слово стоит не первым в предложении, значит, рассматриваемый символ должен быть прописной буквой.

Распознать начало предложения возможно по символу, предшествующему пробелу:

- Если перед пробелом стоит точка, вопросительный знак или восклицательный знак, то рассматриваемый символ – начало предложения.
- Если перед пробелом стоит любой иной символ – то рассматриваемый символ – продолжение предложения.

Наметив общие положения алгоритма, разберем техническую возможность анализа строковых данных. Большая часть языков программирования поддерживает концепцию, в которой строка является массивом символов. Например, переменная типа `string s = "блок"` является массивом размером 4 символьных элементов. К каждому символу можно обращаться, как к элементу массива:

- `s[1]='б'`
- `s[2]='л'`
- `s[3]='о'`
- `s[4]='к'`

В приведенном примере нумерация элементов начинается с 1; так бывает не всегда. В `C#` нумерация массива начинается с 0.

Но, обращение к отдельным элементам массива возможно только на чтение. То есть каждый символ можно выделить, сравнить, напечатать, скопировать в переменную символьного типа, но нельзя поменять значение символа в массиве, тем самым отредактировав исходную строку.

В приведенном примере возможна команда `c=s[4]` (подразумевается, что `c` – переменная символьного типа `char`; то есть копируется символ `'к'` в переменную `c`). Но невозможна команда `s[1]=c` (то есть изменить в строке «блок» один символ и получить «клок»).

Из всего изложенного можно предложить следующую техническую реализацию задачи – для получения выходного результата требуется ввести в программе две переменных типа string:

- переменная s, в которую вводится исходная строка,
- переменная result, которая исходно равна "" (пустой строке), а в процессе анализа исходной строки в нее добавляются символы исходной строки: result=result + s[i]; но, перед добавлением символа требуется произвести анализ предшествующих символов (s[i-1] и s[i-2]), чтобы определить, не требуется ли коррекция текущего символа.

Обсудим вопрос коррекции символа. Согласно условию задачи дошкольник не понял, когда слово пишется с заглавной (большой) буквы, а когда – с прописной (маленькой). Следовательно, коррекция подразумевает определение типа буквы и изменение типа буквы. Обе эти задачи решаются исходя из свойств символьного типа данных.

Каждый символ имеет свой код; в условии задачи приведены коды некоторых символов. Из приведенной таблицы можно увидеть, что код любой прописной буквы имеет значение на 32 больше, чем код заглавной буквы. Следовательно, для того, что изменить символ прописной буквы на соответствующий ему символ заглавной буквы, надо взять символ, код которого формируется как код исправляемого символа минус 32. И, напротив, для изменения заглавной буквы на прописную, надо исправляемый символ заменить на символ с кодом на 32 большим.

Операция получения кода символа и обратная ей – получения символа по его коду – существует в любом языке программирования. Рассмотрим в качестве примера преобразование символа 'a' в символ 'A', предполагая, что некая переменная символьного типа ch содержит 'a' (ch='a'):

Действие	C#	PascalABC	VBA	Результат
Получить числовой код символа	Convert.ToInt16(ch)	Ord(ch)	Asc(ch)	224
Уменьшить значение кода на 32	Convert.ToInt16(ch)-32	Ord(ch)-32	Asc(ch)-32	192
Получить символ по коду	Convert.ToChar (Convert.ToInt16(ch)-32)	Chr(Ord(ch)-32)	Chr(Asc(ch)-32)	A

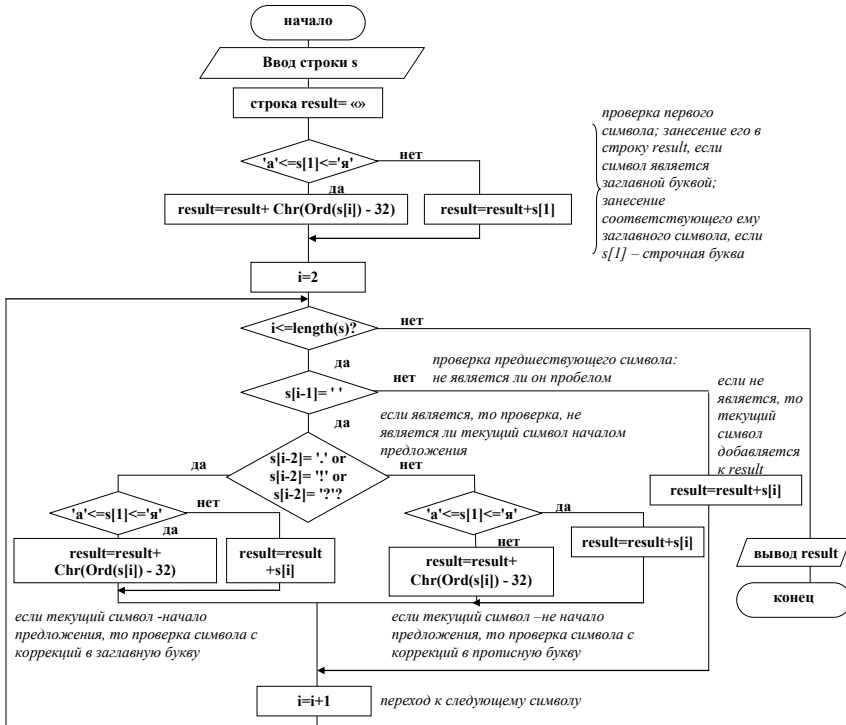
Кроме того, для символов предусмотрена возможность сравнения «>», «<» и «=»): сравнение символов производится по соответствующему им коду. Из приведенной в условии таблицы видно, что коды символам назначены согласно алфавитному порядку. Следовательно, 'a' < 'б'.

Возможность сравнения символов по коду позволяет определить тип буквы: заглавная или прописная:

- если символ $\geq 'a'$ и, одновременно, $\leq 'я'$, следовательно, этот символ – прописная буква русского алфавита;
- если символ $\geq 'A'$ и, одновременно, $\leq 'Я'$, следовательно, этот символ – заглавная буква русского алфавита.

Первая написанная буква должна быть заглавной.

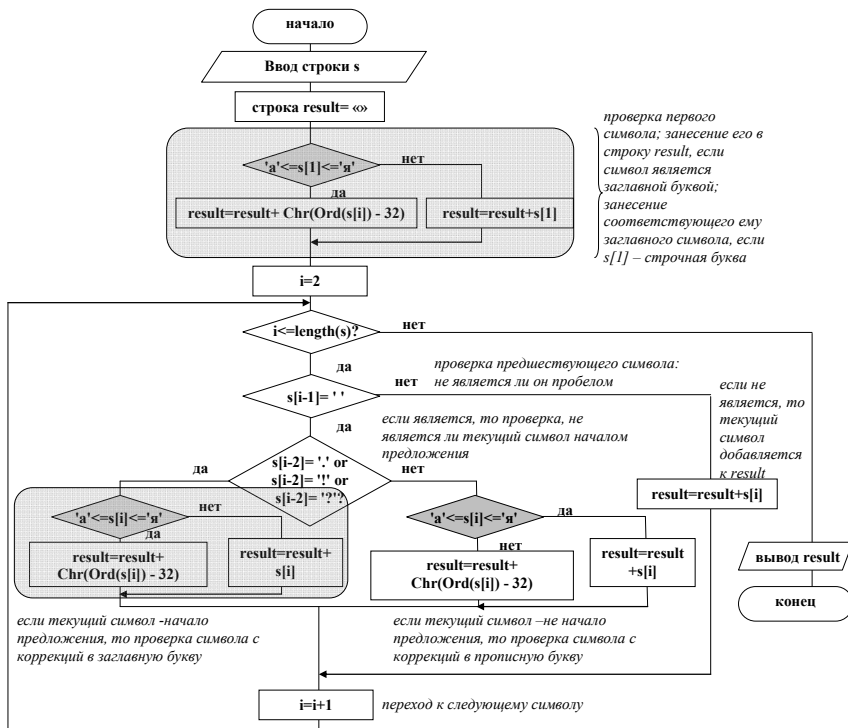
В целом алгоритм работы программы можно представить следующим образом:



Из приведенной блок-схемы можно сделать вывод о крайне запутанной логике программы, о большом количестве операторов условного перехода if, которые потребуется ввести в программу.

Кроме того, некоторые фрагменты блок-схемы повторяются. На блок-схеме ниже дублирующиеся блоки отображены одинаковым цветом.

В таком и подобных случаях рекомендуется использовать подпрограммы для облегчения написания программы и улучшения ее восприятия.



Оформим светло-серый блок как процедуру `addressult`, которая будет получать от основной программы формируемую строку `ges`, и добавляемый символ `s`. Если символ является прописной буквой, то процедура должна преобразовать символ в соответствующий ему заглавный. На выходе процедуры должна быть та же строка `ges`, но уже удлиненная на один символ.

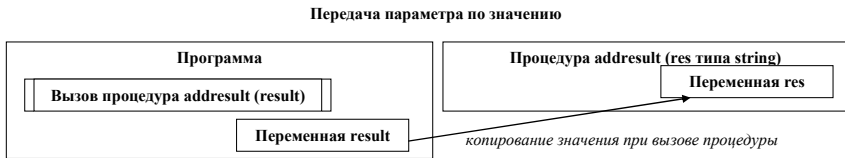
Для реализации подобной процедуры необходимо передавать строку `ges` процедуре не по значению, а по ссылке.

Как известно, переменная – это поименованная область памяти.

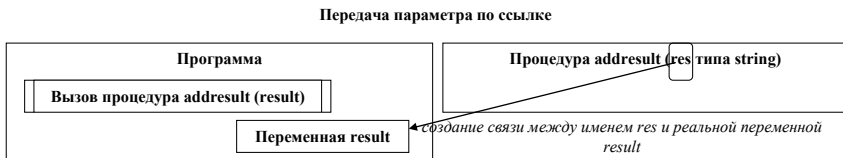
Существует два способа передачи параметра подпрограмме: по значению и по ссылке.

При передаче по значению в программе существует переменная `result` (то есть выделяется отдельная память под переменную `result`) и при вызове процедуры создается отдельная переменная `ges` (то есть выделяется отдельная память под переменную `ges`). При вызове процедуры `addressult` значение переменной `result` копируется в переменную `ges` процедуры. По окончании работы процедуры изменится

значение переменной `res`, но не изменится значение переменной `result`: обратного копирования значения `res` в `result` по окончании работы процедуры `addressresult` не предусмотрено.



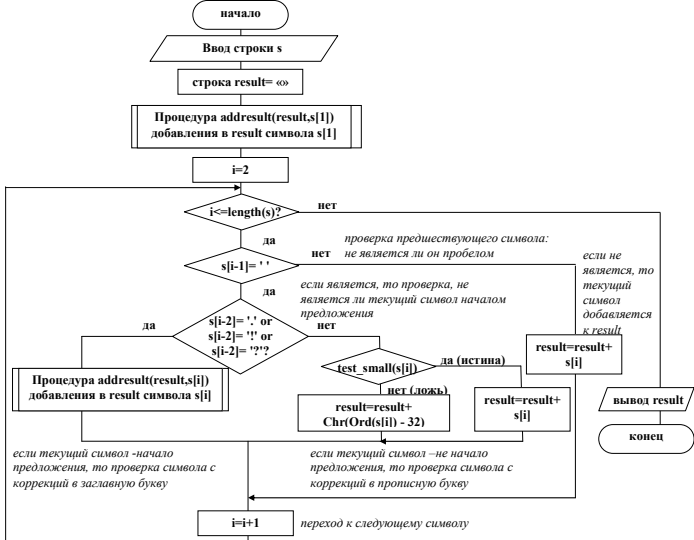
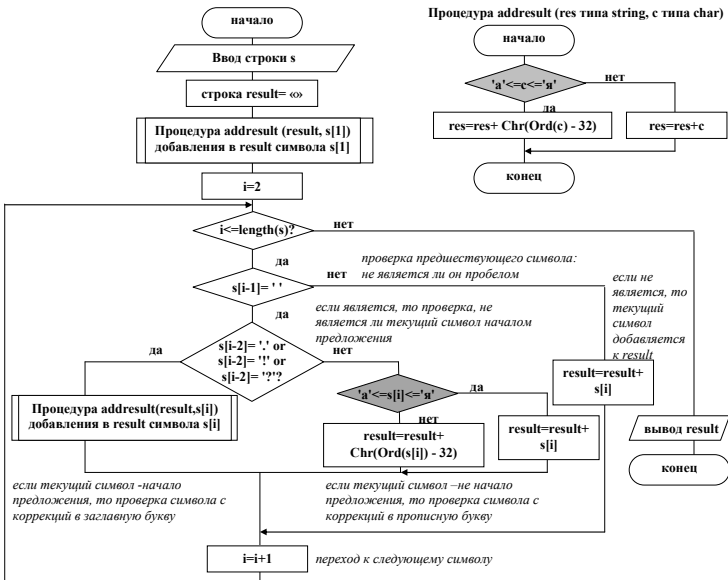
При передаче параметра по ссылке по-прежнему выделяется память под переменную `result` в основной программе. Но, при вызове процедуры `addressresult` не производится копирование содержимого строки `result` в переменную `res` процедуры. Вместо этого переменная `res` начинает указывать на область памяти, выделенную под переменную `result`. Таким образом, при изменении значения переменной, условно названной `res`, меняется значение реальной переменной `result`.



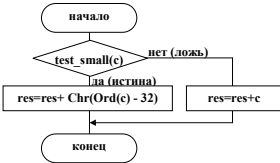
Передавая параметр `result` процедуре `addressresult` по ссылке, обеспечиваем изменение значение переменной `result` по окончании работы процедуры.

Оформим темно-серый блок как функцию `test_small`, которая будет получать от вызываемой программы проверяемый символ, проверять, является ли символ прописной буквой, и возвращать результат как булево значение: то есть ложь (если буква заглавная) или истина (если буква прописная).

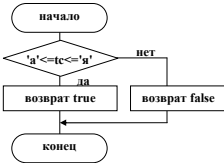
В этом случае блок-схема будет иметь следующий вид:



Процедура address (res типа string, с типа char)



Функция test_small (tc типа char)



Программная реализация представлена ниже.

Реализация в C# содержит класс Program, в котором, помимо основного исполняемого метода Main присутствуют метод test_small, возвращающий значение типа bool, и метод addressult, которому передается значение по ссылке с помощью ref:

```
class Program
{
    static bool test_small(char tc)
    {
        if ((tc >= 'а') && (tc <= 'я')) return true;
        else return false;
    }

    static void addressult(ref string res, char c)
    {
        if (test_small(c))
            res = res + Convert.ToChar(Convert.ToInt16(c)-32);
        else
            res += res + c;
    }

    static void Main(string[] args)
    {
        Console.WriteLine("Введите строку");
        string s=Console.ReadLine();
        string result = "";
        addressult(ref result, s[0]);
        int i;
        for (i = 1; i < s.Length; i++)
        {
            if (s[i - 1] == ' ')
            {
                if ((s[i - 2] == '.' || (s[i - 2] == '!') || (s[i - 2] == '?'))
                    addressult(ref result, s[i]);
                else
                {
                    if (test_small(s[i]))
                        result = result + s[i];
                    else
                        result = result + onvert.ToChar(Convert.ToInt16(s[i])+32);
                }
            }
            else
            {
                result = result + s[i];
            }
        }
        Console.WriteLine(result);
    }
}
```

Реализация в PascalABC содержит функцию test_small, возвращающую значение типа boolean, процедуру addressult, которому передается значение по ссылке за счет var в описании типа передаваемых

параметров. Переменная result в основной программе переименована в results, так как в PascalABC «result» не может быть использовано как имя переменной.

```
function test_small(tc:char):boolean;
begin
if (tc>='a') and (tc<='я') then test_small:=true else test_small:=false;
end;
```

```
procedure addresult(var res:string; c:char);
begin
if test_small(c) then
res:=res+Chr(ord(c) - 32)
else
res:=res+c;
end;
```

```
var i:integer;
s,res:string;
begin
writeln();
readln(s);
results:='';
addresult(results,s[1]);
for i:=2 to length(s) do
begin
if s[i-1]=' ' then
if (s[i-2]='.') or (s[i-2]='!') or (s[i-2]='?') then
addresult(results,s[i])
else
begin
if test_small(s[i]) then results:=results+s[i]
else results:=results+Chr(ord(s[i]) + 32);
end
else
results:=results+s[i];
end;
writeln(results);
end.
```

Реализация в VBA содержит функцию test_small, возвращающую значение типа Boolean, процедуру addresult, которому передается значение по ссылке в скобках умолчанию.

```
Function test_small(tc As String) As Boolean
If tc >= "a" And tc <= "я" Then
test_small = True
Else
test_small = False
End If
End Function
```

```
Sub addresult(res As String, c As String)
If test_small(c) Then
res = res + Chr(Asc(c) - 32)
Else
res = res + c
End If
End Sub
```

```

Sub task24_2()
Dim s As String
Dim result As String
result = ""
s = InputBox("введите строку")
Call addressult(result, Mid(s, 1, 1))
For i = 2 To Len(s)
    If Mid(s, i - 1, 1) = " " Then
        If Mid(s, i - 2, 1) = "." Or _
            Mid(s, i - 2, 1) = "?" Or _
            Mid(s, i - 2, 1) = "!" Then
            Call addressult(result, Mid(s, i, 1))
        Else
            If test_small(Mid(s, i, 1)) Then
                result = result + Mid(s, i, 1)
            Else
                result = result + Chr(Asc(Mid(s, i, 1)) + 32)
            End If
        End If
    Else
        result = result + Mid(s, i, 1)
    End If
Next i
MsgBox (result)
End Sub

```

Приведенные коды и блок-схемы являются допустимыми вариантами решения поставленной задачи.

Возможна также реализация алгоритма без применения функций и процедур, но надо понимать, что логика подобного алгоритма будет сложна.

Возможно введение дополнительных функций и процедур. Например, введение функции проверки, является ли символ заглавной буквой, или символом окончания предложения, или процедуры, которая добавляет новый символ в строку result, но с проверкой и коррекций символа до прописного. И т. д.

В завершение темы программирования поставленной задачи полезно обсудить вопрос тестирования написанной программы. Данный вопрос не входит в исходную задачу, но является важным при проверке написанных программ. Для проверки программы надо подать на вход различные входные тесты, и проверить результат.

Далеко не всегда при получении правильного результата можно делать вывод о правильной работе программы:

- в некоторых случаях правильный результат получает по случайному совпадению (или когда одна ошибка накладывается на другую ошибку);

- или программа при некоторых входных тестовых данных работает правильно, а при иных – обрабатывает некорректно.

Таким образом, встает вопрос грамотного составления тестовых входных данных, которые позволили бы за минимальное количество запусков программы проверить ее работоспособность.

Воспользуемся для тестирования стихотворением Некрасова «Нежатая полоса». В качестве первой тестовой строки можно использовать входное предложение:

«Поздняя осень. грачи улетели, Лес обнажился, Поля опустели.»
получив в качестве выходных данных строку:

«Поздняя осень. Грачи улетели, лес обнажился, поля опустели.»

Но предложенный тестовый набор неполон:

- в нем отсутствуют символы «?», «!»
- не проверяется первая буква предложения
- не рассматриваются крайние случаи (слово из одной буквы и т.п.)

И, если первое и последнее замечания можно внести в предложенный тестовый набор, то проверить коррекцию первой буквы можно только повторным тестом.

Следовательно, делаем вывод, что для всестороннего тестирования бывает необходимо разработать набор тестовых входных последовательностей. Например, для тестирования написанной программы можно предложить следующую строку:

«нет! мы не хуже других и давно В нас налилось И созрело зерно.»

В данной тестовой строке проверяется коррекция первого символа, коррекция символа после «!», коррекция одиночного символа.

В качестве завершающего тестового набора стоит проверить предложения с «?»:

«Где Же наш пахарь? чего еще ждет?»

Внимание: в каждом тестовом наборе специально вносится некоторое количество «ошибок дошкольника», причем в каждом тестовом наборе часть слов обязательно написана корректно.

Подобный тестовый набор (из трех строк) с большой долей вероятности обеспечивает корректность работы программы.

Но надо понимать, что входная строка:

«Кажется, шеПчут колосья друг другу: «скучно нам слушать осеннюю вьюгу»»

не будет должным образом исправлена:

- в программу согласно исходному условию задачи исходно не заложена коррекция букв в середине слова

- аналогично в программе не предусмотрены такие знаки препинания, как кавычки, двоеточия, и проч., а также – правила их обработки.

Следовательно, подобные тестовые строки в данной ситуации неуместны.

Также не стоит добавлять к перечисленным тестовым последовательностям, например, следующую:

«Только **Не** сжата полоска одна. грустную думу наводит **Она.**»

по причине того, что данный набор по сути совпадает с первым: тестируются символы-начала предложений и требуется коррекция заглавных букв в середине предложения.

В результате можно сделать вывод:

- Тестовые данные должны быть подобраны так, чтобы всесторонне проверять все ситуации задачи.

- Их количество не должно быть слишком велико, но достаточно для того, чтобы охватить различные ситуации.

- Тестовые данные должны учитывать условия задачи и не тестировать непредусмотренные ситуации; применительно к данной задаче наличие в исходной строке числовых или английских символов не предусмотрено условием. Очевидно, что их наличие приведет к неверным результатам. А в некоторых случаях – может нарушить работу программы.

Типичные ошибки, допущенные при реализации задачи

1. Отсутствие навыков работы с символьными и строковыми типами данных

2. Ошибки алгоритма: непредусмотренные или некорректно обработанные «ошибки дошкольника».

Заключение

Олимпиада «МИСиС зажигает звезды» предполагает решение заданий вне дисплейного класса. При этом разрешается приводить в качестве решения программы, написанные на любом языке программирования. Во избежание спорных ситуаций советуем приводить комментарии к программе. Особо актуальна данная рекомендация при использовании мало распространенных языков программирования (к широко распространенным авторы относят любой Basic, любой Pascal, любой C).

Авторы будут благодарны за отзывы и замечания читателей по данному пособию и ждут Ваших предложений по адресу galo-krab@mail.ru, sidorovsv@mail.ru.

Учебное издание

КРЫНЕЦКАЯ Галина Сергеевна
СИДОРОВ Сергей Васильевич

**Методическое пособие по подготовке к олимпиадам
школьников инженерной направленности**

**Информационно-технологическое направление
«Математика в информатике.
Решение олимпиадных задач»**

9–10-й классы

В авторской редакции

Подписано в печать 06.12.17	Бумага офсетная	Уч.-изд. л. 4,25
Формат 60 × 90 ¹ / ₁₆	Печать офсетная	Заказ

Национальный исследовательский
технологический университет «МИСиС»,
119049, Москва, Ленинский пр-т, 4

Издательский Дом МИСиС,
119049, Москва, Ленинский пр-т, 4
Тел. (495) 638-45-22

Отпечатано в типографии Издательского Дома МИСиС
119049, Москва, Ленинский пр-т, 4
Тел. (499) 236-76-17, тел./факс (499) 236-76-35