№ 11 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

# ИНФОРМАТИКА

## “Динозавру” NC — 30 лет!

электронная ВЕРСИЯ ЖУРНАЛА в Личном кабинете на сайте www.1september.ru

издательский дом 1september.ru
Первое сентября | ноябрь 2015
ИНФОРМАТИКА Подписка на сайте www.1september.ru или по каталогу «Почта России»: 79066 — бумажная версия, 12684 — CD-версия

## НА ОБЛОЖКЕ

Наши ученики (да и некоторые наши молодые коллеги) много лучше представляют себе, скажем, птеродактиля, нежели, например, Norton Commander или VisiCalc. Но в недолгой истории программных продуктов было немало "динозавров", оставивших столь заметный след, что помнить их будут еще долгие годы. "Лексикон", "Тетрис", "Микромир" (последний, конечно, наше родное и специальное). Легенды. У каждой — своя история и своя судьба. Свой взлет, свое угасание. Одной из таких легенд — NC — исполняется 30 лет.

## В НОМЕРЕ



## В ЛИЧНОМ КАБИНЕТЕ

### Облачные технологии от Издательского дома "Первое сентября"

Все подписчики журнала имеют возможность получать электронную версию, которая является полной копией бумажной. Для получения электронной версии:

1) Откройте Личный кабинет на портале "Первое сентября" (www.1september.ru).

2) В разделе "Газеты и журналы / Получение" выберите свой журнал и кликните на кнопку "Я — подписчик бумажной версии".

3) Появится форма, посредством которой вы сможете отправить нам копию подписной квитанции.

После этого в течение одного рабочего дня будет активирована электронная подписка на весь период действия бумажной. Справки: podpiska@1september.ru или через службу поддержки на портале "Первого сентября".


ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ
по каталогу "Почта России": 79066 — бумажная версия, 12684 — электронная версия

http://inf.1september.ru

Учебно-методический журнал для учителей информатики
*Основан в 1995 г.*
Выходит один раз в месяц

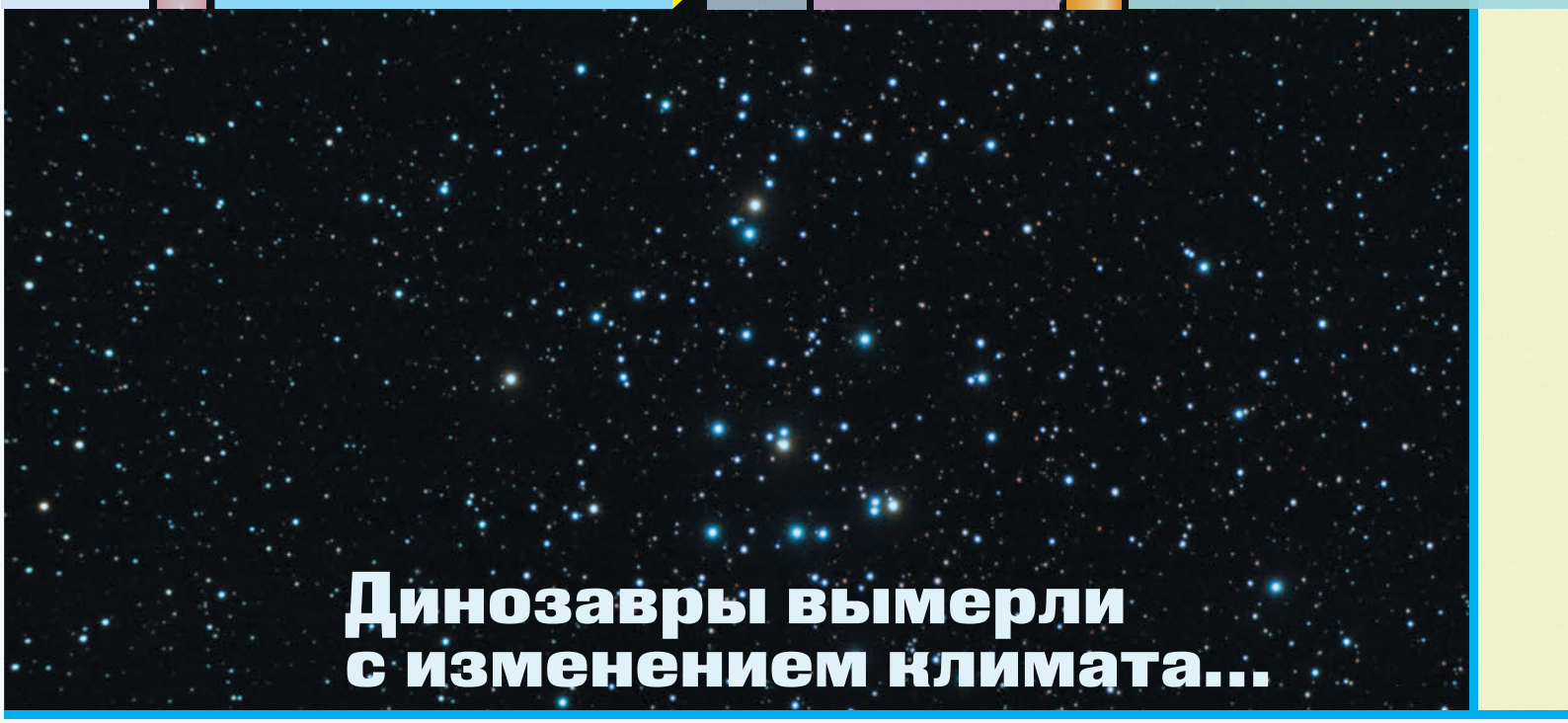
РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки "В мир информатики")
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 18 000 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ "ПЕРВОЕ СЕНТЯБРЯ"
**Главный редактор:** Артем Соловейчик (генеральный директор)
**Коммерческая деятельность:** Константин Шмарковский (финансовый директор)
**Развитие, IT и координация проектов:** Сергей Островский (исполнительный директор)
**Реклама, конференции и техническое обеспечение Издательского дома:** Павел Кузнецов
**Производство:** Станислав Савельев
**Административно-хозяйственное обеспечение:** Андрей Ушков
**Педагогический университет:** Валерия Арсланьян (ректор)

**Школа цифрового века:** facebook.com/School.of.Digital.Age

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА "ПЕРВОЕ СЕНТЯБРЯ"
**Английский язык** – Е.Богданова
**Библиотека в школе** – О.Громова
**Биология** – Н.Иванова
**География** – и.о. А.Митрофанов
**Дошкольное образование** – Д.Тюттерин
**Здоровье детей** – Н.Сёмина
**Информатика** – С.Островский
**Искусство** – О.Волкова
**История** – А.Савельев
**Классное руководство и воспитание школьников** – М.Битянова
**Литература** – С.Волков
**Математика** – Л.Рослова
**Начальная школа** – М.Соловейчик
**Немецкий язык** – М.Бузоева
**ОБЖ** – А.Митрофанов
**Русский язык** – Л.Гончар
**Спорт в школе** – О.Леонтьева
**Технология** – А.Митрофанов
**Управление школой** – Е.Рачевский
**Физика** – Н.Козлова
**Французский язык** – Г.Чесновицкая
**Химия** – О.Блохина
**Школа для родителей** – Л.Печатникова
**Школьный психолог** – М.Чибисова

УЧРЕДИТЕЛЬ:
ООО "ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»"
**Зарегистрировано ПИ № ФС77-58447 от 25.06.2014 в Роскомнадзоре**
Подписано в печать:
по графику 12.08.2015,
фактически 12.08.2015
Заказ №
Отпечатано в ОАО "Первая Образцовая типография"
Филиал "Чеховский Печатный Двор"
ул. Полиграфистов, д. 1,
Московская область,
г. Чехов, 142300
Сайт: www.chpd.ru
E-mail: sales@chpk.ru
Факс: 8 (495) 988-63-76
АДРЕС ИЗДАТЕЛЯ:
ул. Киевская, д. 24,
Москва, 121165
**Тел./факс:** (499) 249-31-38
**Отдел рекламы:**
(499) 249-98-70
**http://1september.ru**
ИЗДАТЕЛЬСКАЯ ПОДПИСКА:
**Телефон:** (499) 249-47-58
**E-mail: podpiska@1september.ru**



# Динозавры вымерли с изменением климата...

…А “динозавры” программного обеспечения — с приходом новых поколений операционных систем.

Лет 20 назад планирование, при котором все уроки информатики в четверти были посвящены изучению возможностей программы Norton Commander, не вызывало особого удивления. Сейчас среди наших молодых коллег удивление может вызвать даже само название NC — многие о таком никогда не слышали. Что же такого уникального было в двухпанельном файловом менеджере, что на протяжении стольких лет он был практически непременным атрибутом персоналок нескольких поколений, породил множество клонов и ремейков, некоторые из которых живы и поныне?

Не знаю. Нет, правда, я потратил кучу времени и перечитал кучу статей и заметок на эту тему. Рискну сказать, что ни в одной из них я не встретил ни одного убедительного объяснения секрету популярности NC. Это, кстати, с точки зрения образовательных целей, на мой взгляд, совсем не так плохо. Если у вас есть пытливые ребята — почему бы им не поломать голову? Задачка-то интересная! Факт бешеной популярности имеет место — это общепризнано. А вот причины — загадка.

Что, как правило, пишут в этих статьях и заметках?

Что Microsoft в эпоху DOS не озаботилась удобной оболочкой командной строки. Чистая правда. Но NC точно не был единственной оболочкой. Названия иных решений давно канули в лету, но решения эти точно были. При этом в годы тотального господства NC на их долю не приходился и 1% этого рынка.

Что NC был интуитивно понятным и очень удобным. Да простят меня повзрослевшие пользователи NC (сам к таким отношусь), но тезис про понятность и удобство весьма спорный. В тех же статьях в виде горьких шуток приводится множество реальных случаев, когда люди решали, что им не нужны два экземпляра файла в двух панелях, и убивали “лишний”.

Что беспрецедентную популярность NC имел только в России и на пространстве бывшего Союза. Что его распространенность на родине (на Западе) была существенно меньше. Похоже, что это так (можно встретить множество русскоязычных статей о “феномене NC” и ни одной англоязычной). Однако этот факт, скорее, в копилку вопросов, нежели ответов.

Что популярности программы во многом поспособствовал бестселлер Фигурнова — культовая книга русскоязычных пользователей ПК. Наверняка — да. Но и тут не все так однозначно с причиной и следствием, — в ту же книгу Фигурнова NC уже попал как крайне популярная по факту программа.

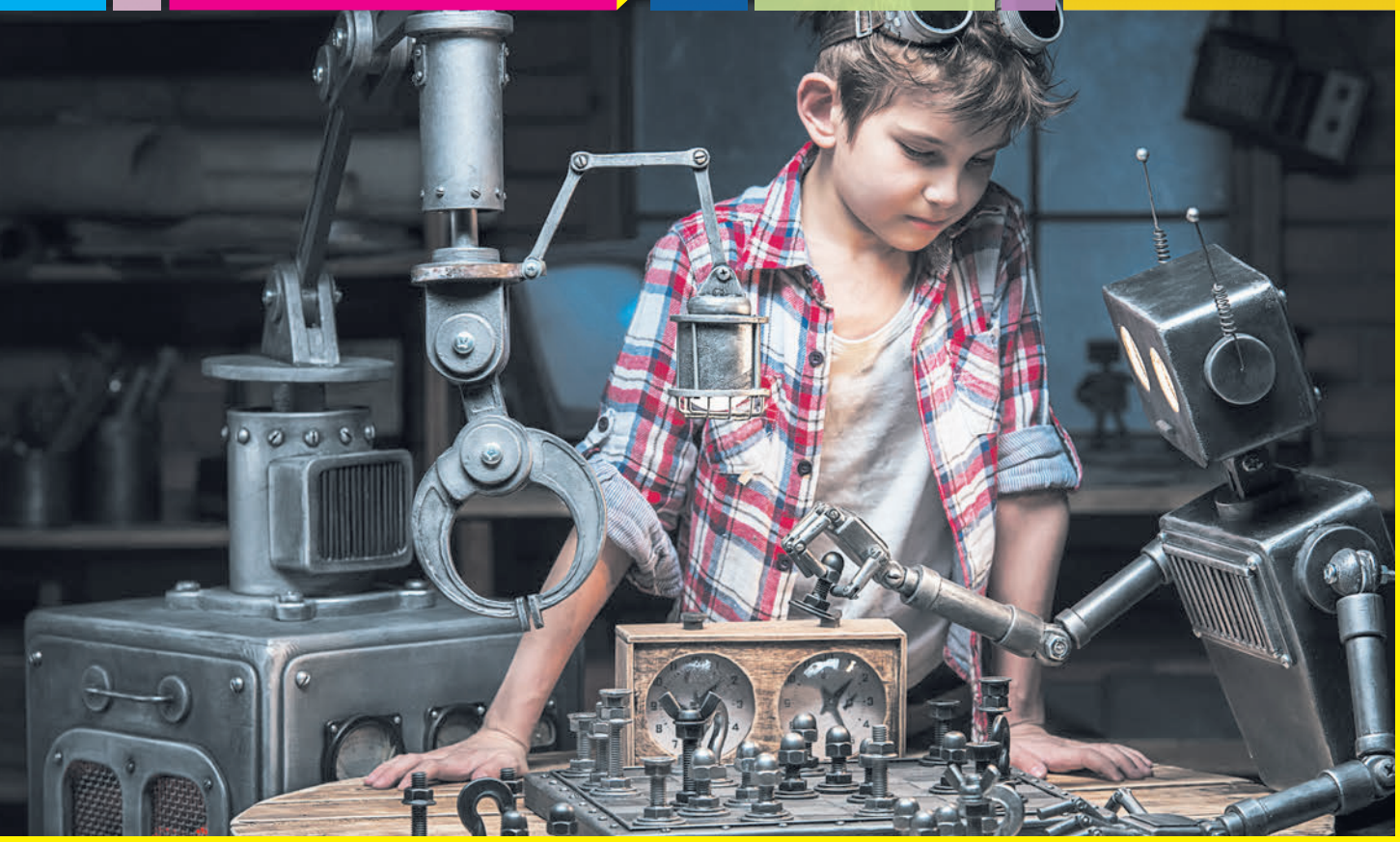
Что скринсейвер NC “звездное небо” зачаровывал пользователей. Кстати, да — хороший был скринсейвер. Но все же этого, согласитесь, маловато.

В общем — сплошные вопросы… А у вас, если вы, конечно, знаете, о чем речь ☺, есть ли своя версия?

А NC между тем исполнилось 30 лет.

**P.S.** И, кстати, к самому Питеру Нортону Norton Commander имел крайне опосредованное отношение.

**С.Л. Островский**,
гл. редактор, *so@1september.ru*



# Робототехника

**К.Ю. Поляков**,
д. т. н., Санкт-Петербург,
*http://kpolyakov.spb.ru*,
**Е.А. Еремин**,
к. ф.-м. н., г. Пермь

***Условные обозначения:***

⊕ — материал для расширенного курса

## Введение

Ключевые слова:

- робот
- автономный робот
- андроид
- робототехника
- исполнительное устройство
- микроконтроллер
- датчик

### Роботы и робототехника

Конечно, вы слышали слово “робот” и знаете, что роботом называют техническое устройство, которое может заменить человека во время выполнения сложных, утомительных и опасных работ.

❶ *Используя дополнительные источники, выясните, как появилось слово “робот”. Кто его придумал?*

Большинство роботов не похожи на человека. На современных заводах используется огромное количество промышленных роботов — станков с числовым программным управлением (ЧПУ). Такие станки обрабатывают детали по заложенной в них программе. Для того чтобы перестроить станок на изготовление другого типа деталей, достаточно просто заменить программу.

Роботы используются на конвейерных линиях, изготавливающих микросхемы для компьютеров: процессоры, память и др.

Все более популярными становятся технологии 3D-печати, позволяющие с помощью роботов специального типа (3D-принтеров) послойно изготавливать различные детали.

Нас окружают автоматизированные системы, которые тоже можно назвать роботами, например, система управления движением поездов метро, система управления отопления дома.

Роботы в отличие от людей не устают, работают 24 часа в сутки, не болеют, могут работать при повышенной

и пониженной температуре, в опасных для человека условиях. Во многих странах используются роботизированные установки пожаротушения. Уже продаются бытовые роботы — автоматические пылесосы, которые могут убрать пыль и самостоятельно вернуться на место для подзарядки. В XXI веке роботам стали "доверять" даже выполнение хирургических операций.

Некоторые роботы похожи на человека, их называют *андроидами*. Самый известный робот-андроид *Asimo* выпущен компанией *Honda*.


*Робот Asimo (www.robotonline.net)*

Существует особый класс роботов, которые работают "самостоятельно". Они называются **автономными роботами**. Полностью автономный робот может:

- перемещаться и работать длительное время без вмешательства человека;
- собирать информацию об окружающей среде;
- приспосабливаться к изменению обстановки, изменяя алгоритмы своей работы.

Автономные роботы очень важны для исследования космоса и океана, существуют также беспилотные автомобили и самолеты.

❷ *Используя дополнительные источники, выясните:*

*— зачем используются беспилотные автомобили Google;*

*— что такое "дрон";*

*— какие автономные роботы применялись при исследованиях Луны и Марса?*

**Робототехника** — это наука о разработке и использовании автоматизированных технических систем.

❸ *Используя дополнительные источники, выясните, кто и когда впервые использовал слово "робототехника" и как оно записывается по-английски.*

Робототехника использует результаты таких наук, как механика, автоматика, кибернетика, информатика. Некоторые алгоритмы управления роботами используют элементы "искусственного интеллекта".

❹ *Используя дополнительные источники, найдите и запишите в тетрадь три закона робототехники, сформулированные известным писателем-фантастом. Согласны ли вы с ними?*

## Из чего состоит робот?

Прежде всего робот — это механическое устройство. Поэтому его постройка — это инженерная, конструкторская работа. Нужно, чтобы все детали были хорошо закреплены и робот не потерял равновесия и не развалился во время выполнения задания. В то же время, все подвижные детали должны исправно двигаться и крутиться.

Роботом нужно как-то управлять, поэтому необходима *система управления*, которая в современных роботах строится на *микроконтроллерах*. Микроконтроллер — это миниатюрный компьютер, все части которого размещены на одном кристалле кремния. Он содержит процессор, разъемы для управления внешними устройствами (порты), оперативную и постоянную память. В память микроконтроллера записана программа, которую он при включении сразу начинает выполнять.

❺ *Используя дополнительные источники, выясните, кто и когда впервые получил патент на микроконтроллер.*

Для того чтобы робот мог двигаться, нужны *исполнительные устройства* — моторы, которые преобразуют электрическую энергию в механическую энергию вращения. Автономные роботы чаще всего передвигаются с помощью колес или гусениц, для неровных поверхностей иногда используют шагающие системы. Существуют также ползающие и плавающие роботы.

Для того чтобы получать данные об окружающей обстановке, роботу нужны *датчики* (сенсоры, чувствительные элементы) — устройства, которые измеряют какую-то физическую величину и выдают информацию о ней в виде электрического сигнала. Поскольку этот сигнал затем будет обрабатывать цифровой компьютер, его нужно преобразовать в двоичный код, в число. Эту работу выполняет аналого-цифровой преобразователь (АЦП).


*Рис.* 1

Существуют различные типы датчиков (*рис.* 1):

- датчики температуры, давления, скорости;
- датчики освещенности, датчики цвета (различающие цвета);
- датчики касания ("кнопки");
- датчики расстояния (используют время, за которое ультразвук отражается от препятствия и возвращается обратно);
- инфракрасные датчики для поиска и обнаружения объектов;

и многие другие.

Более сложные роботы используют системы компьютерного зрения, которые могут "узнавать" простые предметы, определять их расположение в пространстве и достраивать невидимые части, используя информацию из своей базы данных.

Таким образом, встроенный микроконтроллер робота управляет исполнительными устройствами и обрабатывает данные, поступающие с датчиков (*рис.* 2).

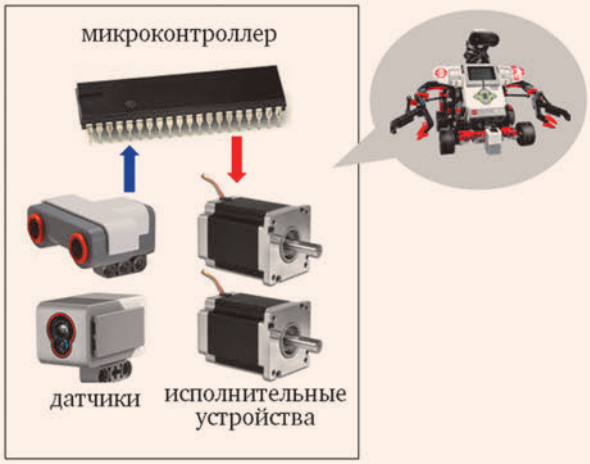



*Рис.* 2

❻ *Используя дополнительные источники, выясните, какие микроконтроллеры используются*
*— в наборах LEGO Mindstorms;*
*— на платах Arduino.*
*На какой тактовой частоте они работают?*

**Выводы:**

- Робот — это техническое устройство, которое может заменить человека во время выполнения сложных, утомительных и опасных работ.
- Автономные роботы способны собирать информацию об окружающей среде и работать длительное время без вмешательства человека.
- Робототехника — это наука о разработке и использовании автоматизированных технических систем.
- Системы управления роботов строятся на микроконтроллерах — миниатюрных компьютерах, все части которых размещены на одном кристалле кремния.
- Встроенный микропроцессор робота управляет исполнительными устройствами и обрабатывает данные, поступающие с датчиков.

❼ *Нарисуйте интеллект-карту этого параграфа.*

## Контрольные вопросы

1. В каких ситуациях, на ваш взгляд, роботы не способны заменить человека?
2. Какие требования должны предъявляться к встроенным микроконтроллерам?
3. Петя написал программу для управления роботом, которая занимает 6 Мбайт. С какими проблемами он может столкнуться?
4. Если у робота откажут все датчики, к каким последствиям это может привести?

**Темы сообщений:**

а) “Беспилотные транспортные средства”
б) “Боевые роботы”
в) “Роботы Big Dog”
г) “Роботы в медицине”
д) “Роботы-андроиды”

# Управление роботами

Ключевые слова:

- контакты ввода и вывода
- порты
- команды управления
- команды обратной связи

## Порты

Встроенный компьютер робота использует электрические сигналы для управления моторами, лампочками, устройствами вывода звука и т.п. Данные из внешнего мира (от датчиков) он тоже получает в виде электрических сигналов. Поэтому у него должны быть контакты ввода и вывода для связи с внешним миром. На *рис.* 3 показаны контакты ввода и вывода (так называемые “пины”) на плате *Arduino Uno*.

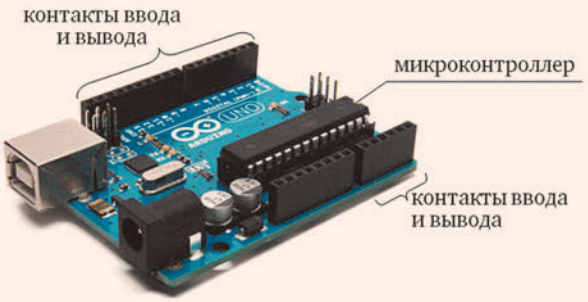



*Рис.* 3

❶ *Используя дополнительные источники, выясните, от какого слова образовано слово “пин”.*

Микроконтроллер может подать на любой контакт вывода электрический сигнал. Если мы подключим между этим выходом и сигналом “земля” (“минусом”) какую-нибудь лампочку, например, светодиод, то при подаче сигнала она загорится, причем яркость лампочки будет зависеть от величины сигнала (точнее, от величины электрического напряжения между контактом и “землей”). Светодиод имеет очень маленькое сопротивление, поэтому, для того чтобы ограничить силу тока, в цепь включают небольшой резистор (иначе светодиод может сгореть).

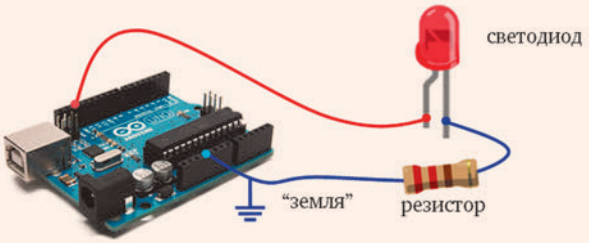



*Рис.* 4

❷ *Используя дополнительные источники, выясните, почему на электрических схемах и платах “земля” обозначается буквами GND.*

Часто используют не отдельные контакты, а многоконтактные разъемы — *порты*. Например, для ввода и вывода данных в настольных компьютерах и ноутбуках часто используют порты USB. На *рис.* 5 показан управляющий блок из набора *LEGO Mindstorms EV3*, имеющий четыре порта вывода (для управления моторами) и четыре порта ввода для приема данных с датчиков.

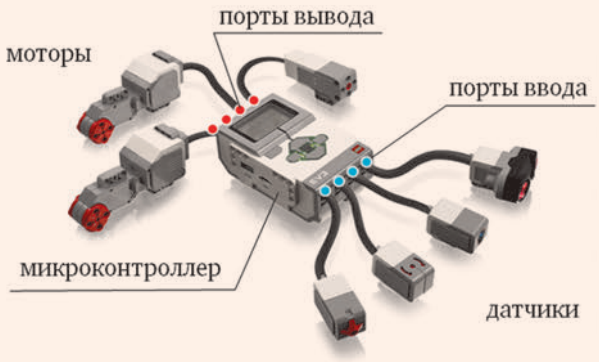



*Рис.* 5

## Управление лампочками

При записи алгоритмов управления отдельными контактами (“пинами”) мы будем использовать алгоритмический язык, расширив его командой

```
записать ( n, d )
```

Здесь **n** — это номер пина, а **d** — данные, которые определяют уровень сигнала на пине. Будем считать, что **d** — это целое число от 0 до 255, где 0 обозначает нулевой уровень сигнала (то же, что и “земля”), а 255 — максимальный уровень.

❸ *Определите, сколько бит нужно для хранения значения от 0 до 255? От 0 до 1023? От 0 до 65 535?*

Рассмотрим схему на *рис.* 4, в которой лампочка подсоединена к пину 11. Если ввести и выполнить такую программу:

```
записать ( 11, 255 )
ждать(1000)
записать ( 11, 0 )
```

то лампочка загорится, будет гореть 1000 миллисекунд (пока работает команда “ждать”) и снова потухнет. Можно сделать, чтобы лампочка мигала бесконечно долго, “завернув” эти команды в бесконечный цикл:

```
нц
  записать ( 11, 255 )
  ждать(1000)
  записать ( 11, 0 )
кц
```

В заголовке этого цикла нет ни условия, ни пределов изменений переменной. Это бесконечный цикл, который будет работать, пока мы не остановим программу. Такие циклы считаются плохим стилем в “обычном” программировании, но часто применяются при программировании роботов.

❹ *В только что рассмотренной программе есть ошибка. Исправьте ее.*

## Система команд роботов

Управлять движущимся роботом можно вручную или с помощью программы, заранее записанной в память микроконтроллера. Система команд робота, то есть перечень действий, которые он может выполнять, очень ограниченна. Как правило, можно

- задать уровень мощности каждого мотора (например, в процентах от максимальной);
- выбрать направление вращения;
- включить мотор;
- выключить мотор;
- подождать некоторое время.

В более сложных системах есть возможность провернуть вал мотора на определенное число оборотов или градусов.

Для приема данных с датчиков существуют команды обратной связи. Их действие сводится к тому, что встроенный компьютер читает числа, которые пришли с датчика на один из портов ввода. Что дальше делать с этими данными, определяет управляющая программа. Таким образом, алгоритмы работы роботов полностью определяются программным обеспечением и могут быть легко изменены.

## Управление без обратной связи

В задачах с движущимися роботами мы будем управлять моторами, поэтому расширим алгоритмический язык командой

```
мотор[n] = d
```

Здесь **n** — это порт вывода, а **d** — мощность мотора в процентах, которая нам требуется (от 0 до 100%). Будем обозначать порты для подключения моторов заглавными латинскими буквами: A, B, C и т.д. Вот так можно включить мотор A на полную мощность на две секунды, а затем выключить:

```
мотор[A] = 100
ждать(2000)
мотор[A] = 0
```

Далее мы будем работать с роботом, у которого два мотора, связанные с двумя ведущими колесами. Мотор 0 вращает левое колесо, а мотор 1 — правое.

❺ *Запишите программу, выполняя которую робот сначала проедет вперед, потом назад, потом снова вперед, и так будет продолжать бесконечно долго.*

## Практическая работа № 33. Управление без обратной связи

### Выводы:

- Встроенный компьютер робота обменивается данными с внешним миром с помощью электрических сигналов.
- Для ввода и вывода данных используются контакты ввода и вывода (так называемые “пины”).
- На каждый из контактов вывода микроконтроллер может подавать сигналы разных уровней.
- Часто несколько контактов вывода объединяют в многоконтактный разъем — порт.
- При простейшем управлении моторами движущегося робота можно установить уровень мощности мотора и включить мотор на заданное время.
- Для приема данных с датчиков существуют команды обратной связи. Их действие сводится к тому, что встроенный компьютер читает числа, которые пришли с датчика на один из портов ввода.

❻ *Нарисуйте интеллект-карту этого параграфа.*

### Контрольные вопросы

1. Что требуется для того, чтобы мотор можно было повернуть на заданное число оборотов?

2. Какие задачи управления движущимся роботом сложно решить, не используя сигналы с датчиков?

### Задачи

1. Напишите программу для моделирования работы железнодорожного семафора, в котором попеременно мигают две красные лампочки. Лампочки подключены к пинам 10 и 11.

2. Напишите программу для моделирования работы автомобильного семафора. Красная, желтая и зеленая лампочки подключены, соответственно, к пинам 10, 11 и 12.

3. Напишите программу, выполнив которую робот поворачивается на 90 градусов по часовой стрелке. Для выбора временной задержки используйте тренажер или реальную модель робота.

4. Напишите программу, выполняя которую робот будет двигаться по квадрату.

**Темы сообщений:**

а) “Платы Arduino”

б) “Порты персональных компьютеров”

## Алгоритмы управления роботами

Ключевые слова:

- порт ввода
- чтение из порта
- датчик касания
- датчик расстояния
- датчик освещенности
- ошибка измерения
- калибровка датчика
- движение по линии
- релейный регулятор
- П-регулятор

### Управляющие кнопки

Простейший управляющий элемент — кнопка, с помощью которой включается или выключается какой-то режим работы робота. Рассмотрим простой пример — управление светодиодом. При нажатии на кнопку лампочка должна загораться. Схема, которую можно собрать на плате *Arduino*, показана на *рис.* 6.

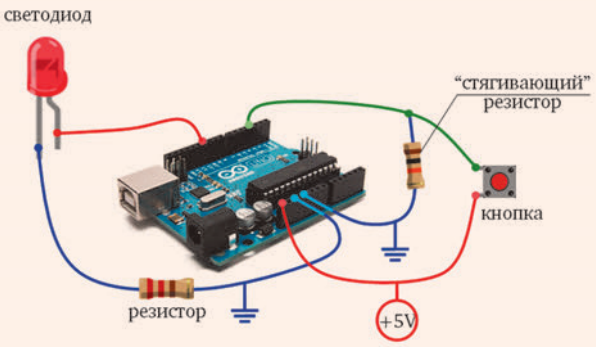



*Рис.* 6

На один контакт кнопки подается питание, второй подсоединен к входному пину, с которого микроконтроллер может прочитать значение сигнала. При нажатии на кнопку цепь замыкается и на входной пин подается сигнал высокого уровня (например, 5 В). А какой сигнал будет на этом пине, если кнопка не нажата? Оказывается, в этом случае он может “плавать”, то есть меняться случайным образом из-за электромагнитных шумов. Чтобы этого не происходило, ставят специальный “стягивающий” резистор, который связывает входной пин с сигналом нулевого уровня.

Перейдем к программированию. Нам нужно постоянно читать значение из пина. Если на пин подан высокий уровень сигнала (кнопка нажата), нужно включить лампочку, то есть подать сигнал высокого уровня (255) на выходной пин, с которым она связана. Если кнопка не нажата, на выходной пин подаем сигнал низкого уровня (0), лампочка выключается.

Будем считать, что лампочка управляется пином 11, а сигнал от кнопки приходит на пин 2. Команда **прочитать** читает значение из пина, номер которого записан в скобках. Это функция, она возвращает значение, которое можно записать в переменную.

```
цел x
нц
  x := прочитать ( 2 )
  если x = 255
    записать ( 11, 255 )
  иначе
    записать ( 11, 0 )
кц
```

❶ *Запишите в тетради вариант программы, в которой нет оператора “если”.*

❷ *Запишите в тетради программу для схемы, которая работает “наоборот” — когда кнопка нажата, лампочка не горит, а когда отпущена — лампочка включается. Постарайтесь найти два варианта программы.*

### Движение в лабиринте

Теперь перейдем к программированию движущихся роботов в лабиринте. Роботу нужно как-то ориентироваться — определить, куда можно ехать, а куда нельзя. Если человеку завязать глаза, то он все равно сможет обнаружить стену — на ощупь. У роботов для этого служат *датчики касания*. Они подают на входной порт значение 0, если касания нет, и значение 255, если касание произошло. Если перед роботом есть стена, то, для того чтобы доехать до стены, нужно использовать цикл “пока не сработал датчик касания”.

Будем считать, что у платы управления роботом есть порты ввода с номерами 0, 1, 2 и т.д., и датчик касания подключен к порту 0. Программу движения до стенки можно записать так:

```
нц пока датчик[0] <> 255
  мотор[A] := 100
  мотор[B] := 100
  ждать(1)
кц
мотор[A] := 0
мотор[B] := 0
```

Мы дополнили язык программирования командой **датчик[A]**. Это функция, которая возвращает число, записанное датчиком в порт ввода A. Команда **ждать** в конце тела цикла нужна для того, чтобы микроконтроллер смог при необходимости выполнить какие-то другие задачи, а не "зависал" во время выполнения цикла.

❸ *Как вы думаете, можно ли сразу разворачивать робота влево или вправо, если сработал датчик касания? Проверьте ваш вывод на тренажере или при эксперименте с роботом.*

Чаще всего не нужно допускать столкновение робота с препятствием — его лучше остановить заранее. Для решения этой задачи датчик касания уже не подходит — нужно определять расстояние до ближайшей стенки. Для этого используют датчики расстояния, чаще всего ультразвуковые (их еще называют *сонарами*).

❹ *Используя дополнительные источники, выясните, откуда произошло слово "сонар". Что оно означает?*

❺ *Используя дополнительные источники, выясните, что такое ультразвук. Можно ли его услышать?*

Ультразвуковые датчики расстояния состоят из излучателя и приемника. Датчик излучает звуковую волну высокой частоты, она отражается от препятствия и возвращается обратно, где ее "ловит" приемник. Чем больше интервал между запуском волны и приемом отраженного сигнала, тем больше расстояние.

Значение, поступающее на выходной порт с датчика расстояния[1], — это целое число от 0 до 255. Как же нам определить само расстояние, например, в сантиметрах? Для этого нужно прочитать описание этого датчика и выяснить, как вычислить расстояние по коду. Например, если датчик измеряет расстояния от 0 до 10 м, то расстояние в сантиметрах можно получить так:

$$S = d \cdot \frac{10 \cdot 100}{255} \text{ см},$$

где $d$ — числовое показание датчика. Обычно максимальное кодовое значение (в нашем случае — 255) означает, что препятствий не обнаружено.

❻ *Определите, какому расстоянию соответствует показание датчика 100.*

❼ *Определите, какое показание датчика соответствует расстоянию 20 см.*

Пусть нам нужно остановить робота на расстоянии 0,3 м от стены. Вычислив соответствующее значение $d$, получаем 7,65. Но показание датчика должно быть целым числом, поэтому округляем это значение до 8.

У любого датчика есть неизбежная **ошибка измерения**, вызванная неидеальностью деталей и электромагнитными помехами от систем самого робота и других устройств. Обычно допустимая ошибка составляет одну единицу, то есть показание датчика вместо 8 может быть равно 7 или 9. Для $d = 7$ получаем расстояние 27,45 см, а для $d = 9$ — значение 35,39 см. Это значит, что с помощью такого датчика мы определим расстояние с ошибкой ±5 см. Относительная ошибка в процентах составит

$$\Delta S = \frac{5}{30} \cdot 100\% \approx 17\% .$$

❽ *Как вы думаете, при увеличении измеряемого расстояния, скажем, до 1 м относительная ошибка будет уменьшаться или увеличиваться? Проверьте свое предположение вычислениями.*

Будем считать, что датчик расстояния "смотрит" вперед (по направлению движения) и подключен к порту ввода A. Программа, согласно которой робот едет вперед и останавливается, очень похожа на предыдущую:

```
нц пока датчик[A] > 8
  мотор[A] := 100
  мотор[B] := 100
  ждать(1)
кц
мотор[A] := 0
мотор[B] := 0
```

❾ *Объясните, почему не стоит записывать условие работы цикла в виде*

```
нц пока датчик[A] <> 8
  ...
кц
```

## Движение по линии

В последние годы проводятся многочисленные соревнования роботов, в том числе Всемирные олимпиады. Классические задачи спортивной робототехники — движение робота по линии. Мы рассмотрим только одну задачу этого типа: робот должен ехать вдоль границы между черной и белой полосами. Для того чтобы "увидеть" эту границу, робот использует один или несколько датчиков освещенности (*рис.* 7).

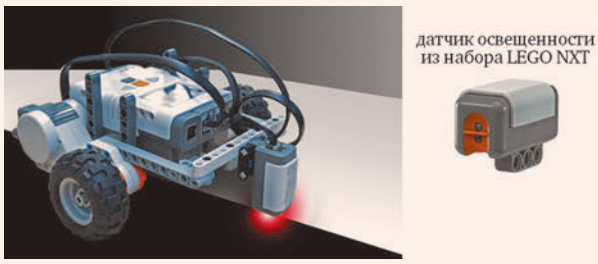



*Рис.* 7

Датчик освещает поверхность светодиодом и измеряет количество отраженного света. Черные предметы отражают меньше всего света, белые — больше всего.

Датчик освещенности выдает код в интервале от 0 до 255, где 0 означает полное отсутствие отраженного света, а 255 — полное отражение. Однако не существует идеально черных и идеально белых предметов. Даже полоска, которая кажется нам совсем черной, отражает немного света, поэтому датчик покажет на ней не 0, а, например,10. Также и на белой полосе вместо 255

[1] Многие реальные датчики выдают цифровое значение в диапазоне от 0 до 1023.

мы получим, например, 230. Среднее арифметическое между этими значениями называют “средним значением серого”. Теоретически такой сигнал выдает датчик, когда он находится точно над границей черной и белой полос (*рис.* 8). К этому состоянию мы и будем стремиться.

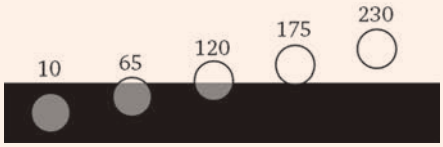



*Рис.* 8

Определение среднего значения серого в конкретных условиях называется *калибровкой датчика*.

Пусть требуется, чтобы слева от робота была белая полоса, а справа — черная (как на *рис.* 7). Для управления мы будем использовать очень простой принцип, который называют **релейным регулированием**. Если показания датчика больше, чем 120 (среднее значение серого), робот зашел на белую полосу и нужно повернуть его вправо. Если датчик показывает меньше, чем 120, поворачиваем влево.

10 *Используя дополнительные источники, узнайте, что такое реле. Откуда произошло это слово?*

11 *Запишите в тетради команды управления моторами для поворота влево и вправо. Мотор A находится с левой стороны робота, мотор B — с правой.*

Получается такая программа, работающая в бесконечном цикле:

```
нц
  если датчик[A] > 120 то
    motorA := 100
    motorB := 0
  иначе
    motorA := 0
    motorB := 100
  все
кц
```

Если запустить такую программу, мы увидим, что робот движется зигзагом, рывками. Дело в том, что мы задали ему такой алгоритм — резкое переключение между двумя состояниями. Это и есть релейное управление.

Для более плавного движения нужно создавать вращающий момент, который зависит от ошибки, то есть от того, насколько робот отклонился от линии. Чем меньше ошибка, тем меньше вращающий момент. Если ошибки нет (датчик выдает значение, равное среднему значению серого), подаем одинаковую мощность на оба мотора, робот едет прямо.

Можно использовать, например, такой закон управления:

```
цел k = 0.5
нц
  u = k*(120 - датчик[A])
  мотор[A] = 50 - u
  мотор[B] = 50 + u
кц
```

12 *Изучите программу и определите, какой сигнал подается на моторы, когда датчик выдает значение 100? 120? 140?*

Выражение `120 - датчик[A]` — это ошибка управления, то есть отклонение значения, полученного с датчика, от заданного значения 120. Сигнал управления $u$, который изменяет мощность моторов, пропорционален ошибке, поэтому такой регулятор называется **пропорциональным**, или **П-регулятором**.

Коэффициент $k$ — это коэффициент усиления П-регулятора. Правила выбора коэффициентов в законах управления изучает *теория автоматического управления*. Мы же будем выбирать коэффициенты экспериментально, не углубляясь в теорию.

Чем больше $k$, тем сильнее реакция робота на ошибку; чем меньше $k$, тем более плавно движется робот. Но при малых значениях $k$ его реакция замедлена, и он может потерять линию на повороте.

13 *Проведите эксперимент (с тренажером или с реальным роботом) и выясните, при каком наименьшем коэффициенте k робот проходит всю трассу.*

14 *Проведите эксперимент и выясните, что происходит при выборе слишком большого значения k. При каком наибольшем коэффициенте k робот выполняет задание?*

15 *По результатам экспериментов выберите наилучшее, на ваш взгляд, значение k.*

## Практическая работа № 34. Движение в лабиринте

## Практическая работа № 35. Движение по линии

Выводы:

- Для получения данных от внешних источников микропроцессор читает значения с портов, к которым подключены датчики.
- Для движения в помещении с препятствиями роботы могут использовать датчики касания и датчики расстояния (чаще всего ультразвуковые).
- Числовой код, полученный с датчика, нужно пересчитать в значение физической величины в соответствии с документацией на датчик.
- У датчика есть ошибка измерения, вызванная неидеальностью деталей и электромагнитными помехами от систем самого робота и других устройств.
- При движении робота по линии используется один или несколько датчиков освещенности.
- Для удержания робота на линии используют регуляторы — автоматические управляющие устройства.
- Релейный регулятор включает один из двух режимов движения в зависимости от сигналов с датчиков.
- Пропорциональный регулятор (П-регулятор) вырабатывает сигнал управления, пропорциональный ошибке управления. Для выбора коэффициента усиления П-регулятора используют методы теории автоматического управления или эксперимент.

16 *Нарисуйте интеллект-карту этого параграфа.*

## Контрольные вопросы

1. Почему в программах управления роботами часто используются бесконечные циклы?

2. У робота вышел из строя датчик касания. Можно ли использовать вместо него датчик расстояния, который остался работоспособен?

3. Как нужно установить датчик расстояния, чтобы робот мог двигаться параллельно стенке?

4. У вас оказался датчик расстояния, на который нет документации. Как вы будете его использовать?

5. Как нужно установить два датчика освещенности, если нужно обеспечить движение робота вдоль тонкой черной линии?

6. Почему на черной линии датчик освещенности выдает ненулевое значение?

7. Зависит ли результат, который выдает датчик освещенности, от внешнего освещения?

8. Какой регулятор обеспечивает более плавное движение: релейный или пропорциональный?

## Задачи

1. Напишите программу работы кнопочного выключателя: первое нажатие кнопки включает лампочку, второе — выключает, следующее — снова включает и т.д.

2. Напишите программу движения робота по границе черной и белой областей, так чтобы слева от робота оставалась черная полоса.

3. Напишите программу движения робота вдоль стены. Датчик расстояния находится на оси робота и направлен вправо перпендикулярно ей.

4. Напишите программу для работы светофора на перекрестке. Три лампочки (красная, желтая и зеленая) регулируют движение автомашин, еще две (красная и зеленая) — переход улицы пешеходами. Пешеходный переход включается только тогда, когда кто-то нажмет кнопку.

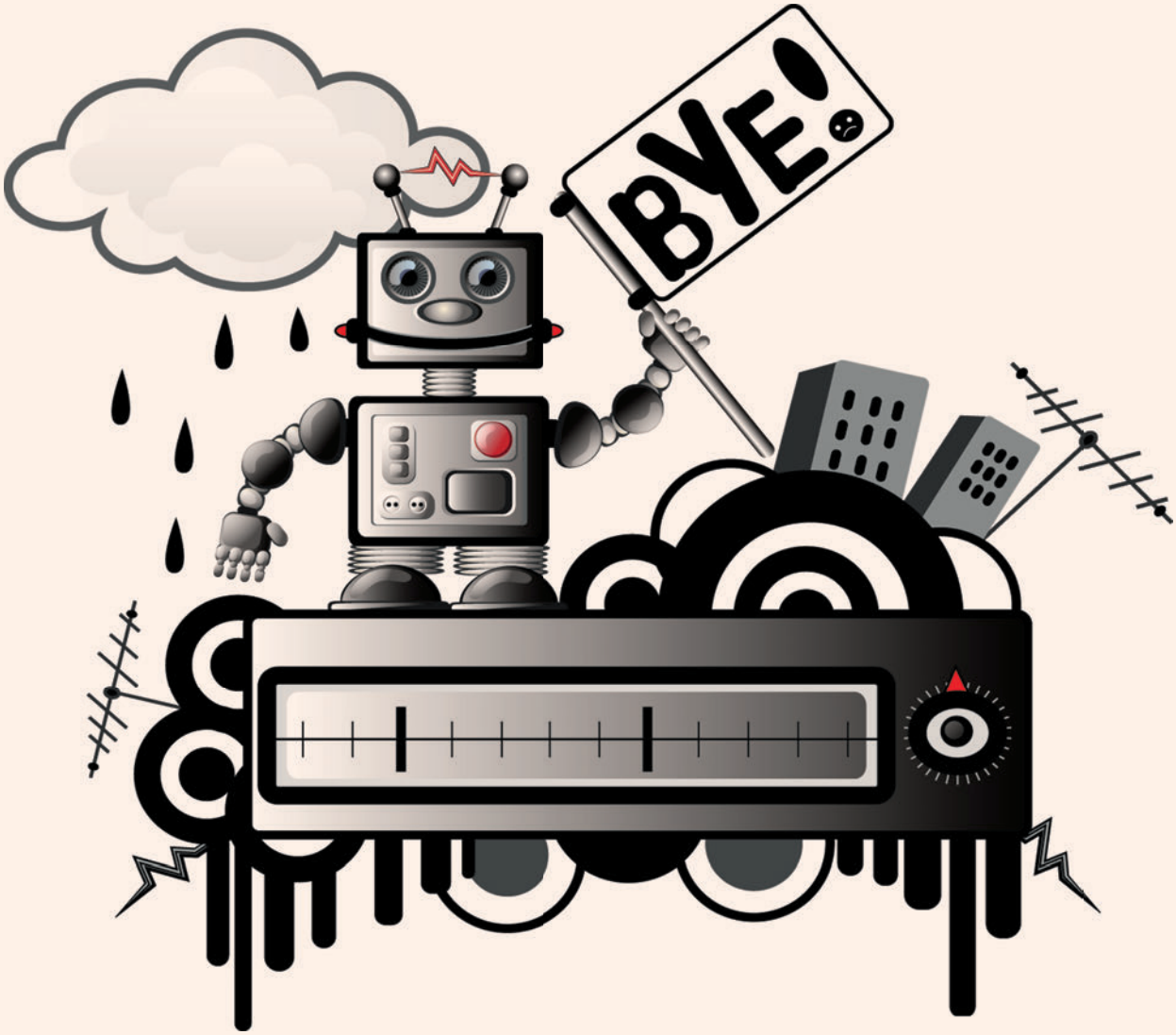
**Темы сообщений:**

а) “Соревнования роботов”

б) “Кегельринг”

в) “ПИД-регулятор”

Для изучения алгоритмов управления роботами LEGO и платой Arduino можно использовать разработанные автором компьютерные тренажеры, работающие в современных браузерах. Они размещены по адресу *http://kpolyakov.spb.ru/school/robotics.htm*.



# Таблицы выигрышных позиций в задачах на стратегию

**К.Ю. Поляков**,
д. т. н., Санкт-Петербург,
*http://kpolyakov.spb.ru*

*На фото: Пейзаж после сдачи ЕГЭ.*

Согласно спецификации КИМ ЕГЭ по информатике [1], в одной из задач (в 2015 году это была задача 26) требуется построить дерево игры по заданному алгоритму и обосновать выигрышную стратегию.

До 2011 года в КИМ включалась задача С3 с двумя кучами камней [2–3] или с фишкой на координатной плоскости [4–5]. В 2013–2014 годах на экзаменах предлагалась задача с одной кучей камней, где требовалось определить выигрышные стратегии при различном начальном количестве камней в этой куче [1]. Подробный разбор задач такого типа приведен в статье [6].

На экзамене 2015 года вновь появились задачи с двумя кучами, значительно более трудоемкие в сравнении с демонстрационным вариантом [1]. В опубликованном КИМ досрочного экзамена 2015 года [7] предлагается задача, в которой начальное количество камней в одной куче известно, а во второй может изменяться. Приведем (с некоторыми сокращениями) условие задачи из [7]:

*Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в одну из куч (по своему выбору) один камень или увеличить количество камней в куче в два раза. Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней.*

*Выигрывает игрок, первым получивший такую позицию, что в кучах всего будет 55 или больше камней. В начальный момент в первой куче было пять камней, во второй куче — S камней;* $1 \le S \le 49$.

***Задание 1.*** *а) Укажите все такие значения числа S, при которых Петя может выиграть за один ход, и соответствующие выигрывающие ходы. б) Сколько существует значений S, при которых Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом?*

***Задание 2.*** *Укажите такое значение S, при котором у Пети есть выигрышная стратегия, причем одновременно выполняются два условия:*

*— Петя не может выиграть за один ход;*

*— Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.*

*Для указанного значения S опишите выигрышную стратегию Пети.*

***Задание 3.*** *Укажите значение S, при котором одновременно выполняются два условия:*

*— у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;*

*— у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.*

*Для указанного значения S опишите выигрышную стратегию Вани.*

В настоящей статье предлагается способ решения таких задач, аналогичный [6], использующий таблицу выигрышных и проигрышных позиций. Задачи для тренировки с ответами интересующийся читатель может найти на сайте [8].

## Введение

Рассматриваемая игра относится к играм с полной информацией, в которой можно при известной начальной позиции сразу определить, кто выиграет при правильной игре (если противники не будут ошибаться).

Все игровые позиции делятся на выигрышные и проигрышные. Выигрышная позиция — это такая позиция, в которой игрок, делающий первый ход, может гарантированно выиграть при любой игре соперника, если не сделает ошибку. При этом говорят, что у него есть выигрышная стратегия — алгоритм выбора очередного хода, позволяющий ему выиграть.

Если игрок начинает играть в проигрышной позиции, он обязательно проиграет, если ошибку не сделает его соперник. В этом случае говорят, что у него нет выигрышной стратегии. Таким образом, общая стратегия игры состоит в том, чтобы своим ходом создать проигрышную позицию для соперника.

Выигрышные и проигрышные позиции можно охарактеризовать так:

• позиция, из которой все возможные ходы ведут в выигрышные позиции, — проигрышная;

• позиция, из которой хотя бы один из возможных ходов ведет в проигрышную позицию, — выигрышная, при этом стратегия игрока состоит в том, чтобы перевести игру в эту проигрышную (для соперника) позицию.

Через $(a, b)$ будем обозначать текущую позицию в игре, при которой в первой куче находится $a$ камней, а во второй — $b$ камней.

***Задание 1***

Начнем с **Задания 1а**: в позиции $(5, S)$ Петя должен выиграть первым ходом. После этого хода количество камней в двух кучах может стать равным

$6+S$ (после добавления одного камня в любую кучу);

$10+S$ (после удвоения количества камней в первой куче);

$5+2S$ (после удвоения количества камней во второй куче).

Выпишем условия выигрыша на первом ходу для всех трех вариантов:

$$6 + S \geq 55 \Rightarrow S \geq 49$$
$$10 + S \geq 55 \Rightarrow S \geq 45$$
$$5 + 2S \geq 55 \Rightarrow S \geq 25$$

Отсюда следует, что **при $S \geq 25$ Петя всегда может выиграть первым же ходом, удвоив количество камней во второй куче**.

Для дальнейшего анализа составим таблицу, где по вертикали будем отмечать количество камней в первой куче, а по горизонтали — во второй:

| | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|
| 5 | | | | | | | * | | |

*Рис.* 1

Например, ячейка, отмеченная звездочкой, соответствует позиции (5, 25). Это выигрышная позиция, все выигрышные позиции будем отмечать зеленым цветом. Если во второй куче 6 камней (вторая строка таблицы), выигрышная позиция определяется условием

$$6 + 2S \geq 55 \Rightarrow 2S \geq 49 \Rightarrow S \geq 25$$

(так же, как и для позиции $(5, S)$!)

| | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |

*Рис.* 2

Если во второй куче семь камней, получаем

$$7 + 2S \geq 55 \Rightarrow 2S \geq 48 \Rightarrow S \geq 24$$

и так далее:

| | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|
| 5 | | | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | | | | | | | | | |
| 9 | | | | | | | | | |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |
| 13 | | | | | | | | | |

*Рис.* 3

Теперь попробуем найти **проигрышные позиции** — такие позиции, из которых все ходы ведут в выигрышные позиции (отмеченные зеленым фоном).

Ход “+1” смещает позицию в таблице на одну клетку вправо или вниз, а ход “*2” — на соответствующее число клеток вправо или вниз. Легко показать, что все позиции в углах зеленой лесенки — проигрышные. На рисунке показаны все возможные ходы из позиции (6, 24), все они ведут в зеленые ячейки:

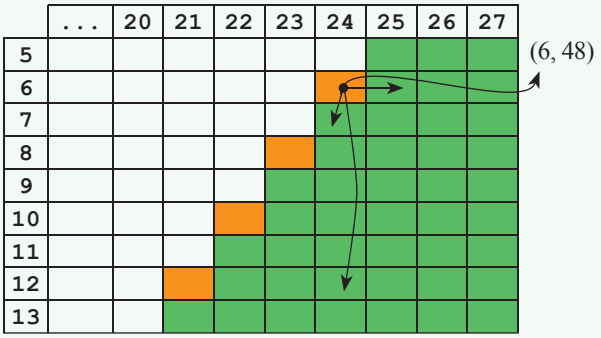



*Рис.* 4

Выигрышными будут все позиции, из которых есть ход хотя бы в одну проигрышную позицию:

| | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|
| 5 | | | | 2 | | 2 | | | |
| 6 | | | 2 | | 2 | | | | |
| 7 | | 2 | | | 2 | | | | |
| 8 | 2 | | | 2 | | | | | |
| 9 | | | | 2 | | | | | |
| 10 | | | 2 | | | | | | |
| 11 | | | 2 | | | | | | |
| 12 | | 2 | | | | | | | |
| 13 | | 2 | | | | | | | |

*Рис.* 5

Число два в ячейке таблицы указывает на выигрыш за два хода.

Перейдем к **Заданию 1б**. Как мы уже знаем, после первого хода Петя не может выиграть, то есть, как следует из решения Задания 1*а*, $S < 25$. Нам нужно найти в первой строке таблицы (когда в первой куче пять камней) позицию с проигрышем за один ход, то есть такую, из которой все возможные ходы ведут в пустые зеленые ячейки. Легко убедиться, что таких позиций нет. Например, в позиции (5, 23) один из возможных ходов ведет в позицию (5, 24), которая обеспечивает выигрыш за два хода, а не за один, как требуется. Поэтому **значений *S*, удовлетворяющих условию Задания 1б, нет**!

Несложно доказать это математически. Рассмотрим первый возможный ход Пети (6, *S*) и все возможные ответы Вани:

(6, *S*) → Ваня: (7, *S*) или (6, *S* + 1) или (12, *S*) или (6, 2*S*)

На каждый ход Пети у Вани должен быть выигрышный ход. Для хода Пети (6, *S*) получаем условия выигрыша Вани:

$$7 + S \geq 55 \quad \Rightarrow \quad S \geq 48$$
$$12 + S \geq 55 \quad \Rightarrow \quad S \geq 43$$
$$6 + 2S \geq 55 \quad \Rightarrow \quad S \geq 25$$

Таким образом, при ходе Пети (6, *S*) Ваня может гарантированно выиграть только при $S \geq 25$, но в этом случае Петя и сам может выиграть своим первым ходом! Остальные варианты первого хода Пети можно уже не проверять.

В первой строке построенной таблицы (для позиций, в которых в первой куче пять камней) нет оранжевой клетки (проигрышной позиции) между выигрышем в один ход (позиция (5, 25)) и выигрышем в два хода (позиция (5, 24)). Это связано с тем, что в позиции (5, 24) возможен “выжидающий” ход в проигрышную позицию (6, 24).

***Задание 2***

Поскольку Петя не может выиграть за один ход, то $S < 25$. Как мы выяснили ранее, позиция (6, 24) — проигрышная, она ведет к проигрышу за один ход. Поэтому возможный ответ на это задание — такое значение *S*, что у Пети есть ход, который переводит игру в позицию (6, 24).

Начав с позиции (5, 24), Петя может перевести игру в проигрышную позицию (6, 24), в которой Ваня не может выиграть одним ходом, но всегда создаст Пете выигрышную позицию на втором ходу.

Таким образом, при $S = 24$ Петя не может выиграть за один ход, но может выиграть за два. Для этого ему нужно добавить один камень в первую кучу, получив позицию (6, 24), которая является проигрышной. Для любого хода Вани в этой позиции есть выигрышный второй ход Пети — удвоение второй кучи:

(6, 24) → Ваня: (7,24) → Петя: (7,48)
(6, 24) → Ваня: (6,25) → Петя: (6,50)
(6, 24) → Ваня: (12,24) → Петя: (12,48)
(6, 24) → Ваня: (6,48) → Петя: (6,96)

Возможен и другой вариант. Дело в том, что при $S = 22$ Петя своим первым ходом тоже может получить проигрышную (для Вани) позицию, только другую: (10, 22). Поэтому при $S = 22$ Петя тоже не может выиграть за один ход, но может выиграть за два. Для этого ему нужно удвоить число камней в первой куче, получив позицию (10, 22), которая является проигрышной. Для любого хода Вани в этой позиции есть выигрышный второй ход Пети — удвоение второй кучи:

(10, 22) → Ваня: (11,22) → Петя: (11,44)
(10, 22) → Ваня: (10,23) → Петя: (10,46)
(10, 22) → Ваня: (20,22) → Петя: (20,44)
(10, 22) → Ваня: (10,44) → Петя: (10,88)

***Задание 3***

Нам нужно найти такое значение *S*, что из начальной позиции (5, *S*) **любой** ход Пети ведет в выигрышную (для Вани) позицию. Попробуем первое нерассмотренное значение, $S = 23$. Возможные ходы Пети: (6, 23), (5, 24), (10, 23) и (5, 46). В первых двух случаях Ваня своим ходом сведет игру в проигрышную (для Пети) позицию (6, 24) и выиграет своим вторым ходом, а в последних двух случаях выиграет на первом же ходу, удвоив число камней во второй куче.

Еще легче определить нужный ход по таблице. Все клетки в углах новой лесенки — это проигрышные позиции, потому что все ходы из них ведут на зеленые клетки:

| | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|---|---|---|---|---|---|---|---|---|---|
| 5 | | | 2 | 2 | 2 | 2 | | | |
| 6 | | 2 | 2 | | 2 | | | | |
| 7 | 2 | 2 | | 2 | 2 | | | | |
| 8 | 2 | | | 2 | | | | | |
| 9 | | | 2 | 2 | | | | | |
| 10 | | | 2 | | | | | | |
| 11 | | 2 | 2 | | | | | | |
| 12 | | 2 | | | | | | | |
| 13 | 2 | 2 | | | | | | | |

*Рис. 6*

Число два в черных клетках показывает, что начавший игру в этих позициях проиграет за два хода.

На этом можно остановиться, потому что по условию задачи нас интересуют только позиции с выигрышем в один или два хода (позиции в остальных клетках требуют для выигрыша больше двух ходов). Смотрим на верхнюю строку: позиции при $S = 21$ и $S = 23$ ведут к проигрышу за два хода, это и есть два возможных ответа на Задание 3.

Таким образом, для $S = 21$ и $S = 23$ у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, но у него нет стратегии, которая позволит ему гарантированно выиграть первым ходом. Все возможные ходы Пети и выигрышные ответы Вани при $S = 23$ приведены в таблице:

| Начало | Петя | Ваня | Петя | Ваня |
|---|---|---|---|---|
| (5,23) | (6,23) | (6,24) | (7,24) | (7,48) |
| | | | (6,25) | (6,50) |
| | (5,24) | | (12,24) | (12,48) |
| | | | (6,48) | (6,96) |
| | (10,23) | (10,46) | | |
| | (5,46) | | | |

*Рис. 7*

Вместо таблицы можно построить аналогичное дерево (конечно, это не совсем дерево, но для упрощения схемы можно объединить две ветки к узлу (6, 24), а также две ветки к узлу (10, 46)):

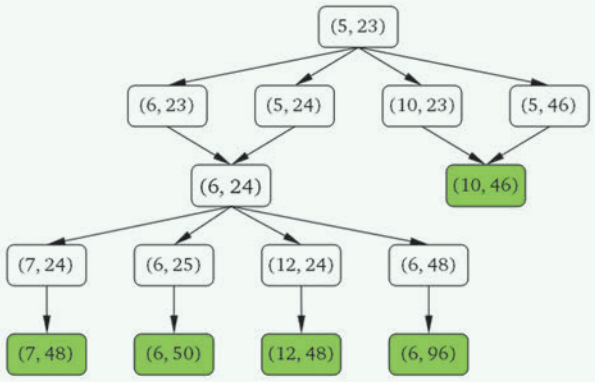



*Рис. 8*

Возможен и другой ответ на этот вопрос. При $S = 21$ у Вани тоже есть выигрышная стратегия, он выигрывает на первом или на втором ходу:

| Начало | Петя | Ваня | Петя | Ваня |
|---|---|---|---|---|
| (5,21) | (10,21) | (10,22) | (11,22) | (11,44) |
| | | | (10,23) | (10,46) |
| | (5,22) | | (20,22) | (20,44) |
| | | | (10,44) | (10,88) |
| | (6,21) | (12,21) | (13,21) | (13,42) |
| | | | (12,22) | (12,44) |
| | | | (24,21) | (24,42) |
| | | | (12,42) | (12,84) |
| | (5,42) | (5,84) | | |

*Рис. 9*

## Литература


1. Демоверсия, спецификация, кодификатор ЕГЭ 2015. Информатика и ИКТ [Электронный ресурс] URL: *http://fipi.ru/sites/default/files/document/1409834615/inf11_2015.zip* (дата обращения 01.08.2015).
2. Демоверсия, спецификация, кодификатор ЕГЭ 2011. Информатика и ИКТ [Электронный ресурс] URL: *http://www.fipi.ru/sites/default/files/document/1408710028/inf_10_11_10.zip* (дата обращения 01.08.2015).
3. Демоверсия, спецификация, кодификатор ЕГЭ 2008. Информатика и ИКТ [Электронный ресурс] URL: *http://fipi.ru/sites/default/files/document/1408710129/inform.rar* (дата обращения 01.08.2015).
4. Демоверсия, спецификация, кодификатор ЕГЭ 2010. Информатика и ИКТ [Электронный ресурс] URL: *http://fipi.ru/sites/default/files/document/1408710098/inf.zip* (дата обращения 01.08.2015).
5. Демоверсия, спецификация, кодификатор ЕГЭ 2009. Информатика и ИКТ [Электронный ресурс] URL: *http://fipi.ru/sites/default/files/document/1408710129/infZIP%20-%20WinRAR.zip* (дата обращения 01.08.2015).
6. *Поляков К.Ю.* ЕГЭ: новые стратегии (задача С3) // Информатика, № 1, 2013, с. 22–27.
7. КИМ ЕГЭ 2015 по информатике (досрочный период) [Электронный ресурс] URL: *http://www.fipi.ru/sites/default/files/document/2015/05.pdf* (дата обращения 01.08.2015).
8. *Поляков К.Ю.* Подготовка к ЕГЭ по информатике [Электронный ресурс] URL: *http://kpolyakov.spb.ru/school/ege.htm* (дата обращения 01.08.2015).

# Задание нового типа в ЕГЭ по информатике

**Д.М. Златопольский,**
Москва

Начиная с 2014 года в демонстрационных вариантах ЕГЭ по информатике и ИКТ вместо задания С1, связанного с принадлежностью точки с заданными координатами некоторой области на плоскости (см., например, [1]), приводится задание другого типа.

В варианте 2014 года [2] это задание обозначено так же, как и в демонстрационных вариантах экзамена предыдущего периода, — С1.

*Условие*

Требовалось написать программу, при выполнении которой с клавиатуры считывается натуральное число, не превосходящее $10^9$, и выводится максимальная цифра этого числа. Программист поторопился и написал программу неправильно. (Ниже для вашего удобства программа представлена на четырех языках программирования.)

**Бейсик**

```
DIM N AS LONG
INPUT N
max_digit = 9
WHILE N >= 10
  digit = N mod 10
  IF digit > max_digit THEN
    max_digit = digit
  END IF
  N = N \ 10
WEND
PRINT max_digit
END
```

**Паскаль**

```
Var N: longint;
  digit, max_digit: integer;
begin
  readln(N);
  max_digit := 9;
  while N >= 10 do
    begin
      digit := N mod 10;
      if digit > max_digit THEN
        max_digit := digit;
```

*На фото: Программист поторопился.*

```
      N := N div 10
    end;
  writeln(max_digit);
end.
```

**Си**

```
#include <stdio.h>
int main()
{
  long int N;
  int digit, max_digit
  scanf("%ld", &N);
  max_digit = 9;
  while (N >= 10)
  {
    digit = N % 10;
    if (digit > max_digit)
      max_digit = digit;
    N = N / 10;
  }
  printf("%d", max_digit);
}
```

**Алгоритмический язык**[1]

```
алг
нач
  цел N, digit, max_digit
  ввод N
  max_digit := 9
  нц пока N >= 10
    digit := mod(N, 10)
    если digit > max_digit то
      max_digit := digit
    все
    N := div(N, 10)
  кц
  вывод max_digit
кон
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 423.

2. Найдите все ошибки в этой программе (их может быть одна или несколько). Для каждой ошибки:

1) выпишите строку, в которой сделана ошибка;

2) укажите, как исправить программу, — приведите правильный вариант строки.

Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

*Анализ программ*

Используемые величины:

— *N* — заданное натуральное число, которое в ходе работы программы будет изменяться;

— *digit* — его отдельная цифра;

— *max_digit* — максимальное значение цифры.

Основные этапы алгоритма:

1) ввод значения числа *N*;

2) задание начального значения величины *max_digit*;

3) цикл действий:

— выделение последней цифры числа *N*;

— ее сравнение со значением *max_digit* и при необходимости изменение последнего;

— “отбрасывание” последней цифры числа *N*;

4) вывод ответа (значения *max_digit*).

*Решение заданий*

1. Трассировка программы (на бумаге — см. таблицу ниже — или “в уме”) показывает, что будет выведено число 9, что является неверным результатом.

| | **Начальное значение** | **Меняющиеся значения** | | |
|---|---|---|---|---|
| *N* | | 423 | 42 | 4 |
| *N* >= 10 | | Да | Да | Нет |
| *digit* | | 3 | 2 | |
| *digit* > *max_digit* | | Нет | Нет | |
| *N* | | 42 | 4 | |
| *max_digit* | 9 | 9 | 9 | 9 |

2. В программе имеются две ошибки.

2.1. Первая ошибка — неверное задание начального значения (инициализация) искомой максимальной цифры *max_digit*.

1) строка с ошибкой (здесь и далее — применительно к программе на школьном алгоритмическом языке):

```
max_digit := 9
```

В результате в программе значение *max_digit* в любом случае будет равно 9;

2) возможные варианты исправления ошибки:

— `max_digit := 0`

— `max_digit := -1`

— `max_digit := -2`

и т.д.

2.2. Вторая ошибка — неверное условие продолжения цикла (`N >= 10`).

1) строка с ошибкой:

```
нц пока N >= 10
```

В результате в программе не будет выделена и рассмотрена первая цифра числа *N*;

2) возможные варианты исправления ошибки:

```
нц пока N > 0
```

или

```
нц пока N >= 1
```

В демонстрационном варианте ЕГЭ 2015 года [3] соответствующее задание обозначено как задание 24 в части 2 (часть 3 в [3] отсутствует).

*Условие*

На обработку поступает последовательность из четырех неотрицательных чисел (некоторые числа могут быть одинаковыми). Нужно написать программу, которая выводит на экран коли-

[1] В системе программирования КуМир, использующей школьный алгоритмический язык, подчеркивание ключевых слов не проводится.

чество нечетных чисел и максимальное нечетное число. Если нечетных чисел нет, требуется вывести на экран “NO”. Известно, что вводимые числа не превышают 1000. Программист написал программу неправильно. Ниже эта программа для вашего удобства приведена на пяти языках программирования.

**Бейсик**

```
CONST n = 4
count = 0
maximum = 999
FOR i = 1 TO n
  INPUT x
  IF x mod 2 <> 0 THEN
    count = count + 1
    IF x > maximum THEN
      maximum = i
    END IF
  END IF
NEXT i
IF count > 0 THEN
  PRINT count
  PRINT maximum
ELSE
  PRINT "NO"
END IF
```

**Паскаль**

```
const n = 4;
var i, x: integer;
    maximum, count: integer;
begin
  count := 0;
  maximum := 999;
  for i = 1 to n do
    begin
      readln(x);
      if x mod 2 <> 0 then
        begin
          count := count + 1;
          if x > maximum then
            maximum := i;
        end;
    end;
  if count > 0 then
    begin
      writeln(count);
      write(maximum);
    end
  else
    writeln('NO');
end.
```

**Си**

```
#include <stdio.h>
int main(void)
{
  const int n = 4;
  int i, x, maximum, count;
  count = 0;
  maximum = 999;
  for (i = 1; i <= n; i++) {
    scanf("%d",&x);
    if (x % 2 != 0 {
      count++;
       if (x > maximum)
          maximum = i;
    }
  }
  if (count > 0) {
    printf("%d\n", count);
    printf("%d\n", maximum);
  }
  else
    printf("NO\n");
```

**Алгоритмический язык**

```
алг
нач
  цел n = 4[2]
  цел i, x
  цел maximum, count
  count := 0
  maximum := 999
  нц для i от 1 до n
    ввод x
    если mod(x, 2) <> 0 то
      count := count + 1
      если x > maximum то
        maximum := i
      все
    все
  кц
  если count > 0 то
    вывод count, нс
    вывод maximum
  иначе
    вывод "NO"
  все
кон
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе последовательности:

2 9 4 3

2. Приведите пример такой последовательности, содержащей хотя бы одно нечетное число, что, несмотря на ошибки, программа напечатает правильный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:

[2] В системе программирования КуМир одновременное описание константы и задание ее значения не допускается.

1) выпишите строку, в которой сделана ошибка;

2) укажите, как исправить программу, т.е. приведите правильный вариант строки.

Достаточно указать ошибки и способ их исправления для одного языка программирования.

Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

*Анализ программ*

Основные используемые величины:

— *n* (константа, равная 4) — количество чисел обрабатываемой последовательности;

— *x* — очередное число последовательности;

— *maximum* — максимальное нечетное число последовательности;

— *count* — количество нечетных чисел последовательности.

Основные этапы алгоритма:

1) задание начального значения величин *count* и *maximum*;

2) цикл действий для каждого числа *x*:

— ввод значения *x*;

— проверка значения *x* на "нечетность"; если *x* — нечетное число, то:

— увеличение количества *count* на 1;

— сравнение *x* со значением *maximum* и при необходимости изменение последнего;

3) вывод ответа (два варианта в зависимости от значения *count*).

*Решение заданий*

1. Трассировка программы показывает, что будут выведены числа 2 и 999 (второй результат является неверным).

| | **Начальные значения** | **Меняющиеся значения** | | | |
|---|---|---|---|---|---|
| *x* | | 2 | 9 | 4 | 3 |
| `mod(x, 2) <> 0` | | Нет | Да | Нет | Да |
| *count* | 0 | 0 | 1 | 1 | 2 |
| `x > maximum` | | | Нет | | Нет |
| *maximum* | 999 | | 999 | 999 | 999 |

2. Пример последовательности, для которой программа работает правильно, определим и приведем после анализа ошибок (так же целесообразно поступить и на экзамене).

3. В программе имеются две ошибки.

3.1. Первая ошибка — неверное задание начального значения величины *maximum*.

1) строка с ошибкой:

```
maximum := 999
```

В результате в программе значение *maximum* в любом случае будет равно 999;

2) возможные варианты исправления ошибки:

```
maximum := 1
maximum := 0
maximum := -1
maximum := -2
```

и т.д.

3.2. Вторая ошибка — неверное присваивание значения величине *maximum* в случае ее изменения (при *x* > *maximum*).

1) строка с ошибкой:

```
maximum := i
```

(присваивается не очередное значение числа *x*, а его порядковый номер);

2) правильный вариант:

```
maximum := x
```

Учитывая, что приведенная ошибочная программа в любом случае выводит значение *maximum*, равное 999, а количество нечетных чисел *count* определяет верно, последовательность чисел, для которой программа работает правильно, должна содержать число 999.

Примеры:

3 999 4 1

999 2 8 999

и т.п.

Рассмотрим также задание, аналогичное[3] заданию 24 из [4].

*Условие*

На обработку поступает натуральное число, не превосходящее $10^9$. Нужно написать программу, которая выводит минимальную нечетную цифру этого числа. Если в числе нет нечетных цифр, требуется вывести на экран "NO". Программист написал программу неправильно. Ниже эта программа для вашего удобства приведена на пяти языках программирования.

*Напоминание:* Ноль делится на любое натуральное число.

**Бейсик**

```
DIM N, DIGIT, MINDIGIT AS LONG
INPUT N
MINDIGIT = N MOD 10
WHILE N > 0
  DIGIT = N MOD 10
  IF DIGIT MOD 2 = 1 THEN
    IF DIGIT < MINDIGIT THEN
      MINDIGIT = DIGIT
    END IF
  END IF
  N = N \ 10
WEND
IF MINDIGIT = 9 THEN
  PRINT "NO"
ELSE
  PRINT MINDIGIT
END IF
```

**Python**

```
N = int(input())
minDigit = N % 10
while N > 0:
  digit = N % 10
  if digit % 2 == 1:
```

[3] Разработчики ЕГЭ запрещают копирование материалов экзамена.

```
    if digit < minDigit:
      minDigit = digit
    N = N // 10
if minDigit == 9:
  print("NO")
else:
  print(minDigit)
```

**Алгоритмический язык**

```
алг
нач
  цел N, digit, minDigit
  ввод N
  minDigit := mod(N, 10)
  нц пока N > 0
    digit := mod(N,10)
    если mod(digit, 2) = 1 то
      если digit < minDigit то
        minDigit := digit
      все
    все
    N := div(N,10)
  кц
  если minDigit = 9 то
    вывод "NO"
  иначе
    вывод minDigit
  все
кон
```

**Паскаль**

```
var N,digit, minDigit: longint;
begin
  readln(N);
  minDigit := N mod 10;
  while N > 0 do
    begin
      digit := N mod 10;
      if digit mod 2 = 1 then
        if digit < minDigit then
          minDigit := digit;
      N := N div 10;
    end;
  if minDigit = 9 then
    writeln('NO')
  else
    writeln(minDigit)
end.
```

**Си**

```
#include <stdio.h>
int main()
{
  int N, digit, minDigit;
  scanf("%d", &N);
  minDigit = N % 10;
  while (N > 0)
  {
    digit = N % 10;
    if (digit % 2 == 1)
      if (digit < minDigit)
        minDigit = digit;
    N = N / 10;
  }
  if (minDigit == 9)
    printf("NO");
  else
    printf("%d",minDigit);
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 732.

2. Приведите пример такого числа, при вводе которого программа выдает верный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:

1) выпишите строку, в которой сделана ошибка;

2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Достаточно указать ошибки и способ их исправления для одного языка программирования. Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

*Анализ программ*

Используемые величины:

— *N* — заданное натуральное число, которое в ходе работы программы будет изменяться;

— *digit* — его отдельная цифра;

— *minDigit* — минимальное значение цифры.

Основные этапы алгоритма:

1) ввод значения числа *N*;

2) расчет начального значения величины *minDigit*;

3) цикл действий:

— выделение последней цифры числа *N*;

— проверка этой цифры на нечетность; в случае нечетности — ее сравнение со значением *minDigit* и при необходимости изменение последнего;

— “отбрасывание” последней цифры числа *N*;

4) вывод ответа (значения *minDigit*).

*Решение заданий*

1. Трассировка программы:

| | Начальное значение | Меняющиеся значения | | |
|---|---|---|---|---|
| *N* | 732 | 732 | 73 | 7 |
| *minDigit* | 2 | 2 | 2 | 2 |
| *digit* | | 2 | 3 | 7 |
| *digit* — нечетная цифра | | Нет | Да | Да |
| *digit* < *minDigit* | | | Нет | Нет |
| *N* | | 14 | 4 | 0 |

— показывает, что программа выведет число 2, которое нечетной цифрой не является.

2. В программе имеются две ошибки.

2.1. Первая ошибка — неверное задание начального значения искомой минимальной цифры *minDigit*.

1) строка с ошибкой:

```
minDigit := mod(N, 10)
```

В результате в программе значение *minDigit* становится равно последней цифре числа *N*, которая в заданном случае не является нечетной;

2) возможные варианты исправления ошибки:

```
max_digit := 10
max_digit := 11
max_digit := 12
```

и т.д.

2.2. Вторая ошибка — неверное условие вывода ответа “NO” (*maxDigit* = 9).

1) строка с ошибкой:

```
если maxDigit = 9 то
```

В результате в программе не будет учтена возможно имеющаяся в числе *N* цифра 9;

2) возможные варианты исправления ошибки (соответствуют вариантам исправления первой ошибки):

```
если maxDigit = 10 то
если maxDigit = 11 то
```

и т.д.

**Примечание**. Здесь возникает вопрос для обсуждения. Можно ли в данном случае считать, что каждая ошибка может быть исправлена без изменения других строк? (Варианты исправления ошибок связаны.) Аналогичная коллизия возникает и при выполнении задания из [4], в котором также имеется соответствующее указание.

При определении примеров чисел *N*, для которых приведенная ошибочная программа работает правильно, следует учитывать указанные выше ошибки:

— последняя цифра числа должна быть нечетной; примеры:

4437
2469

и т.п.;

— последняя цифра числа может быть четной, но должны быть также нечетные цифры, меньшие последней:

8134
5666

и т.п.

Анализ приведенных заданий позволяет выявить ряд типовых ошибок, которые приводятся в программах[4].

1. Неправильная инициализация (задание начального значения) искомой максимальной или минимальной величины (числа заданной последовательности или цифры заданного числа).

2. Ошибочное оформление оператора цикла с условием, использованного для обработки цифр заданного числа *N*, в частности, условия продолжения цикла.

3. Неправильное присваивание значения искомой величине в ходе работы оператора цикла, использованного для ввода и обработки последовательности чисел, — присваивается порядковый номер числа вместо его значения.

В связи с этим можно сформулировать ряд указаний по разработке правильных вариантов соответствующих строк программ.

1. Начальное значение искомого максимального значения величины должно быть не больше нижней границы диапазона значений данной величины. Например, для цифр это начальное значение может быть равно нулю.

При поиске минимума его начальное значение должно быть не меньше верхней границы диапазона значений данной величины. Например, для цифр это начальное значение может быть равно 9, для четырехзначных чисел — 9999.

Если в задании идет речь о числах заданной последовательности или о цифрах заданного числа, обладающих некоторыми свойствами (четных или т.п.), то при указании начального значения максимума/минимума следует применять “строгое неравенство” для диапазона, упоминавшегося чуть выше. Например, при поиске минимальной четной цифры начальное значение должно быть равно 9, 10 (конечно — условно), 11 или т.п., при поиске максимальной четной цифры — –1, –2 или т.п.

При выводе ответа условие для проверки того факта, что соответствующие цифры или числа отсутствуют, должно учитывать это начальное значение.

2. Когда для обработки цифр заданного числа *N* используется оператор цикла с предусловием, условие продолжения цикла должно быть следующим $N > 0$ или $N >= 1$:

1) на школьном алгоритмическом языке:

```
нц пока N > 0
...
```

2) на языке Паскаль:

```
while N > 0 do …
```

и т.п.

В случае применения оператора цикла с постусловием условие окончания цикла должно быть таким — $N = 0$:

1) на школьном алгоритмическом языке:

```
нц
...
кц_при N = 0
```

2) на языке Паскаль:

```
repeat
...
until N = 0;
```

и т.п.

3. В программах, определяющих число из некоторой последовательности или из некоторого массива, при нахождении искомой величины (при соблюдении заданного условия) в правой части оператора присваивания должно указываться значение очередного числа или элемента массива:

[4] В будущем, конечно, возможны и другие ошибки.

1) на школьном алгоритмическом языке:

```
если …
  то
    x := a
…
```

или

```
если …
  то
    x := a[i]
…
```

— где $x$ — искомая величина, $a$ — очередное число, $a[i]$ — очередной элемент массива;

2) на языке Паскаль:

```
if …
  then x := a
…
```

или

```
if …
  then x := a[i]
…
```

и т.п.

## Задачи для самостоятельной работы учащихся

**1.** На обработку поступает последовательность из шести двузначных положительных чисел (некоторые числа могут быть одинаковыми). Нужно написать программу, которая выводит на экран минимальное число последовательности. Программист написал программу неправильно. Ниже эта программа приведена на четырех языках программирования.

**Бейсик**

```
CONST n = 6
DIM mini, x AS INTEGER
mini = 11
FOR i = 1 TO n
  INPUT x
  IF x < mini THEN
    mini = i
  END IF
NEXT i
PRINT mini
```

**Паскаль**

```
const n = 6;
var mini, x, i: integer;
begin
  mini := 11;
  for i = 1 to n do
    begin
      readln(x);
      if x < mini then
        mini := i;
    end;
  write(mini);
end.
```

**Си**

```
#include <stdio.h>
int main(void)
{
  const int n = 6;
  int i, x, mini;
  mini = 11;
  for (i = 1; i <= n; i++) {
    scanf("%d",&x);
    if (x < mini)
      mini = i;
  }
  printf("%d\n", mini);
}
```

**Алгоритмический язык**[5]

```
алг
нач
  цел n, mini, x, i
  n := 6
  mini := 11
  нц для i от 1 до n
    ввод x
    если x < mini то
        mini := i
    все
  все
  кц
  вывод mini
кон
```

Последовательно выполните следующее:

1. Напишите, что выведет эта программа при вводе чисел 49, 11, 33, 81, 10, 25.

2. Приведите пример такой последовательности, при вводе которой программа выдает верный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Для каждой ошибки:

1) выпишите строку, в которой сделана ошибка;

2) укажите, как исправить программу, — приведите правильный вариант строки.

Достаточно указать ошибки и способ их исправления для одного языка программирования.

Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

**Примечание.** Задание 3 и последний комментарий относятся также к другим задачам для самостоятельной работы.

**2.** Требовалось написать программу, при выполнении которой с клавиатуры считывается натуральное число, не превосходящее $10^7$, и выводится минимальная цифра этого числа. Программист поторопился и написал программу неправильно.

[5] Мы приводим оформление программ на алгоритмическом языке в виде, аналогичном представленному в [2–4].

Ниже эта программа представлена на четырех языках программирования.

**Бейсик**

```
DIM A AS LONG
DIM dig, min_dig AS INTEGER
INPUT A
min_dig = 0
WHILE A >= 10
  dig = A mod 10
  IF dig < min_dig THEN
    min_dig = dig
  END IF
  A = A \ 10
WEND
PRINT min_dig
END
```

**Паскаль**

```
Var A: longint;
  dig, min_dig; integer;
begin
  readln(A);
  min_dig := 0;
  while A >= 10 do
    begin
      dig := A mod 10;
      if dig < min_dig then
        min_dig := dig;
      A := A div 10
    End;
  writeln(min_dig);
end.
```

**Си**

```
#include <stdio.h>
int main()
{
  long int A;
  int dig, min_dig;
  scanf("%ld", &A);
  min_dig = 0;
  while (A >= 10)
  {
    dig = A % 10;
    if (dig < min_dig)
      min_dig = dig;
    A = A / 10;
  }
  printf("%d", min_dig);
}
```

**Алгоритмический язык**

```
алг
нач
  цел A, dig, min_dig
  ввод A
  min_dig := 0
  нц пока A >= 10
    dig := mod(A, 10)
    если dig < min_dig то
      min_dig := dig
    все
    A := div(A, 10)
  кц
  вывод min_dig
кон
```

Последовательно выполните следующее:

1. Напишите, что выведет эта программа при вводе числа 3482.

2. Приведите пример такого числа, при вводе которого программа выдает верный ответ.

3. См. выше.

**3.** На обработку поступает последовательность из пяти неотрицательных чисел (некоторые числа могут быть одинаковыми). Нужно написать программу, которая выводит на экран минимальное четное число и количество четных чисел. Если четных чисел нет, требуется вывести на экран “NO”. Известно, что вводимые числа не превышают 100. Программист написал программу неправильно. Ниже эта программа приведена на четырех языках программирования.

**Бейсик**

```
CONST n = 5
DIM x, mini, k, i AS INTEGER
mini = 0
k = 0
FOR i = 1 TO n
  INPUT x
  IF x mod 2 = 0 THEN
    k = k + 1
    IF x < mini THEN
      mini = x
    END IF
  END IF
NEXT i
IF mini = 0 THEN
  PRINT "NO"
ELSE
  PRINT mini
  PRINT k
END IF
```

**Паскаль**

```
const n = 5;
var x, mini, k, i: integer;
begin
  mini := 0;
  k := 0;
  for i := 1 to n do
    begin
      readln(x);
      if x mod 2 = 0 then
        begin
          k := k + 1;
          if x < mini then
            mini := x;
```

```
        end;
    end;
  if  mini = 0 then
    write('NO')
  else
    begin
      writeln(mini);
      write(k);
    end;
end.
```

**Си**

```
#include <stdio.h>
int main(void)
{
  const int n = 5;
  int k, x, mini;
  mini = 0;
  k = 0;
  for (i = 1; i <= n; i++) {
    scanf("%ld", &x);
    if (x % 2 = 0) {
      k++;
      if (x < mini)
        mini = x;
    }
  }
  if (k > 0) {
    printf("%d\n", k);
    printf("%d\n", mini);
  {
  else
    printf("NO\n");
}
```

**Алгоритмический язык**

```
алг
нач
  цел n, x, mini, k, i
  n := 5
  mini := 0
  k := 0
  нц для i от 1 до n
    ввод x
    если mod(x, 2) = 0 то
      k := k + 1
      если x < mini то
        mini := x
      все
    все
  кц
  если mini = 0 то
    вывод "NO"
  иначе
    вывод mini, нс
    вывод k
  все
кон
```

Последовательно выполните следующее:

1. Напишите, что выведет эта программа при вводе чисел 5 0 23 4 8.

2. Приведите пример такой последовательности, при вводе которой программа выдает верный ответ.

3. См. выше задачу 1.

*Напоминание:* Ноль делится на любое натуральное число.

**4.** На обработку поступает натуральное число, не превосходящее $10^6$. Нужно написать программу, которая выводит максимальную четную цифру этого числа. Если в числе нет четных цифр, требуется вывести на экран “NO”. Программист написал программу неправильно. Ниже эта программа приведена на четырех языках программирования.

**Бейсик**

```
DIM n, digit, maxeven AS LONG
INPUT n
maxeven = n MOD 10
WHILE n > 0
  digit = n MOD 10
  IF digit MOD 2 = 0 THEN
    IF digit > maxeven THEN
      maxeven = digit
    END IF
  END IF
  n = n \ 10
WEND
IF maxeven = 0 THEN
  PRINT "NO"
ELSE
  PRINT maxeven
END IF
```

**Паскаль**

```
var N, digit, maxeven: longint;
begin
  readln(N);
  maxeven  := N mod 10;
  while N > 0 do
    begin
      digit := N mod 10;
      if digit mod 2 = 0 then
        if digit > maxeven then
          MAXEVEN  := digit;
      N := N div 10;
    end;
  if maxeven = 0 then
    writeln('NO')
  else
    writeln(maxeven);
end.
```

**Си**

```
#include <stdio.h>
int main()
{
  int N, digit, maxeven;
```

```
  scanf("%d", &N);
  maxeven  = N % 10;
  while (N > 0)
  {
    digit = N % 10;
    if (digit % 2 == 0)
      if (digit > maxeven)
        maxeven = digit;
    N = N / 10;
  }
  if (maxeven == 0)
    printf("NO");
  else
    printf("%d",maxeven);
  return 0;
}
```

**Алгоритмический язык**

```
алг
нач
  цел N, digit, maxeven
  ввод N
  maxeven := mod(N, 10)
  нц пока N > 0
    digit := mod(N,10)
    если mod(digit, 2) = 0 то
      если digit > maxeven то
        maxeven  := digit
      все
    все
    N := div(N,10)
  кц
  если maxeven = 0 то
    вывод "NO"
  иначе
    вывод maxeven
  все
кон
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 647.

2. Приведите пример такого числа, при вводе которого программа выдает верный ответ.

3. См. выше.

*Напоминание:* Ноль делится на любое натуральное число.

## Ответы к задачам для самостоятельной работы

| Задача | Задание 1 | Задание 2 | Задание 3 |
|---|---|---|---|
| 1 | 5 | Любая последовательность, в которой есть число 11 и нет числа 10 | 1-я ошибка:<br>`mini := 11`[6]<br>Правильный вариант:<br>`mini := 99`<br>2-я ошибка:<br>`mini := i`<br>Правильный вариант:<br>`mini := x` |
| 2 | 0 (всегда) | Любое число, в котором есть цифра 0 | 1-я ошибка:<br>`min_dig := 0`<br>Правильный вариант:<br>`min_dig := 9`<br>или<br>`min_dig := 10`<br>или<br>`min_dig := 11`<br>и т.д.<br>2-я ошибка:<br>`нц пока A >= 10`<br>Правильный вариант:<br>`нц пока A > 0` |
| 3 | NO (всегда) | Любая последовательность, в которой нет четных чисел | 1-я ошибка:<br>`mini := 0`<br>Правильный вариант:<br>`mini := 100`<br>или<br>`mini := 101`<br>или<br>`mini := 102`<br>и т.п.<br>2-я ошибка:<br>`если mini = 0 то`<br>Правильный вариант:<br>`если count = 0 то` |

[6] Здесь и далее приводятся фрагменты программ на алгоритмическом языке.

| | | | |
|---|---|---|---|
| 4 | 7 | 1. Любое число с последней четной цифрой (если это цифра 0, то должны быть и другие четные цифры, большие 0):<br>32674<br>или<br>97650<br>и т.п.<br>2. Любое число с последней нечетной цифрой, в котором есть также четные цифры, большие последней, например:<br>4587 | 1-я ошибка:<br>`maxeven := mod(N, 10)`<br>Правильные варианты:<br>`maxeven := -1`<br>или<br>`maxeven := -2`<br>или<br>`maxeven := -3`<br>и т.д.<br>2-я ошибка:<br>`если maxeven = 0 то`<br>Правильные варианты (соответствуют вариантам исправления первой ошибки):<br>`если maxeven = -1 то`<br>или<br>`если maxeven = -2 то`<br>и т.д. |

## Литература


1. *Богомолова О.Б., Усенков Д.Ю.* Задачи на пересечение областей (С1): “на стыке алгебры и логики”. / Информатика, № 5/2011.

2. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена по информатике и ИКТ 2014 года. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*.

3. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена по информатике и ИКТ 2015 года. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*.

4. КИМ ЕГЭ 2015 (досрочный период). *http://www.fipi.ru/sites/default/files/document/2015/05.pdf*.


# Об изменении системы нумерации заданий ЕГЭ

Как известно, в 2015 году изменилась система нумерации заданий Единого государственного экзамена. Публикуем таблицу соответствия новых номеров заданий обозначениям заданий из ЕГЭ по информатике и ИКТ 2014 года.

В таблице перечислено 31 задание, а не 32. Это объясняется тем, что задание В2 экзамена 2014 года, связанное с определением значения переменной величины после выполнения заданного фрагмента программы, в котором массивы не используются, в экзамене 2015 года не представлено (в аналогичных заданиях 19/А12 массивы используются).

*Подготовил*
*Д.М. Златопольский,*
*Москва*

| 2015 год (27 заданий) | 2014 год (32 задания) |
|---|---|
| 1 | А9 |
| 2 | А3 |
| 3 | А4, А6 |
| 4 | А1 |
| 5 | А2 |
| 6 | А5, В1 |
| 7 | А7, В3 |
| 8 | В5 |
| 9 | А8, В10 |
| 10 | В4 |
| 11 | В6 |
| 12 | В11 |
| 13 | А11 |
| 14 | А13 |
| 15 | В9 |
| 16 | В7 |
| 17 | В12 |
| 18 | А10 |
| 19 | А12 |
| 20 | В8 |
| 21 | В14 |
| 22 | В13 |
| 23 | В15 |
| 24 | С1 |
| 25 | С2 |
| 26 | С3 |
| 27 | С4 |

издательство ДРОФА

# 2015/16 учебный год



Формат: ePUB 3.0

Поддерживает: Windows, Android, MacOS


# БЕСПЛАТНЫЙ ДОСТУП
## к электронным учебникам

Подробная информация об условиях бесплатного доступа к электронным учебникам в 2015/16 учебном году на сайте:

**efu.drofa.ru** (раздел «Акции»)


Электронные учебники издательства «ДРОФА» созданы в полном соответствии с требованиями приказа Минобрнауки России № 1559. Разнообразие методически обоснованных электронных образовательных ресурсов в сочетании с интуитивно понятным интерфейсом, удобной навигацией и встроенными возможностями автоматической адаптации к различным размерам экранов делает ЭФУ издательства «ДРОФА» уникальным образовательным продуктом, использование которого будет способствовать достижению лучших образовательных результатов.

# Вспоминаем прошлое, или Методика выполнения задания 27 из демонстрационного варианта ЕГЭ по информатике 2015 года

**Д.М. Златопольский,**
Москва

Прежде всего заметим, что в демонстрационном варианте контрольно-измерительных материалов единого государственного экзамена 2015 года по информатике и ИКТ [1], впервые за время проведения ЕГЭ, как задание 27 в части 2 обозначено самое сложное задание по программированию, которое до этого традиционно обозначалось как задача С4 в части 3 экзамена (в [1] часть 3 отсутствует).

*Условие*

"На спутнике "Фотон" установлен прибор, предназначенный для измерения энергии космических лучей. Каждую минуту прибор передает по каналу связи неотрицательное вещественное число — количество энергии, полученной за последнюю минуту, измеренное в условных единицах. Временем, в течение которого происходит передача, можно пренебречь.

Необходимо найти в заданной серии показаний прибора минимальное произведение двух показаний, между моментами передачи которых прошло не менее шести минут. Количество энергии, получаемой за минуту, не превышает 1000 условных единиц. Общее количество показаний прибора в серии не превышает 10 000.

*Вам предлагаются два задания, связанные с этой задачей: задание А и задание Б. Вы можете решать оба задания или одно из них по своему выбору.*

*Итоговая оценка выставляется как* ***максимальная*** *из оценок за задания А и Б. Если решение одного из заданий не представлено, то считается, что оценка за это задание составляет 0 баллов.*

*Задание Б является усложненным вариантом задания А, оно содержит дополнительные требования к программе.*

А. Напишите на любом языке программирования программу для решения поставленной задачи, в которой входные данные будут запоминаться в массиве, после чего будут проверены все возможные пары элементов. Перед программой укажите версию языка программирования.

ОБЯЗАТЕЛЬНО укажите, что программа является решением ЗАДАНИЯ А. Максимальная оценка за выполнение задания равна двум баллам.

Б. Напишите программу для решения поставленной задачи, которая будет эффективна как по времени, так и по памяти (или хотя бы по одной из этих характеристик).

Программа считается эффективной по времени, если время работы программы пропорционально количеству полученных показаний прибора $N$, т.е. при увеличении $N$ в $k$ раз время работы программы должно увеличиваться не более чем в $k$ раз.

Программа считается эффективной по памяти, если размер памяти, использованной в программе для хранения данных, не зависит от числа $N$ и не превышает 1 килобайта[1].

Перед программой укажите версию языка программирования и кратко опишите использованный алгоритм.

ОБЯЗАТЕЛЬНО укажите, что программа является решением ЗАДАНИЯ Б. Максимальная оценка за выполнение задания равна четырем баллам.

Максимальная оценка за правильную программу, эффективную по времени, но неэффективную по памяти, равна четырем баллам.

Входные данные представлены следующим образом. В первой строке задается число $N$ — общее количество показаний прибора. Гарантируется, что $N > 6$. В каждой из следующих $N$ строк задается одно неотрицательное вещественное число — очередное показание прибора.

*Пример входных данных:*

```
11
12
45.3
5.5
4
25
23
21
20
10
12
26
```

Программа должна вывести одно число — описанное в условии произведение.

*Пример выходных данных для приведенного выше примера входных данных:*

```
48”.
```

Начнем с решения задания А, как более простого[2].

Здесь метод решения задачи указан в условии применительно к заданию А — сохранить все показания прибора в массиве, после чего рассмотреть все возможные пары значений и сравнить их произведение с минимальным произведением среди уже рассмотренных пар. Правда, при этом следует уточнить, что пары должны образовывать только показания, между моментами передачи которых прошло не менее шести минут. Так, для примера, показанного на *рис.* 1, для показания 12,4 нужно рассматривать не все предыдущие значения, а только оттененные.

В программе, реализующей описанный метод решения, кроме величины $N$, используем следующие основные величины:

— *м_пр* — искомое значение;

— *п* — массив для хранения показаний прибора.

Программа на школьном алгоритмическом языке:

```
алг Задание_27А
нач цел N, вещ м_пр, цел i, j
  |Ввод количества показаний
  |прибора значений N
  ввод N
  |Описание массива значений п
  вещ таб п[1:N]
  |Ввод всех показаний и сохранение
  |их в массиве п
  нц для i от 1 до N
  ввод п[i]
  кц
  |Поиск ответа
```

| **Номер показания** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Очередное показание** | 14,4 | 3,2 | 23,0 | 17,2 | 5,8 | 2,2 | 67,0 | 44,3 | 11,2 | **12,4** | … |
| | | … | $i-7$ | $i-6$ | $i-5$ | $i-4$ | $i-3$ | $i-2$ | $i-1$ | $i$ | |

*Рис.* 1

[1] Заметим, что критерии эффективности программ по памяти и по времени в демонстрационных вариантах ЕГЭ в [1] указаны впервые за все время проведения экзамена. — **Прим. автора.**

[2] В [1] разбор решения начинается с самого сложного варианта решения задания Б.

```
  |Начальное значение минимального
  |произведения
  м_пр := 1000 * 1000 + 1
  |Перебор всех пар значений
  |Для каждого значения, начиная с 7-го
  нц для i от 7 до N
    |и для каждого допустимого
    |предшествующего значения
    нц для j от 1 до i - 6
      |сравниваем их произведение
      |со "старым" значением величины м_пр
      если п[i] * п[j] < м_пр
        то |Встретилась пара чисел
           |с минимальным произведением
           |Меняем значение м_пр
           м_пр := п[i] * п[j]
      все
    кц
  кц
  |Вывод ответа
  вывод нс, м_пр
кон
```

Обратим внимание, что описание массива в теле программы, после задания значения *N*, как это сделано в приведенном случае, допустимо не во всех языках программирования. Кроме того, в школьном алгоритмическом языке (система КуМир) для проверки и возможного изменения величины *м_пр* (и в большинстве случаев в других программах, приведенных в статье далее) можно использовать стандартную функцию *min*, возвращающую минимальное значение двух ее аргументов:

```
…
|сравниваем их произведение со "старым"
|значением величины м_пр
м_пр := min(п[i] * п[j], м_пр)
…
```

Заметим также, что в приведенной в [1] программе решения задания А на языке Паскаль (она обозначена как *Программа 4*):

1) вместо значения 6 (требуемое расстояние между двумя показаниями) используется константа *s*;

2) перебор всех возможных пар значений проводится следующим образом:

```
for i := 1 to N - s do
  begin
    for j := i + s to N - s do
      begin
        if a[i] * a[j] < m then
                        m := a[i] * a[j]
      end;
  end;
```

или на школьном алгоритмическом языке и при принятых именах величин:

```
нц для i от 1 до N - 6
  нц для j от i + 6 до N
    если п[i] * п[j] < м_пр
      то
        м_пр := п[i] * п[j]
    все
  кц
кц
```

что по сути аналогично варианту, приведенному выше[3].

**Примечание**. Использование в программе на Паскале двух составных операторов является излишним. Можно применить такой, более компактный, вариант фрагмента:

```
for i := 1 to N - s do
  for j := i + s to N - s do
    if a[i] * a[j] < m then
       m := a[i] * a[j]
```

Хотя задание А мы решили, обсудим одно из направлений совершенствования разработанной программы, которое будет полезно при решении задачи В. Оно заключается в следующем.

Разработанная выше программа является неэффективной по времени ее выполнения из-за большого числа сравнений (оно пропорционально $N^2$). Этот недостаток можно ликвидировать, если для каждого вводимого показания знать минимальное показание среди предыдущих, отстоящих от него не менее, чем на шесть значений. Эти минимальные показания можно хранить в специальном массиве, заполнять который можно по мере ввода показаний прибора[4]:

Соответствующая программа:

```
алг Задание_27_2
нач цел N, вещ м_пр, цел i
  ввод N
  |Описание массивов значений п
  вещ таб п[1:N]
  |и мин_п (минимальные значения
  |среди заданных)
  вещ таб мин_п[1:N]
  |Ввод первых шести показаний
  |1-е показание (и одновременно
  |значение 1-го элемента массива мин_п)
  ввод мин_п[1]
  |2..6 показания
  нц для i от 2 до 6
    ввод п[i]
    |Заполняем массив мин_п
    |Можно также использовать функцию min
    если п[i] < мин_п[i - 1]
      то
        мин_п[i] := п[i]
      иначе
        мин_п[i] := мин_п[i - 1]
    все
  кц
  |Поиск ответа
  м_пр := 1000 * 1000 + 1
```

[3] В программах, в которых массив со всеми показаниями прибора не используется (см. далее), приведенный в [1] способ формирования пар значений невозможен.

[4] Значения элементов этого массива будут образовывать невозрастающую последовательность (убедитесь в этом!).

```
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п[i]
     |Заполняем массив мин_п
    |(можно также использовать функцию min)
     если п[i] < мин_п[i - 1]
       то
         мин_п[i] := п[i]
       иначе
         мин_п[i] := мин_п[i - 1]
     все
     |Проверяем произведение очередного
     |показания на минимальное значение
     |"предыдущих"
     если п[i] * мин_п[i - 1] < м_пр
       то  |Встретилась пара чисел
           |с минимальным произведением
         м_пр := п[i] * мин_п[i - 1]
     все
  кц
  |Вывод ответа
  вывод нс, м_пр
кон
```

Приведенный вариант программы более эффективен по времени, чем программа Задание_27А, но менее эффективен по памяти, так как использует два массива. Последний недостаток можно устранить, если массив *мин_п* не использовать.

Обсудим вопрос — какое значение следует использовать “в паре” с очередным введенным значением $п[i]$ и сравнивать их произведение со “старым” значением искомой величины *м_пр*? Ответ на этот вопрос дает анализ *рис.* 1. Для очередного значения, равного 12,4, следует использовать минимальное из оттененных значений — 3,2 (их произведение равно 39,68). Значит, нужно иметь информацию о минимальном значении среди уже заданных показаний, причем не всех, а отстоящих от очередного введенного показания не менее чем на шесть значений. Величине, хранящей информацию о таком минимальном значении, в программе дадим имя *мин_п*.

Соответствующая программа:

```
алг Задание_27_3
нач цел N, вещ мин_п, м_пр, цел i
  ввод N
  |Описание массива значений п
  вещ таб п[1:N]
  |Ввод первых шести показаний
  нц для i от 1 до 6
    ввод п[i]
  кц
  |Поиск ответа
  мин_п := 1000
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п[i]
     |Проверяем значение мин_п
     |(обратите внимание на индекс элемента
     |массива п)
     если п[i - 6] < мин_п
       то
         мин_п := п[i - 6]
     все
     |Проверяем произведение очередного
     |показания
     |на минимальное значение "предыдущих"
     если п[i] * мин_п < м_пр
       то
         м_пр := п[i] * мин_п
     все
  кц
  вывод нс, м_пр
кон
```

Можно также не использовать массив показаний *п*, а сразу обрабатывать очередное введенное значение (“сохранив” при этом массив *мин_п*). Прежде чем представлять программу, заметим, что очередное показание имеет в ней имя *п*.

```
алг Задание_27_4
нач цел N, вещ п, м_пр, цел i
  ввод N
  |Описание массива мин_п
  |(минимальные значения среди заданных)
  вещ таб мин_п[1:N][5]
  |Ввод первых шести показаний
  |1-е показание (и одновременно
  |значение 1-го элемента массива мин_п)
  ввод мин_п[1]
  |2..6 показания
  нц для i от 2 до 6
    ввод п
     |Заполняем массив мин_п
     если п < мин_п[i - 1]
       то
         мин_п[i] := п
       иначе
         мин_п[i] := мин_п[i - 1]
     все
  кц
  |Поиск ответа
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п
     |Заполняем массив мин_п
     если п < мин_п[i - 1]
       то
         мин_п[i] := п
       иначе
         мин_п[i] := мин_п[i - 1]
     все
     |Сравниваем произведение
     |(можно также использовать функцию min)
```

[5] Использование массива *мин_п* размером $N - 6$ невозможно в случае, когда $12 > N > 6$.

```
    если п * мин_п[i - 6] < м_пр
      то
        м_пр := п * мин_п[i - 6]
    все
  кц
  вывод нс, м_пр
кон
```

После разработки программ `Задание_27_3` и `Задание_27_4` мы подошли к решению задания В.

Прежде всего здесь при выполнении задания следует обратить внимание на комментарий в условии, касающийся эффективности программы по памяти: “Программа считается эффективной по памяти, если размер памяти, использованной в программе для хранения данных, не зависит от числа *N* и не превышает 1 килобайта”.

Этот комментарий говорит о том, что использование массива для хранения *всех* показаний прибора в данном случае невозможно. Вариант программы без этого массива мы только что разработали. Осталось “отказаться” от массива *мин_п*. Как?

Выше уже отмечалось, что значения элементов этого массива образуют невозрастающую последовательность. Например, для следующих показаний прибора:

45, 53, 33, 46, 47, 15, 26, 15, 44, 14, 13, 37, 22, …

— массив *мин_п* будет иметь вид:

| 45 | 45 | 33 | 33 | 33 | 15 | 15 | 15 | 15 | 14 | 13 | 13 | 13 | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | … |

*Рис.* 1

Первый элемент этого массива будет “использован” седьмым показанием прибора (равным 26). И больше не понадобится! И вообще, какие из элементов этого массива надо знать в момент задания некоторого очередного показания прибора, например, 11-го (равного 13)? Ответ показан на *рис*. 2.

| 45 | 45 | 33 | 33 | 33 | 15 | 15 | 15 | 15 | 14 | **13** | 13 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* |

*Рис.* 2

А после ввода 11-го показания “ненужным” будет 5-й элемент (33), но станет “нужным” (для последующих расчетов) введенное значение 13. Следовательно, вместо массива размером *N* – 6 элементов достаточно использовать только массив из шести элементов, меняя его содержание по мере ввода новых показаний[6]. Для приведенного примера особенности изменения этого массива показаны на *рис*. 3.

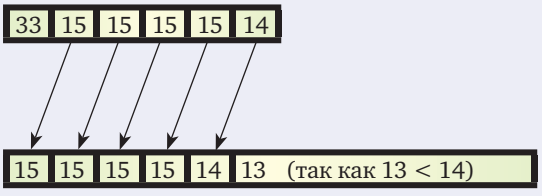



*Рис.* 3

[6] Можно вместо массива из шести элементов применить шесть “простых” величин, однако использование массива упрощает изменение значений по мере ввода новых показаний (см. программу далее).

Итак, идея решения — использовать массив из шести элементов, хранящий минимальные значения показаний среди отстоящих от текущего показания не менее чем на шесть значений, и менять его содержание по мере ввода новых показаний.

*Этапы соответствующего алгоритма*

1. Ввод значения *N*.
2. Ввод первых шести показаний и сохранение их в массиве.
3. Для каждого показания, начиная с 7-го:
   3.1. Ввод очередного показания.
   3.2. Расчет произведения этого показания на 1-й элемент указанного массива и сравнение произведения с минимальным произведением уже рассмотренных пар.
   3.3. Изменение массива минимальных значений.
4. Вывод ответа.

Соответствующая программа:

```
алг Задание_27Б
нач цел N, вещ п, м_пр, вещ таб мин_п[1:6],
цел i, j
  ввод N
  |Ввод первых шести показаний
  |1-е показание (и одновременно
  |значение 1-го элемента массива мин_п)
  ввод мин_п[1]
  |2..6 показания
  нц для i от 2 до 6
    ввод п
    |Заполняем массив мин_п
    если п < мин_п[i - 1]
      то
        мин_п[i] := п
      иначе
        мин_п[i] := мин_п[i - 1]
    все
  кц
  |Поиск ответа
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п
    |Сравниваем произведение
    |(обратите внимание на индекс
    |элемента массива)
    если п * мин_п[1] < м_пр
      то
        м_пр := п * мин_п[1]
    все
    |Меняем массив мин_п
    |Сдвигаем влево 2..6 элементы
    нц для j от 1 до 5
      мин_п[j] := мин_п[j + 1]
    кц
    |При необходимости (!) меняем и
    |6-й элемент
    если п < мин_п[6]
      то
        мин_п[6] := п
    все
  кц
  вывод нс, м_пр
кон
```

Можно также использовать массив не минимальных значений, а меняющийся массив из шести последних показаний. В приведенном ниже варианте программы имя этого массива — *п*2.

При этом понадобится также величина, хранящая минимальное введенное число с учетом требуемого “расстояния” между показаниями (имя величины — *мин_п*). Ее значение сравнивается с первым элементом массива *п*2.

Особенности изменения массива *п*2 и значения величины *мин_п* покажем на примере следующих показаний прибора:

45, 53, 33, 46, 47, 15, 26, 18, 44, 14, 13, 37, 22, …

Перед вводом показания 37 массив *п*2 будет иметь вид:

| 15 | 26 | 18 | 44 | 14 | 13 |
|---|---|---|---|---|---|

При этом *мин_п* = 33.

После ввода показания, равного 37, для обработки станет доступным первый элемент массива (15). Его сравнение со “старым” значением величины *мин_п* показывает, что новым минимумом станет число 15, которое и должно использоваться при расчете произведения. Новый вариант массива *п*2 для дальнейшей обработки показан на *рис*. 4.

| 26 | 18 | 44 | 14 | 13 | 37 |
|---|---|---|---|---|---|

*Рис.* 4

*Этапы соответствующего алгоритма*

1. Ввод значения *N*.

2. Ввод первых шести показаний и сохранение их в массиве.

3. Для каждого показания, начиная с 7-го:

3.1. Ввод очередного показания.

3.2. Уточнение (при необходимости) минимального показания *мин_п*.

3.3. Расчет произведения очередного показания на *мин_п* и сравнение произведения с минимальным произведением уже рассмотренных пар.

3.4. Изменение массива значений.

4. Вывод ответа.

Программа, работающая по последнему алгоритму, имеет вид:

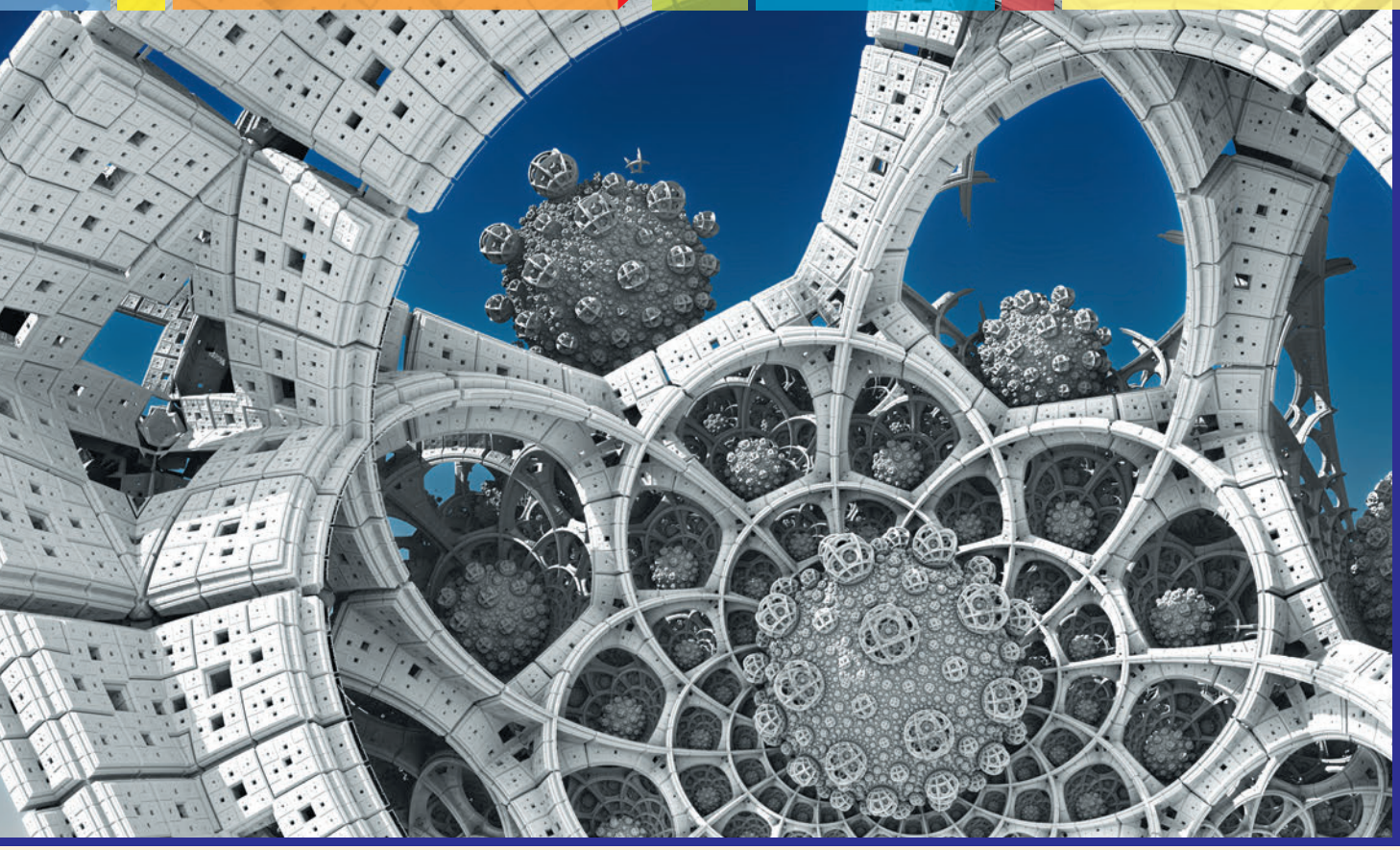
```
алг Задание_27Б_вариант_2
нач цел N, вещ п, мин_п, м_пр,
вещ таб п2[1:6], цел i, j
  ввод N
  м_пр := 1000 * 1000 + 1
  мин_п := 1000
  |Ввод первых шести показаний
  |(и одновременно заполнение массива п2)
  нц для i от 1 до 6
    ввод п2[i]
  кц
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п
    |Уточняем при необходимости
    |значение мин_п
    |(обратите внимание на индекс
    |элемента массива)
    если п2[1] < мин_п
      то
        мин_п := п2[1]
    все
    |Сравниваем произведение
    если п * мин_п < м_пр
      то
        м_пр := п * мин_п
    все
    |Меняем массив п2 (см. рис. 4)
    |Сдвигаем влево 2..6 элементы
    нц для j от 1 до 5
      п2[j] := п2[j + 1]
    кц
    |Записываем в массив очередное показание
    п2[6] := п
  кц
  вывод нс, м_пр
кон
```

В заключение заметим, что в [1] представлена программа решения задачи Б (обозначенная как *Программа 1*), аналогичная программе Задание_27Б, но отличающаяся тем, что изменение массива *мин_п* происходит не путем постоянного сдвига его элементов, а (цитата) “путем циклического заполнения массива”. Приводя соответствующую программу, оценить ее понятность предлагаем читателям.

```
алг
нач
 |требуемое расстояние между показаниями
  цел s = 6
  цел N
  ввод N
  вещ a  | очередное показание прибора
 |текущие минимумы последних s элементов
  вещтаб мини [0:s - 1]
  цел i
  |вводим первые s показаний,
  |фиксируем минимумы
  ввод мини[1]
  нц для i от 2 до s
    ввод a
    мини[mod(i, s)] := min(a, мини[i - 1]
  кц
  вещ м | минимальное значение произведения
  м = 1000 * 1000 + 1
  нц для i от s + 1 до N
    ввод a
    мини[mod(i, s)] :=
    min(a, мини[mod(i - 1, s)]
  кц
  вывод м
кон
```

## Литература

1. Демонстрационный вариант контрольно-измерительных материалов единого государственного экзамена 2015 года по информатике и ИКТ. Подготовлен Федеральным государственным бюджетным научным учреждением “Федеральный институт педагогических измерений”. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*.



# Задания на использование рекурсивных функций и процедур в ЕГЭ (задания 11)

**Д.М. Златопольский,**
Москва

*Начиная с 2013 года в демонстрационных вариантах ЕГЭ по информатике и ИКТ приводится задание на использование рекурсивных функций и процедур. В данной статье описывается методика выполнения нескольких вариантов таких заданий.*

Напомним, что “рекурсивными”[1] называют процедуры или функции, которые используют (вызывают) как вспомогательную саму себя.

В качестве примера приведем функцию для расчета факториала натурального числа $n$ (факториал числа $n$, обозначаемый $n!$, равен $1 \times 2 \times 3 \times \ldots \times n$). Такую функцию можно создать, имея в виду, что $n! = (n - 1)! \times n$. На школьном алгоритмическом языке соответствующая функция имеет вид:

```
алг цел Факториал(арг цел n)
нач
  |Рекурсивный вызов функции
  |Факториал
  знач := Факториал(n - 1) * n
  |Служебное слово знач означает
  |"значение функции"
кон
```

На самом деле эта функция оформлена не по правилам и при выполнении программы появится сообщение об ошибке. Дело в том, что при каждом вызове вспомогательной функции (или процедуры) для нее отводится место в оперативной памяти, которое “освобождается” после завершения ее (функции/процедуры) работы. Но если во вспомогательной процедуре имеется рекурсия, то вызовы вспомогательных подпрограмм будут продолжаться до тех пор, когда место в памяти будет исчерпано. Чтобы устранить этот недостаток, необходимо так

[1] От латинского *recursio* — возвращение.

оформлять функции (процедуры), чтобы рекурсивные вызовы осуществлялись по условию, которое когда-то станет ложным. Например, для приведенной функции это условие записывается так:

```
алг цел Факториал(арг цел n)
нач
  если n > 1
    то
      |Рекурсивный вызов функции Факториал
      знач := Факториал(n - 1) * n
    иначе |Известное значение
      знач := 1
  все
кон
```

или

```
алг цел Факториал(арг цел n)
нач
  если n > 1
    то
      знач := 1
    иначе
      знач := Факториал(n - 1) * n
  все
кон
```

Алгоритм вычисления значения функции $F(n)$, использующий рекурсию, может быть описан по-другому — с указанием двух или более вариантов значения, например, в виде:

$F(1) = 10$
$F(n) = F(n - 1) + 3$, при $n > 1$

или

$F(1) = 10$
$F(n) = F(n - 1) + 3$, при $n > 1$
$F(n) = 0$, при $n < 1$

или

$F(12) = 5$
$F(n) = F(n + 1) - 1$, при $n < 12$

или

$F(n) = 25$, при $n > 12$
$F(12) = 5$
$F(n) = F(n + 1) - 1$, при $n < 12$

и т.п.

Видно, что в двух первых случаях рекурсивные вызовы осуществляются с меньшими значениями аргумента, в остальных — с большими. Во всех случаях когда-то рекурсивные вызовы прекратятся (убедитесь в этом!) и начнется передача результатов в "вызывающую" функцию.

Пример работы программы при вычислении значения факториала числа 4 показан на *рис.* 1–2.


*Рис.* 1

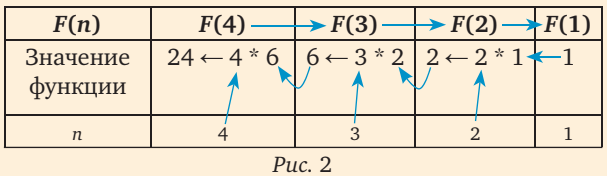



*Рис.* 2

*Ответ*: 24.

В [1] представлено задание, аналогичное рассмотренным выше.

*Условие*

Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 1$
$F(n) = F(n - 1) * n$, при $n > 1$

Чему равно значение функции $F(5)$?

В ответе запишите только натуральное число.

*Решение*

Анализ показывает, что в условии описан рекурсивный алгоритм вычисления значения факториала числа $n$ и требуется вычислить его значение при $n = 5$.

Используя *рис.* 1 или *рис.* 2, можно получить ответ, равный $24 \cdot 5 = 120$.

## Задания для самостоятельной работы

**1.** Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 1$
$F(n) = F(n - 1) + n$, при $n > 1$

Чему равно значение функции $F(6)$?

**2.** Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 2$
$F(n) = F(n - 1) + 2n$, при $n > 1$

Чему равно значение функции $F(5)$?

**3.** Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(n) = F(n - 1) + n - 2$, при $n > 1$
$F(1) = 3$

Чему равно значение функции $F(4)$?

Рассмотрим задание из [2].

*Условие*

Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 1$ при $n \le 2$;
$F(n) = F(n - 1) + 2 \times F(n - 2)$, при $n > 2$

Чему равно значение функции $F(7)$?

В ответе запишите только натуральное число.

*Решение*

Здесь в алгоритме происходят два рекурсивных вызова функции с разными параметрами. Анализ работы программы с помощью таблицы, как на *рис.* 2, в данном случае затруднен. Построим дерево рекурсивных вызовов функции. Это удобно делать так. Так как при первом вызове значение параметра на 1 меньше исходного, то сначала получим его (вызова) "ветвь" до значения, равного 2, для которого результат известен (*рис.* 3).


*Рис.* 3

Оформим также вторые вызовы из функций $F(7)$, $F(6)$, $F(5)$, $F(4)$ и $F(3)$ — см. *рис.* 4. При этом следует

учесть, что при вторых вызовах значение параметра на 1 меньше, чем при первом (“верхнем”).

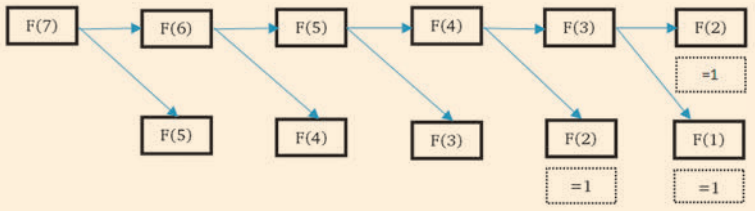



*Рис.* 4

Теперь можно определить значение оттененной функции $F(3)$:

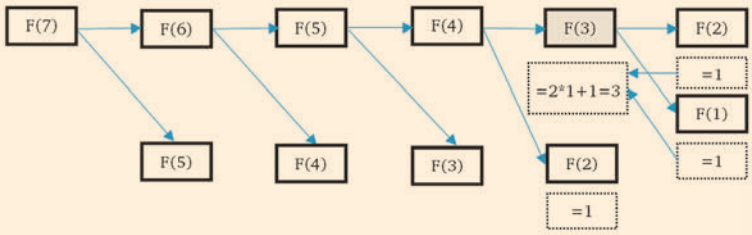



*Рис.* 5

Так как функция $F(3)$ на полученном дереве вызовов присутствует еще раз, то для нее значение отдельно можно не рассчитывать, а использовать только что найденное (см. *рис.* 6). Это даст возможность определить значение функции $F(4)$, которое можно использовать для другой такой же функции. Аналогично можно получить два значения функции $F(5)$. При этом удобно для расчетов записать условную формулу, соответствующую заданной в условии формуле $F(n-1) + 2 \times F(n-2)$, которая на *рис.* 6 оттенена.

$F$ = 2 * “нижнее” значение + “верхнее” значение

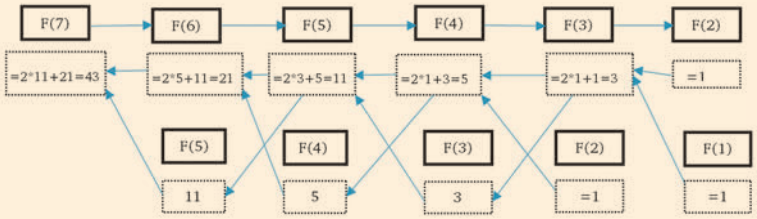



*Рис.* 6

Итак, построив только половину дерева, мы получили значения $F(6)$ и $F(5)$, требующиеся для определения искомого значения функции, которое также можно получить по “оттененной” формуле.

*Ответ*: 43.

Обсудим вопрос — в каких случаях можно не строить полное дерево вызовов, а только его часть, используя найденные значения для других “ветвей”.

Анализ показывает, что в алгоритме, в котором имеются два рекурсивных вызова функции $F(x)$:

$F(x-a)$
$F(x-b)$

или

$F(x+a)$
$F(x+b)$

— указанная “экономия” возможна, когда $b$ кратно $a$, например, при:

$F(x-1)$
$F(x-2)$

или

$F(x-1)$
$F(x-3)$

или

$F(x-2)$
$F(x-4)$

или

$F(x+1)$
$F(x+2)$

или

$F(x+1)$
$F(x+3)$

или

$F(x+2)$
$F(x+4)$

и т.п.

В других случаях “экономия” частичная. Но она может быть больше, когда в алгоритме имеются три рекурсивных вызова (такие примеры приводятся в книгах от разработчиков ЕГЭ, например, в [3]).

## Задания для самостоятельной работы

**4.** Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 1$ при $n \le 2$

$F(n) = F(n-1) + 2 \times F(n-3)$, при $n > 2$

Чему равно значение функции $F(8)$?

**5.** Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 1$ при $n \ge 7$

$F(n) = F(n+1) + 2 \times F(n+2)$, при $n < 7$

Чему равно значение функции $F(1)$?

**6.** Алгоритм вычисления значения функции $F(n)$, где $n$ — натуральное число, задан следующими соотношениями:

$F(1) = 1$ при $n \le 2$

$F(n) = F(n-1) + F(n-2) + F(n-3)$, при $n > 2$

Чему равно значение функции $F(7)$?

## Задание 11 из [4]

*Условие*

Ниже на пяти[2] языках программирования записан рекурсивный алгоритм $F$.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод n, нс
  если n < 5
    то
      F(n + 1)
      F(n + 3)
  все
кон
```

[2] В данной статье приводятся варианты алгоритмов на двух языках. В [4–5] представлены также алгоритмы на языках Си, Бейсик и Python.

**Паскаль**

```
procedure F(n: integer);
begin
  writeln(n);
  if n < 5 then
    begin
      F(n + 1);
      F(n + 3)
    end
end;
```

Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова $F(1)$?

*Решение*

Здесь впервые в демонстрационных вариантах экзамена в алгоритме использована не функция, а процедура (в языке программирования Си оба вида подпрограмм называются “функцией”).

Так как при каждом выполнении процедуры на экран выводится текущее значение параметра, которые и требуется просуммировать, то на дереве вызовов удобно указать только эти значения:

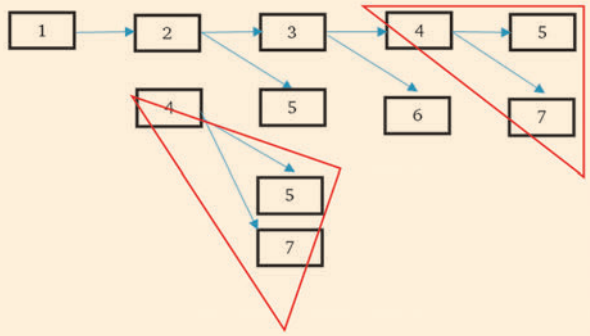



*Рис. 7*

При оформлении дерева и при расчетах следует учесть следующее:

1) вызовы процедур происходят только при значениях параметра, меньших 5;

2) при первых вызовах значение параметра меняется от 1 до 5 с шагом 1;

3) при вторых (“нижних”) вызовах значение меняется от 4 до 7 с шагом 1 (можно также иметь в виду, что эти значения на 2 больше, чем при соответствующем первом вызове);

4) в дереве есть два одинаковых фрагмента (на *рис.* 7 они ограничены красными треугольниками), сумма значений чисел в которых одна и та же (16).

*Ответ*: $16 \times 2 + 1 + 2 + 3 + 5 + 6 = 49$.

## Задания для самостоятельной работы

**7.** Ниже на двух языках программирования записан рекурсивный алгоритм $F$.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод n, нс
  если n > 1
    то
      F(n - 1)
      F(n - 2)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  writeln(n);
  if n > 1 then
    begin
      F(n - 1);
      F(n - 2)
    end
end;
```

Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова $F(5)$?

**8.** Ниже на двух языках программирования записан рекурсивный алгоритм $F$.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод n, нс
  если n < 6
    то
      F(n + 1)
      F(n + 2)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  writeln(n);
  if n < 6 then
    begin
      F(n + 1);
      F(n + 2)
    end
end;
```

Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова $F(2)$?

**9.** Ниже на двух языках программирования записан рекурсивный алгоритм $F$.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод n, нс
  если n > 1
    то
      F(n - 2)
      F(n - 3)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  writeln(n);
  if n > 1 then
    begin
      F(n - 2);
      F(n - 3)
    end
end;
```

Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова $F(7)$?

## Задание 11 из [5]

*Условие*

Ниже на пяти[3] языках программирования записаны две рекурсивные функции (процедуры): $F$ и $G$.

[3] См. выше сноску 2.

**Алгоритмический язык**

```
алг F(цел n)
нач
  если n > 0
    то
      G(n - 1)
  все
кон
алг G(цел n)
нач
  вывод "*"
  если n > 1
    то
      F(n - 2)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  if n > 0 then
    G(n - 1)
end;
procedure G(n: integer);
begin
  write('*');
  if n > 1 then
    F(n - 2)
end;
```

Сколько символов “звездочка” будет напечатано на экране при выполнении вызова *F*(11)?

*Решение*

Прежде всего заметим, что в данном случае процедура *F* вызывает себя как вспомогательную (то есть рекурсивно) не непосредственно, а через процедуру *G*. Такая рекурсия называется “косвенной”, или “непрямой”.

Построим схему вызовов функций с учетом ограничений (см. *рис*. 8).

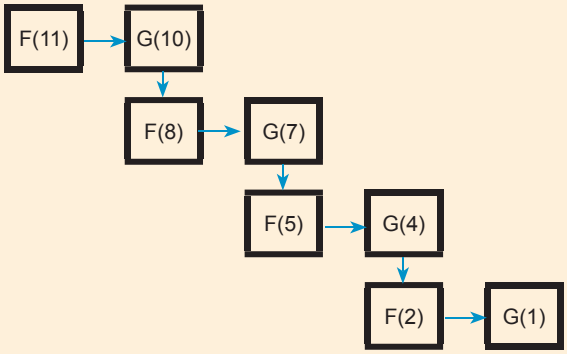



*Рис*. 8

Так как символы “звездочка” печатаются только при выполнении процедуры *G* (один символ при каждом вызове), то искомое количество равно числу вызовов указанной процедуры, то есть 4.

*Ответ*: 4.

В общем случае, если оформить процедуры следующим образом:

```
алг F(цел n)
нач
  если n > C       |или n < C
    то
      G(n - a)     |или G(n + a)
  все
кон
алг G(цел n)
нач
  вывод "*"
  если n > D       |или n < D
    то
      F(n - b)     |или F(n + b)
  все
кон
```

— то схема вызовов функций будет иметь общий вид, показанный на *рис*. 9.

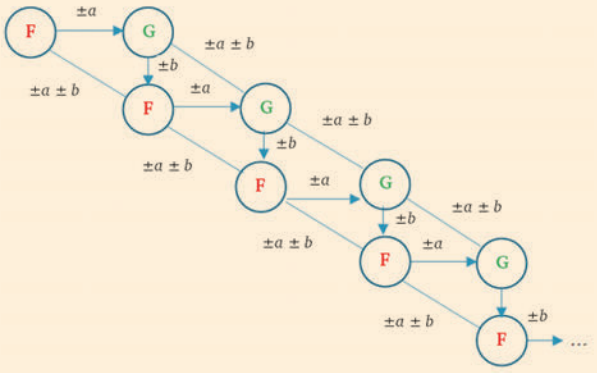



*Рис*. 9

Для задания из [5] соответствующий (упрощенный) вариант схемы будет таким:

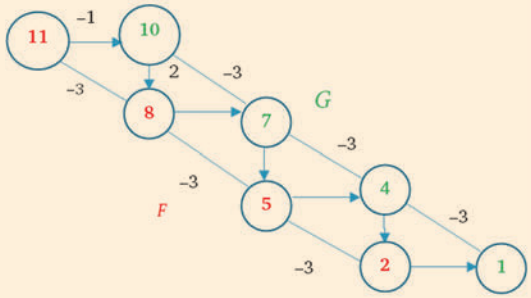



*Рис*. 10

Рассмотренные примеры показывают, что задания, связанные с определением количества выводимых на экран символов, аналогичные приведенному в [5], можно выполнять, нарисовав схему как на *рис*. 10. При этом от начальных значений параметров процедур *F* и *G* (в рассмотренном случае, соответственно, 11 и 10) необходимо вычитать или прибавлять сумму *a* и *b* с учетом их знаков (в рассмотренном случае — вычитать 1 + 2 = 3) до тех пор, когда станет ложным какое-то из условий, записанных в процедурах.

Искомое количество символов будет равно числу вызовов процедуры, в которой они выводятся, умноженному на количество символов, выводимых в процедуре.

## Задания для самостоятельной работы

**10.** Ниже на двух языках программирования записаны две рекурсивные процедуры: *F* и *G*.

**Алгоритмический язык**

```
алг F(цел n)
нач
  если n >= 2
    то
```

```
      G(n - 2)
  все
кон
алг G(цел n)
нач
  вывод "*"
  если n > 0
    то
      F(n - 1)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  if n >= 2 then
    G(n - 2)
end;
procedure G(n: integer);
begin
  write('*');
  if n > 0 then
    F(n - 1)
end;
```

Сколько символов “звездочка” будет напечатано на экране при выполнении вызова *F*(11)?

**11.** Ниже на двух языках программирования записаны две рекурсивные процедуры: *F* и *G*.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод "*"
  если n < 15
    то
      G(n + 1)
  все
кон
алг G(цел n)
нач
  если n < 14
    то
      F(n + 2)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  write('*');
  if n < 15 then
    G(n + 1)
end;
procedure G(n: integer);
begin
  if n < 14  then
    F(n + 2)
end;
```

Сколько символов “звездочка” будет напечатано на экране при выполнении вызова *F*(3)?

**12.** Ниже на двух языках программирования записаны две рекурсивные процедуры: *F* и *G*.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод "**"
  если n < 5
    то
      G(n - 1)
  все
кон
алг G(цел n)
нач
  если n < 4
    то
      F(n + 2)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  write('**');
  if n < 5 then
    G(n - 1)
end;
procedure G(n: integer);
begin
  if n < 4  then
    F(n + 2)
end;
```

Сколько символов “звездочка” будет напечатано на экране при выполнении вызова *F*(2)?

**13.** Ниже на двух языках программирования записаны две рекурсивные процедуры: *F* и *G*.

**Алгоритмический язык**

```
алг F(цел n)
нач
  вывод "!"
  если n > 3
    то
      G(n + 2)
  все
кон
алг G(цел n)
нач
  вывод "!"
  если n > 2
    то
      F(n - 3)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  write('!');
  if n > 3 then
    G(n + 2)
end;
procedure G(n: integer);
begin
  write('!');
  if n > 2 then
    F(n - 3)
end;
```

Сколько восклицательных знаков будет напечатано на экране при выполнении вызова *F*(7)?

В заключение заметим, что задания на использование рекурсии, в которых требуется опреде-

лить количество символов, выводимых на экран, представлены также в [3]. Их особенности:

1) вывод символов проводится после рекурсивных вызовов, например, требуется определить, сколько звездочек будет напечатано в результате вызова *F*(6) приведенной подпрограммы:

**Алгоритмический язык**

```
алг F(цел n)
нач
  если n > 1
    то
      F(n – 1)
      F(n – 2)
  все
  вывод "*"
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  if n > 1 then
    begin
      F(n – 1);
      F(n – 2)
    end;
  write('*')
end;
```

2) в подпрограмме имеется три рекурсивных вызова, например, требуется определить, сколько звездочек будет напечатано в результате вызова *F*(3) приведенной подпрограммы:

**Алгоритмический язык**

```
алг F(цел n)
нач
  если n > 0
    то
      F(n – 2)
      F(n – 1)
      F(n – 1)
  все
  вывод "*"
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  if n > 0 then
    begin
      F(n – 2);
      F(n – 1);
      F(n – 1)
    end;
  write('*')
end;
```

3) вывод символов, как и рекурсивные вызовы, происходит только по условию, например, требуется определить, сколько звездочек будет напечатано в результате вызова *F*(7) приведенной подпрограммы:

**Алгоритмический язык**

```
алг F(цел n)
нач
  если n > 1
    то
      вывод "*"
      F(n – 2)
      F(n – 1)
  все
кон
```

**Паскаль**

```
procedure F(n: integer);
begin
  if n > 1 then
    begin
      write('*');
      F(n – 2);
      F(n – 1)
    end
end;
```

## Литература


1. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена по информатике и ИКТ 2013 года. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*.

2. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена по информатике и ИКТ 2014 года. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*.

3. ЕГЭ-2014: Информатика: самое полное издание типовых вариантов заданий. / авт.-сост. *Д.М. Ушаков, А.П. Якушкин.* М.: АСТ: Астрель, 2014. (Федеральный институт педагогических измерений.)

4. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена по информатике и ИКТ 2015 года. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*.

5. КИМ ЕГЭ 2015 (досрочный период). *http://www.fipi.ru/sites/default/files/document/2015/05.pdf*.


## Ответы к заданиям для самостоятельной работы

| *Задание* | *Ответ* |
|---|---|
| 1 | 16 |
| 2 | 22 |
| 3 | 3 |
| 4 | 25 |
| 5 | 7 |
| 6 | 17 |
| 7 | 26 |
| 8 | 79 |
| 9 | 23 |
| 10 | 4 |
| 11 | 5 |
| 12 | 8 |
| 13 | 9 |

Общероссийский проект

**Школа цифрового века**

**6 тысяч рублей от школы**
**за весь 2015/16 учебный год**
**независимо от количества учителей**
**в образовательной организации**

Каждому учителю:

- 24 предметных ежемесячных журнала
- десятки курсов повышения квалификации

**Не забудьте принять**
**или продлить участие!**

Подробности и форма заявки на сайте:

**digital.1september.ru**

# Вопросы для конкурсов “Что? Где? Когда?” и “Брейн-ринг”

**Д.М. Златопольский,**
Москва

**0.** Как бы сформулировал правило русского языка о написании глаголов с частицей “не” любитель компьютеров?

**1.** Ведущий (обращаясь к одному из участников конкурса): “Вы моете руки после работы на компьютере?”. — (После ответа) — “Правильно” (или “Жаль”). Дело в том, что ученые подсчитали, что компьютерная клавиатура содержит в среднем 3295 микробов на 1 квадратный дюйм, или более 500 микробов на 1 квадратный сантиметр. А на какой клавише их больше всего и почему?”.

*Дополнительный вопрос (если ответ будет “Это самая большая клавиша”)*

А если считать, что эта клавиша такая же, как остальные?

**2.** В одной из первых отечественных электронно-вычислительных машин БЭСМ-4 использовался символический язык программирования, в котором команды записывались русскими буквами сокращенно, но по виду этих сокращений можно было понять, в чем заключается та или иная команда. Например, команда **ВВ** означала ввод данных с перфокарты. Какие команды представлены на табличке?

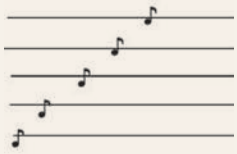
| |
|---|
| В |
| Д |
| С |
| У |

*2-й вариант вопроса (отличается табличкой, показываемой участникам конкурса)*

| |
|---|
| В |
| Д |
| К |
| С |
| У |

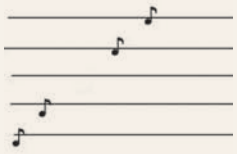
**3.** Ведущий (обращаясь к одному из участников конкурса): “Вы знаете, кто такой Штирлиц? — (После ответа) — Однажды Центр, которому он сообщал разведывательную информацию, вынужден был сменить ключ шифровок, направляемых ему, так как старый

ключ стал известен немцам. Через некоторое время Штирлиц получил пакет с нотами, в котором был такой фрагмент:

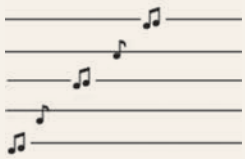


И хотя Штирлиц не знал нот, он понял, что это ключ шифровок. Какой новый ключ был передан нашему разведчику?".

*2-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*



*3-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*



**4.** В греческом их — 24, в латинском — 26. А сколько их в русском?

**5.** В художественных произведениях на военную и приключенческую темы, написанных на русском языке, очень важные события происходят, как правило, во время **Ч**, а написанных на английском языке — в час **X**. Почему именно так?

**6.** Французский писатель Виктор Гюго сказал, что "у истории ее нет". Все вы ею пользовались в жизни, а ее виртуальной разновидностью можете воспользоваться, не выключая компьютер. Что это?

**7.** В чем заключается метод подбора ключа к зашифрованному тексту, который по-английски называют *Brute-force attack* (атака методом грубой силы)?

**8.** 1/2 этой буквы означает одно из основных понятий баз данных. Что это за буква и какое это понятие?

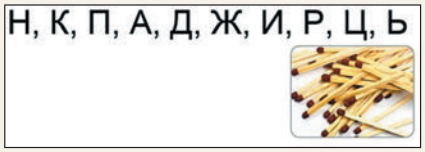
**9.** Какой цветок мог бы стать "цветочным" символом долговременной (внешней) памяти?

**10.** Как известно, двумерный массив называют "матрица". А как называют двумерный массив, в котором число строк равно числу столбцов?

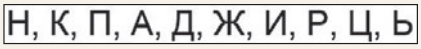
**11.** Чем особенны приведенные ниже буквы?


*2-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*

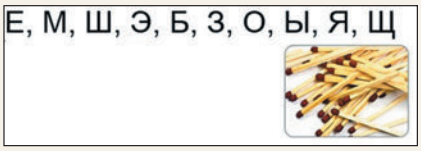

**12.** Чем особенны приведенные ниже буквы?



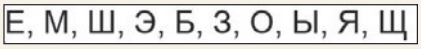

*Вариант вопроса (отличается рисунком, показываемым участникам конкурса)*




**13.** Чем особенны приведенные ниже буквы?




*Вариант вопроса (отличается рисунком, показываемым участникам конкурса)*




**14.** Какая из приведенных на картинке цифр является "лишней"?

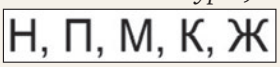

*2-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*


**15.** Какая из приведенных на картинке букв является "лишней"?


*Вариант вопроса (отличается рисунком, показываемым участникам конкурса)*


**16.** Какая из приведенных на картинке букв является "лишней"?

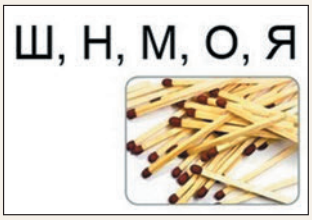

*2-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*




**17.** Какая из приведенных на картинке букв является "лишней"?


*2-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*

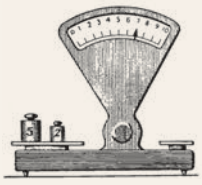

**18.** Какая из приведенных на картинке букв является “лишней”?




*2-й вариант вопроса (отличается рисунком, показываемым участникам конкурса)*


**19.** Существует два типа вычислительных устройств. С одним из них вы хорошо знакомы. Это так называемые “цифровые” вычислительные машины и устройства, в которых обрабатываются величины, представленные в виде цифр. Персональные компьютеры относятся к этому типу.

В устройствах второго типа вычислительный процесс заменяется, или, как говорят, “моделируется”, некоторым физическим процессом, и результат вычислений получается с помощью измерения той или иной физической величины (тока, напряжения и т.п.), часто просто путем чтения этого результата на какой-либо шкале. Например, можно механизировать сложение, заменив эту вычислительную операцию физической операцией взвешивания. Посмотрите, пожалуйста, на экран (участникам конкурса демонстрируется следующее изображение).



Например, если нужно сложить числа 2 и 5, то, положив на чашку весов гири в 2 и в 5 кг, мы получим на шкале сумму, равную 7 кг, и на основании этого заключим, что 2 + 5 = 7.

Как назывались вычислительные устройства такого типа?

**20.** Какое из вычислительных устройств, представленных на рисунке, является “лишним”?

**21.** Посмотрите, пожалуйста, на картинку:

Почему на ней представлены и весы?

## Ответы

**0.** Как известно, правило, о котором шла речь в вопросе, такое: “Частица *не* с глаголами пишется раздельно”. Любитель компьютеров сформулировал бы это правило следующим образом: “Частица *не* с глаголами пишется через пробел”.

**1.** На клавише “Пробел”. Соответствующий “символ” чаще всего встречается в текстах, поэтому и клавиша используется чаще других.

**2.** На табличке приведены команды, по которым, соответственно, выполнялись вычитание, деление, сложение и умножение.

*Ответ на вариант вопроса*

На табличке приведены команды, по которым выполнялись, соответственно, вычитание, деление, извлечение квадратного корня, сложение и умножение.

**3.** В форме нотной записи была передана информация о числе 31, которое в двоичной системе имеет вид 11111 (символ “♪” соответствует единице).

*Ответ на 2-й вариант вопроса*

Был передан ключ, равный 27 (это число в двоичной системе имеет вид 11011).

*Ответ на 3-й вариант вопроса*

Если принять, что символ “♫” соответствует единице, а символ “♪” — нулю, то передана информация 10101, или десятичное число 21. (Ответ 10, соответствующий двоичному числу 01010, также можно считать правильным.)

**4.** 33 (количество букв в алфавите).

**5.** Потому что это одна и та же клавиша на клавиатуре. Писатели, набиравшие текст на русском языке, тоже хотели написать “час X”, но переключать регистр было неохота ☺.

**6.** Папка операционной системы Windows **Корзина**.

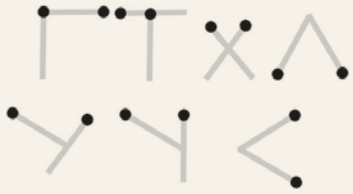
**7.** В переборе всех возможных вариантов.

**8.** Речь идет о букве “Е”: полЕ (половина Е) — поле (столбец в таблице базы данных, в котором представлена информация о каком-то свойстве объектов).

**9.** Незабудка.

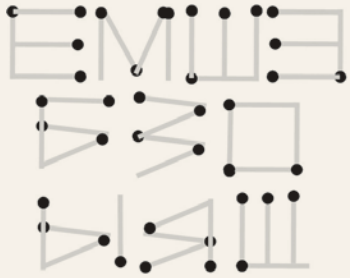
**10.** “Квадратная матрица”.

**11.** Все приведенные буквы (или их схематическое изображение) могут быть получены с помощью двух спичек или т.п. (см. рисунок ниже).

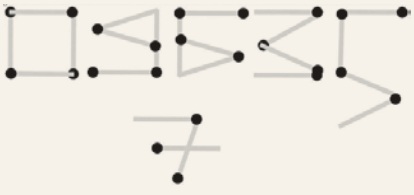


**12.** Все приведенные буквы (или их схематическое изображение) могут быть получены с помощью трех спичек или т.п. (см. рисунок ниже).

**13.** Все приведенные буквы (или их схематическое изображение) могут быть получены с помощью четырех спичек или т.п. (см. рисунок ниже).



**14.** “Лишней” является цифра 7, так как ее схематическое изображение можно получить с помощью трех спичек или т.п., в то время как остальные цифры — с помощью четырех (см. рисунок ниже).



**15.** “Лишней” является буква Н, так как ее схематическое изображение можно получить с помощью трех спичек или т.п., в то время как остальные буквы — с помощью двух (см. рисунок ниже).

**16.** “Лишней” является буква М, так как ее схематическое изображение можно получить с помощью четырех спичек или т.п., в то время как остальные буквы — с помощью трех (см. рисунок ниже).

**17.** “Лишней” является буква Т, так как ее схематическое изображение можно получить с помощью двух спичек или т.п., в то время как остальные буквы — с помощью трех (см. рисунок ниже).

**18.** “Лишней” является буква Н, так как ее схематическое изображение можно получить с помощью трех спичек или т.п., в то время как остальные буквы — с помощью четырех (см. рисунок ниже).

**19.** Вычислительные машины непрерывного действия, или моделирующие устройства, или аналоговые устройства.

**20.** “Лишней” является так называемая “логарифмическая линейка”, которая относится к вычислительным устройствам непрерывного действия, или, иначе, — к аналоговым вычислительным устройствам. В таких устройствах вычислительный процесс заменяется, или, как говорят, “моделируется”, некоторым физическим процессом, и результат вычислений получается с помощью измерения той или иной физической величины (тока, напряжения и т.п.), часто просто путем чтения этого результата на какой-либо шкале. В частности, в логарифмической линейке действие умножения заменяется измерением длин отрезков.

Все остальные представленные на рисунке вычислительные устройства относятся к так называемым “цифровым вычислительным устройствам”, в которых обрабатываются величины, представленные в виде цифр.

**21.** Можно считать, что на пружинных весах операция взвешивания моделирует операцию сложения (если иметь много разных гирь и точную шкалу). Например, если нужно сложить числа 2 и 5, то, положив на чашку весов гири в 2 и в 5 кг, мы получим на шкале сумму, равную 7 кг, и на основании этого заключим, что $2 + 5 = 7$.



## ИСТОРИЯ ИНФОРМАТИКИ

# “Персональные компьютеры” в докомпьютерную эру

**В.В. Шилов,**
Москва

*В статье рассказывается о работах изобретателей XIX века, пытавшихся создать простое, надежное и дешевое карманное счетное устройство.*

В изданном в 1854 г. романе “Тяжелые времена” великий английский писатель Чарльз Диккенс дал такую характеристику одному из персонажей: “Томас Грэдграйнд, сэр. Человек трезвого ума. Человек очевидных фактов и точных расчетов. Человек, который исходит из правила, что дважды два — четыре, и ни на йоту больше <…> Вооруженный линейкой и весами, с таблицей умножения в кармане, он всегда готов взвесить и измерить любой образчик человеческой природы и безошибочно определить, чему он равняется. Это всего-навсего подсчет цифр, сэр, чистая арифметика”[1].

Диккенс, художник крайне наблюдательный и точный в деталях, подметил одну примечательную черту бытового обихода середины XIX столетия. Речь идет о таблице умножения, лежащей в кармане мистера Грэдграйнда. Разумеется, вполне возможно, что для писателя это был просто один из штрихов, необходимых для характеристики героя — человека сухого и прагматичного. Однако и на самом деле во времена Диккенса именно таблица умножения, а также сборники других таблиц (сложных процентов, пересчета валют, мер и весов и т.д.) издавались огромными тиражами и обязательно находились под рукой едва ли не каждого торговца, счетовода или страхового агента. Работа любого мелкого предпринимателя была неразрывно связана с вычислениями и подсчетами, и средства для их ускорения и упрощения были им совершенно необходимы.

При этом альтернативы использованию таблиц практически не было. Ни появление арифмометров Тома́ де Кольмара (хотя их производство началось еще в 1822 г., массовый выпуск был налажен только в последней трети века), ни — гораздо позднее — других счетных машин не изменили ситуацию радикально. Все они были слишком дорогими, не слишком надежными и, самое главное, чересчур большими. Десятки изобретателей в Англии, Франции, России, Германии настойчиво пытались дать ответ на требования практики. Но особенно активно они работали в Америке. С 1843 года, когда в США был выдан первый патент такого рода, и до 1900 года ими были изобретены несколько сот разнообразных счетных устройств: логарифмических линеек, вычислительных шкал, сумматоров и др. Особенно очевидной была потребность в небольшом карманном сумматоре. Это сулило изрядную прибыль тому, кто первым добьется успеха, однако очень долго потребители никак не могли получить достаточно надежный, простой в эксплуатации, компактный и дешевый счетный прибор.

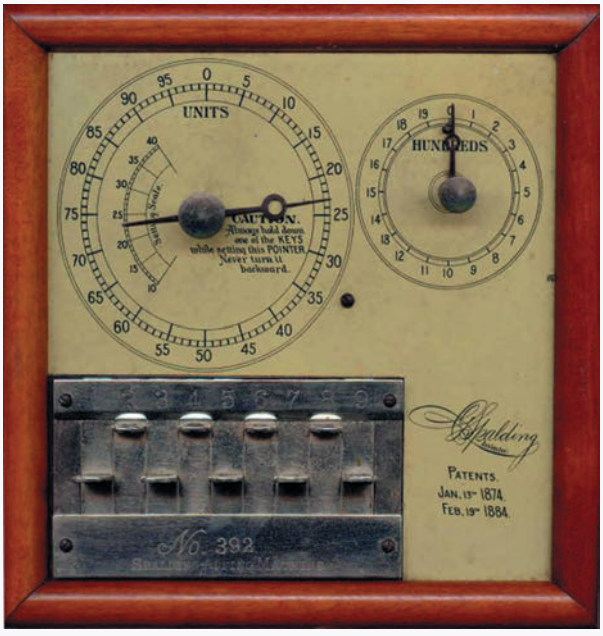
Некоторые сумматоры были основаны на интересных технических решениях, другие имели весьма элегантный и привлекательный дизайн, но покупателя они все же чем-то не устраивали. Житель Огайо Самюэл Янг (Samuel Young), изготовивший в 1849 г. карманный сумматор из дерева; гражданин Пенсильвании Аарон Хэтфилд (Aaron Hatfield), пятью годами позже запатентовавший оригинальный сумматор — ручка, за которую его нужно было держать, придавала прибору сходство с ракеткой для настольного тенниса; Джон Гросбек (John Groesbeck, 1834–1884) из Филадельфии в 1870 г.; Арчибальд Стефенсон (Archibald Stephenson, 1844–1913) из Иллинойса в 1873 г.; Сайрус Спалдинг (Cyrus Spalding) из Массачусетса на следующий год, и многие другие — никто из них так и не преуспел.

[1] *Чарльз Диккенс*. Собрание сочинений в 30 томах. Т. 19. М.: Гос. изд-во художественной литературы, 1960. С. 8. Перевод В.Топер.

*Сумматор Сэмюэла Янга, 1849 г.*

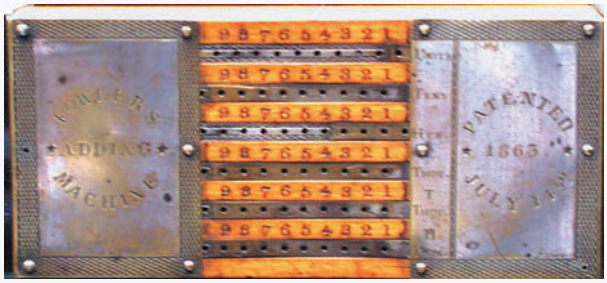
*Сумматор Аарона Хэтфилда, 1854 г.*


*Сумматор Джона Гросбека, 1870 г.*




*Сумматор Сайруса Спалдинга, 1874 г.*

Возможно, ближе всех к успеху подошел Джордж Фаулер (George Fowler) из Чикаго, который стал выпускать сумматор своей конструкции в 1863 г. Его компания *G. B. Fowler & Co*. вошла в историю как первая в США фирма, начавшая серийный выпуск счетных устройств. Поначалу сумматор Фаулера пользовался вниманием прессы, не жалевшей самых восторженных выражений. Журнал *Harper's Weekly* шутливо писал 13 февраля 1869 г.: “Мы предлагаем Министерству просвещения исключить арифметику из школьных программ, поскольку теперь можно складывать посредством машины. Хочу снова стать ребенком! — вскричат все, кто в детстве стенал над таблицей умножения”. Журнал охарактеризовал точность вычислений как бесспорную, быстроту работы — как изумительную, а ее простоту — как замечательную. Однако затем Фаулер ушел в тень, и хотя, судя по всему, дела у него шли неплохо (в 1890 г. он даже возобновил свой патент), все-таки большого распространения его сумматор не получил.




*Сумматор Джорджа Фаулера, 1863 г.*

Здесь стоит заметить, что американский изобретатель ориентировался в первую очередь на коммерческий успех, примеры которого явили Сэмюэл Морзе, Томас А. Эдисон, Александр Белл и некоторые другие великие изобретатели. Его деятельность должна была обеспечить не только признание благодарных потомков, но и прижизненную славу и процветание. Эпоха бурного промышленного развития США (которую Марк Твен метко назвал “позолоченным веком”) открывала перед энергичным, умным и творческим человеком массу возможностей, надо было только суметь ими воспользоваться.

Человеком, в полной мере реализовавшим эту жизненную установку, стал один из самых ярких и оригинальных персонажей американской культурной жизни второй половины XIX в. — предприниматель, журналист, издатель, поэт и изобретатель Чарльз Генри Вебб (Charles Henry Webb). Он родился 24 января 1834 г. в поселке Раузис-пойнт (Rouse's Point) (штат Нью-Йорк). Однако здесь молодому человеку не хватало простора, и, окончив школу, Вебб на три года ушел в море — благо, оно было совсем рядом. Вернувшись на сушу, он вслед за родителями перебрался вглубь страны, в Иллинойс. “Жизнь на Миссисипи” (если воспользоваться названием известного произведения Марка Твена) кипела. Энергичный молодой человек с 1856-го по 1860 г. занимался бизнесом на берегах великой реки, охотно берясь за любое предприятие, сулящее прибыль. Спустя какое-то время Вебб обосновался в Чикаго, центре зерновой торговли страны. А еще позднее он вернулся в Нью-Йорк, где попробовал себя в качестве биржевого брокера на Уолл-стрит.

Однако то ли дела шли не слишком успешно, то ли чего-то иного потребовала душа — но уже в начале 1860-х гг. Вебб начал сотрудничать с газетой “Нью-Йорк таймс”. Вскоре журналистика и писательство захватили его целиком. Вебб переехал в Сан-Франциско, где, неожиданно даже для самого себя, оказался в роли признанного лидера нарождающейся калифорнийской литературы.

В Сан-Франциско Вебб редактировал “Калифорниен” — литературный еженедельник, ставший центром притяжения для многих местных молодых талантов. Именно в нем увидели свет первые литературные опыты будущего выдающегося певца Калифорнии Фрэнсиса Брет Гарта. Здесь же Вебб познакомился с Марком Твеном, также только-только вступавшим на путь журналистики. В самом начале 1867 года в Нью-Йорке их общение возобновилось, и эта новая встреча оставила заметный след в истории американской литературы. Как позднее вспоминал сам Твен, Вебб “предложил мне издать сборник рассказов. Я еще не пользовался такой известностью, чтобы стоило издавать сборник, однако пришел в восторг и настолько пленился этим предложением, что дал свое согласие. <…> Он выпустил очень хорошенькую книжечку, синюю с золотом. Кажется, она у него называлась «Знаменитая скачущая лягушка округа Калаверас» и другие рассказы»”.

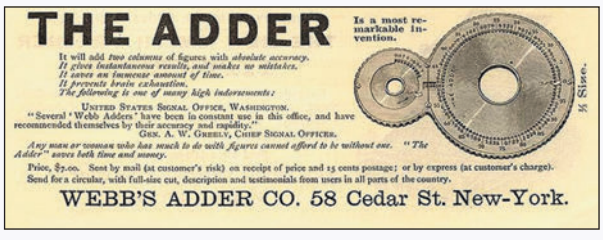
Книга увидела свет в мае 1867 г. и стала первым шагом великого писателя по пути к любви и признанию читателей, а также — без преувеличения — грандиозной всемирной славе. Правда, именно из-за этой книги отношения автора и издателя вскоре испортились: Марк Твен счел (сегодня трудно судить, насколько обоснованно), что Вебб обманул его при расчетах. Всю эту историю Твен, дав полную волю своему неподражаемому юмору и ядовитому сарказму, описал в изданной посмертно “Автобиографии”.

В 1866 г. Вебб в очередной раз вернулся в Нью-Йорк и с головой окунулся в его литературную жизнь. С этого времени он регулярно публиковал в журналах атлантического побережья рассказы, очерки и стихотворения, издал несколько пользовавшихся успехом книг-пародий на популярные сенсационные романы. Однако постепенно выяснилось, что литературный труд также не давал достаточных средств к существованию, и Вебб решил попробовать свои силы в изобретательстве. Вряд ли можно сомневаться в том, что именно богатый опыт предпринимательства подсказал ему, в каком изделии особенно нуждается рынок.

10 марта 1868 года Вебб получил патент на сумматор (Adding-Machine), который сам называл “числовым велосипедом” — происхождение этого названия становится понятным при взгляде на прибор. Прибор размером 16,8 × 13,1 × 2,1 см представлял собой два латунных диска, заключенных в корпус красного дерева и прикрытых сверху латунной пластинкой (спустя несколько лет прибор стал полностью металлическим). Большой диск был проградуирован числами от 00 до 99, маленький — от 0 до 9. Для ввода очередного числа надо было вставить тонкий штифт в отверстие напротив соответствующей цифры и повернуть большой диск. Когда большой диск совершал полный оборот, маленький сдвигался на одну позицию. Цифры в двух окошках показывали результат сложения, который не должен был превышать 999.


*Сумматор Вебба. Модель 1868 г.*

Сумматор Вебба имел оглушительный успех. И, помимо достоинств самого сумматора, немаловажную роль в успехе предприятия Вебба сыграли также и его журналистский опыт, понимание значения рекламы и ее влияния на рыночную судьбу товара. Он помог выбрать удачную стратегию продвижения прибора к потребителю. Вскоре после того, как Вебб наладил производство, американские газеты запестрели рекламными объявлениями типа “Новое чудо”, “Экономит время и деньги”, “Суждено стать самой распространенной и полезной из когда-либо изобретенных математических машин”, “Замечательный триумф изобретательского гения” и т.п.




*Реклама сумматора Вебба*

Наверняка задействовал Вебб и свои связи в мире бизнеса. Благоприятные отзывы о сумматоре дали в печати многие влиятельные лица. Мэр Нью-Йорка писал, что под его наблюдением прибор испытали бухгалтеры и нашли его работу столь точной и надежной, что он распорядился использовать сумма-

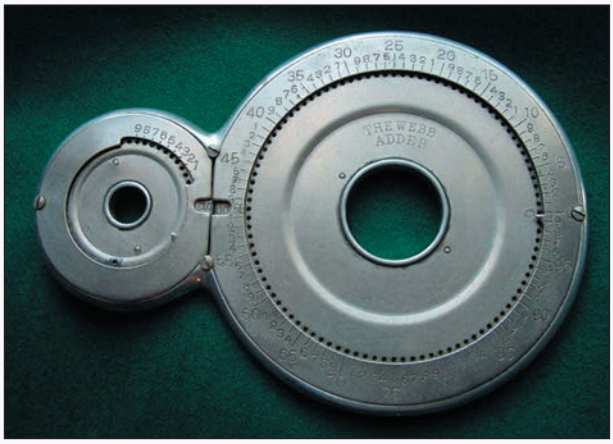
торы в работе различных городских служб. Ему вторил президент Первого Национального банка Нью-Йорка: “Прибор полностью освобождает вашу голову от забот и сводит арифметику к простому движению руки”. *Harper's Weekly* в номере от 2 февраля 1869 г. утверждал, что сумматор Вебба — единственная в мире действительно работающая суммирующая машина и констатировал, что спрос на него пока еще невозможно удовлетворить полностью.

Сегодня сумматор Вебба кажется нам устройством более чем простым, однако полтора столетия назад он стал сенсацией и будоражил умы своими (не столь уж большими, как мы понимаем) возможностями. В том же *Harper's Weekly* спустя две недели было написано (в колонке юмора, но едва ли не всерьез): “Не изобретет ли некий гений машину для написания передовиц, интересных заметок и прочего? В наш передовой век кто-нибудь в самом деле должен взять на себя труд изготовить ее! Едва ли это будет сложнее, чем создать математическое устройство для сложения чисел столь же простое, как сделанное м-ром Ч.Г. Веббом”.

Таким образом, устройство, предназначенное для сложения последовательностей небольших чисел, представлялось современникам прообразом неведомой сложнейшей машины, способной заменить человека в деле сочинительства. Честное слово, похоже, что анонимный журналист предвосхитил позднейшие дискуссии на тему “может ли машина мыслить”!

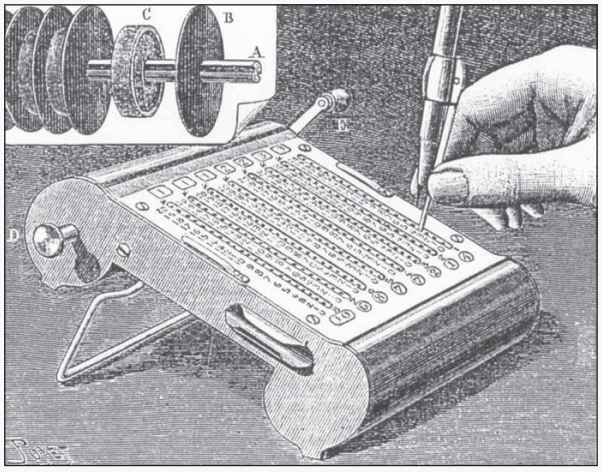
Но интерес к сумматору проявляли не только практики, заинтересовал он и ученых, и инженеров. Известно, что вскоре после появления прибора два экземпляра сумматора Вебба были приобретены Нью-Йоркской метеорологической станцией и использовались для производства расчетов. Не случайно вслед за популярной прессой подробное описание прибора Вебба появилось в научных изданиях: в журнале *Scientific American* (1869 г.), а еще через год — в журнале Института Франклина. Даже военные высоко оценили сумматор — в частности, он нашел применение в войсках связи США. Наконец, сумматор Вебба был удостоен множества отличий на различных региональных и национальных промышленных выставках. В немалой степени рыночному успеху способствовала вполне доступная цена: стоимость сумматора составляла от 6 до 10 долларов, в зависимости от использованного материала и оформления.

Вебб стал первым, кто сумел преуспеть на новом рынке вычислительных устройств. Собственно говоря, именно успех его сумматора и способствовал формированию этого рынка! Стараясь развить свой успех, Вебб продолжал совершенствовать конструкцию сумматора и в 1889 г. получил еще один патент, в котором закрепил все изменения, которые были внесены за два десятка лет производства и эксплуатации прибора, в первую очередь новый механизм переноса. Увеличилась и емкость сумматора, теперь максимальная сумма равнялась 4999.




*Сумматор Вебба. Модель 1889 г.*

Но, кроме того, в декабре 1891 г. он запатентовал суммирующее устройство принципиально новой конструкции, получившее название “Ленточный сумматор” (Ribbon Adder). Механизм прибора заключен в элегантный хромированный корпус с 8 прорезями на лицевой стороне, предназначенными для ввода и суммирования отдельных чисел. Слева от прорезей на корпус нанесены числа от 0 до 20. Внутри, под каждой прорезью, находится широкая металлическая лента с расположенными через равные интервалы отверстиями в ней. По обе стороны от каждого отверстия также нанесены числа. Справа — повторяющиеся последовательности цифр от 0 до 9. Слева же группами по 10 нанесены числа 0, 1, 2 и так далее, вплоть до 29. Они видны в окошке, расположенном над лентой, и означают число, которое необходимо перенести в следующий разряд. Иными словами, фактически на ленту нанесены числа от 0 до 299. Вставляя в отверстия штифт, можно сдвигать ленту, тем самым последовательно вводя в прибор суммируемые числа. Таким образом, на каждой ленте можно получить сумму, максимально равную 299, — в этом случае в нижнем окошке будет показано число 9, а число 29 будет отображено в верхнем окошке, и его надо будет перенести на соседнюю ленту.




*Ленточный сумматор Вебба*

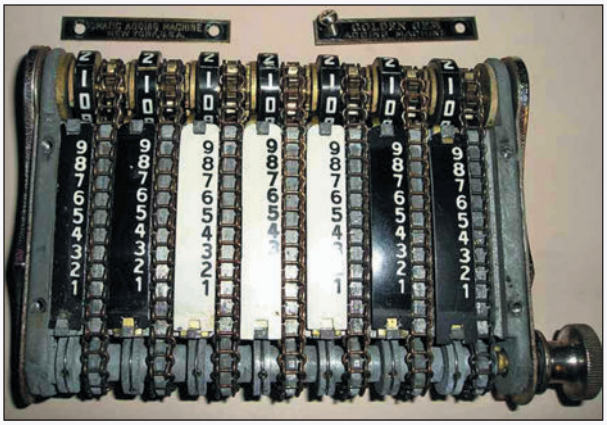
Реклама ленточного сумматора Вебба обещала покупателю счетный прибор ценой всего 15 долларов, но “выполняющий ту же работу, что и в де-

сять раз более дорогая счетная машина". Среди его достоинств отмечались также бесшумная работа, отсутствие в конструкции ненадежных элементов вроде рычагов и пружин, а также простота освоения и использования. Особо подчеркивалось, что это единственная на рынке счетная машина, в которой реализован совершенно новый принцип работы, а не какие-нибудь незначительные усовершенствования давно известного.

В связи с последним рекламным утверждением следует отметить, что идея ленточного сумматора была предложена и реализована за несколько лет до Вебба. Еще в конце 1870-х гг. русский изобретатель Юрий Дьяков изготовил счетный прибор ("новые счеты"), состоявший из помещенных в деревянный ящик движущихся клеенчатых лент с нанесенными на них цифрами; для удобства сдвига лент на них имелись небольшие медные планочки. Этот прибор был удостоен премии на выставке в Париже в 1878 г. Приоритет Дьякова признают ведущие историки вычислительной техники. Не исключают они и того, что "новые счеты" Дьякова были известны Веббу; так ли это — сказать трудно, однако в производстве ленточных сумматоров Вебб, несомненно, стал первым.

Интерес к новинке со стороны потенциальных потребителей был — уже упоминавшаяся Нью-Йоркская метеорологическая станция немедленно приобрела экземпляр на пробу. Но, к сожалению, выход нового счетного прибора на рынок совпал с экономическим кризисом 1893 г., вызвавшим резкий спад покупательной способности населения. По этой причине ленточный сумматор Вебба достаточно быстро исчез из продажи. Очевидно, что это не было связано с недостатками самого сумматора, поскольку позднее выпуск аналогичных устройств наладили несколько других фирм. В их числе можно назвать сумматоры американских фирм *Golden Gem* (1904 г.), *J. H. Bassett & Co.* (1909 г.), а также *Comptator* (Германия, около 1920 г.) и некоторые другие. Устойчивый спрос на ленточные сумматоры сохранялся еще несколько десятилетий: достаточно сказать, что фирма Bassett с успехом продавала свою продукцию вплоть до Второй мировой войны.

*Чарльз Вебб*



*Механизм ленточного сумматора Golden Gem*

Сумматор Вебба имел несомненный успех, количество проданных приборов исчислялось многими тысячами. Именно Вебб проложил дорогу десяткам других фирм, наводнивших американский рынок различными счетными устройствами и машинами. Например, изобретатель из Нью-Джерси Джеймс Уолш запатентовал в 1901 г. карманный сумматор, известный под названием *Calcumeter*. Интересно, что механизм переноса, приводимый в действие энергией пружин, был аналогичен использованному Веббом. *Calcumeter* отличался изящным дизайном и пользовался огромным спросом — за 20 лет было продано свыше 100 000 штук. В начале XX в. сопоставимыми были объемы продаж и многих других карманных сумматоров.


*Calcumeter*

Производство Вебба было пусть и не первым, но одним из первых, и самое главное — первым успешным. А изобретения Чарльза Генри Вебба обеспечили ему скромное, но вполне почетное место в истории вычислительной техники.

## ПОИСК ИНФОРМАЦИИ

### Три вопроса по истории

**1.** В 1904 году перед нейтральным портом Чемульпо появилась японская эскадра, потребовав от двух заблокированных русских кораблей сдаться. Как назывались эти корабли?

**2.** Нижним чином в полиции царской России был городовой. Ему полагались в качестве оружия револьвер и форменная шашка. Остряки того времени дали упомянутому холодному оружию весьма игривое прозвище. Какое?

**3.** Известный анархист Михаил Бакунин, естественно, был противником регулярной армии. Но однажды, когда некое известное лицо нелестно отозвалось о качествах русских солдат, он вызвал на дуэль оскорбителя его державной гордости. Тот от поединка отказался. Кто струсил?

Ответы (можно не на все вопросы) присылайте в редакцию.

# Еще раз о “счастливых” билетах

**Д.М. Златопольский,**
Москва,
**Е.А. Мирончик,**
г. Новокузнецк Кемеровской обл.

В разделе “В мир информатики” была опубликована статья [1], связанная с подсчетом количества так называемых “счастливых” билетов (шестизначный номер автобусного или подобного билета считается “счастливым”, если сумма трех его первых цифр равна сумме трех последних; например, “счастливым” будет билет с номером 627294). В статье были исследованы возможности сокращения размера компьютерной программы, решающей задачу подсчета. В настоящей статье рассмотрим возможности уменьшения времени работы программы.

Самый очевидный алгоритм решения задачи подсчета количества “счастливых” шестизначных номеров билетов аналогичен алгоритму действия человека, решающего эту задачу:

— рассматриваются все номера (числа от 0 до 999999);

— определяется каждая цифра числа;

— рассчитывается сумма трех первых цифр;

— рассчитывается сумма трех последних цифр;

— рассчитанные суммы сравниваются между собой; если они равны, то данный номер — “счастливый” и количество найденных “счастливых” номеров увеличивается на 1.

Разработаем соответствующую программу (как принято в разделе “В мир информатики”, — с использованием школьного алгоритмического языка; его русский синтаксис делает программу максимально понятной и легко переносимой на любой другой язык программирования).

В программе применим следующие величины:

— *номер* — проверяемый номер;

— *пер*, *вт*, *тр*, *чет*, *пят*, *шес* — соответственно первая, вторая, …, шестая цифры номера;

— *кол* — искомое количество.

Прежде чем представлять программу, заметим, что для выделения цифр следует использовать функции *mod* и *div*, возвращающие, соответственно, остаток от деления одного целого числа на другое и их целочисленное частное (в других языках программирования используются не функции, а специальные операции).

Примем также, что номера могут иметь начальные нули и что возможен номер билета 000000.

Итак, программа:

```
алг Число_счастливых_номеров_Вариант_1
нач цел номер, кол, пер, вт, тр, чет,
        пят, шес
  кол := 0
  |Рассматриваем все номера
  нц для номер от 0 до 999999
    |Выделяем каждую цифру номера
    шес := mod(номер, 10)
    пят := mod(div(номер, 10), 10)
    чет := mod(div(номер, 100), 10)
    тр := mod(div(номер, 1000), 10)
    вт := mod(div(номер, 10000), 10)
    пер := div(номер, 100000)
    |Сравниваем суммы троек цифр
    если пер + вт + тр = чет + пят + шес
      то |Встретилось очередное
         |"счастливое" число
        кол := кол + 1
    все
  кц
  |Выводим ответ
  вывод нс, кол
кон
```

Обратим внимание на то, что, хотя часть проверяемых номеров — менее чем шестизначные, для них также рассчитываются шесть цифр (“недостающие” цифры будут равны нулю).

Подумаем над тем, как можно сократить размер программы. Анализ показывает, что в ней для каждого номера дважды определяется сумма трех цифр. Когда в программе имеются одинаковые или аналогичные по смыслу фрагменты, как правило, целесообразно создать вспомогательную процедуру или функцию. В нашем случае — функцию, рассчитывающую сумму цифр трехзначного числа. Эта функция имеет вид:

```
алг цел СуммаЦифр(арг цел число)
нач цел пер, вт, тр
  пер := div(число, 100)
  вт := mod(div(число, 10), 10)
  тр := mod(число, 10)
  знач := пер + вт + тр |Значение функции
кон
```

С ее использованием основная часть программы оформляется достаточно компактно:

```
алг Число_счастливых_номеров_Вариант_2
нач цел номер, полов1, полов2, кол
  кол := 0
  нц для номер от 1000 до 999999
    полов1 := div(n, 1000)
    полов2 := mod(n, 1000)
    если СуммаЦифр(полов1) =
         СуммаЦифр(полов2)
      то
        кол := кол + 1
    все
  кц
  вывод нс, кол
кон
```

— где *полов*1 и *полов*2 — соответственно первая и вторая половины шестизначного (в общем случае) числа.

Однако, как показывают расчеты, время выполнения программы при этом увеличивается пример-

но на 20%. Это связано с многократными обращениями (вызовами) к вспомогательной функции.

Идею разбиения номера на две части можно применить следующим образом. Можно рассмотреть все возможные значения первой половины шестизначного номера (от 0 до 999) и для каждого значения, найдя сумму его цифр, сравнить ее с суммой цифр всех возможных значений второй половины. Соответствующий вариант программы имеет вид:

```
алг Число_счастливых_номеров_Вариант_3
нач цел кол, полов1, полов2, пер, вт, тр,
    сумма123, сумма456 |суммы троек цифр
  кол := 0
  нц для полов1 от 0 до 999
    пер := div(полов1, 100)
    вт := mod(div(полов1, 10), 10)
    тр := mod(полов1, 10)
    сумма123 := пер + вт + тр
    нц для полов2 от 0 до 999
      пер := div(полов2, 100)
      вт := mod(div(полов2, 10), 10)
      тр := mod(полов2, 10)
      сумма456 := пер + вт + тр
      если сумма123 = сумма456
        то
          кол := кол + 1
      все
    кц
  кц
  вывод нс, кол
кон
```

Расчеты показывают, что время выполнения программы по сравнению с первым вариантом уменьшается примерно на 30%!

Разовьем идею разбиения числа на части и разработаем вариант программы, в котором рассматриваются отдельные цифры номера и все их возможные сочетания, то есть используем в программе шесть операторов цикла с параметром:

```
алг Число_счастливых_номеров_Вариант_4
нач цел кол, пер, вт, тр, чет, пят, шес
  кол := 0
  нц для пер от 0 до 9
    нц для вт от 0 до 9
      нц для тр от 0 до 9
        нц для чет от 0 до 9
          нц для пят от 0 до 9
            нц для шес от 0 до 9
              если пер + вт + тр =
                   чет + пят + шес
                то
                  кол := кол + 1
              все
            кц
          кц
        кц
      кц
    кц
  кц
  вывод нс, кол
кон
```

Оказывается, в этом случае время выполнения программы по сравнению с первым вариантом уменьшается более чем на 55%!

Казалось бы, все резервы ускорения программы исчерпаны. Однако это не так.

Одно из направлений ускорения основано на следующих рассуждениях. Если мы знаем первые пять цифр проверяемого номера, например, 16712, 16774 или 10145, то можем определить шестую из условия “счастливости” номера. Для приведенных примеров соответствующие значения равны:

$1 + 6 + 7 – (1 + 2) = 11$

$1 + 6 + 7 – (7 + 4) = 3$

$1 + 0 + 1 – (4 + 5) = –7$

Это означает, что на значение шестой цифры может “претендовать” единственное число, и в программе следует проверить, является ли это число цифрой, то есть принадлежит диапазону 0..9. Поэтому в последнем варианте программы вместо шестого (“самого внутреннего”) оператора цикла следует записать два оператора:

```
алг Число_счастливых_номеров_Вариант_5
нач цел кол, пер, вт, тр, чет, пят, шес
  кол := 0
  …
  нц для пят от 0 до 9
    |Определяем требуемую шестую цифру
    шес := пер + вт + тр - (чет + пят)
    |Проверяем, является ли найденное
    |число цифрой
    если шес >= 0 и шес <= 9
      то     |Подходит
        кол := кол + 1
    все
  кц
…
```

Исключение одного оператора цикла уменьшает, как показывают расчеты, продолжительность выполнения программы еще на 2,0–2,5%.

Еще одно направление оптимизации заключается в ограничении возможных значений четвертой и пятой цифр. Например, если номер начинается с 121, то четвертая цифра может быть 0, 1, 2, 3 или 4. С другой стороны, надо помнить, что это цифра. Поэтому можно записать:

```
если сумма123 < 10
  то
    макс_чет := сумма123
  иначе
    макс_чет := 9
все
```

— где *сумма*123 — сумма трех первых цифр номера, *макс_чет* — максимально возможное значение четвертой цифры (его можно использовать в операторе цикла с параметром *чет*).

Аналогично для пятой цифры максимально возможное значение равно:

```
если сумма123 — чет < 10
  то
    макс_пят := сумма123 — чет
  иначе
    макс_пят := 9
все
```

(В дальнейшем для лучшего понимания и краткости разность *сумма*123 — *чет* будем обозначать *сумм*56.)

Можно также ограничить *минимально* возможные значения четвертой и пятой цифр. Например, если номер начинается с 998, то четвертая цифра может быть 9 или 8. Анализ таблицы:

| Сумма трех первых цифр | Возможные значения четвертой цифры |
|---|---|
| 27 | 9 |
| 26 | 8 9 |
| … | |
| 19 | 1 2 3 4 5 6 7 8 9 |
| 18 | 0 1 2 3 4 5 6 7 8 9 |
| 17 | 0 1 2 3 4 5 6 7 8 9 |

показывает, что минимально возможное значение четвертой цифры может быть определено следующим образом:

```
если сумма123 > 18
  то
    мин_чет := сумма123 - 18
  иначе
    мин_чет := 0
все
```

Аналогично для пятой цифры можно установить:

```
если сумма56 > 9   |сумм123 — чет > 9
  то
    мин_пят := сумма56 — 9
  иначе
    мин_пят := 0
все
```

А теперь — внимание! Сейчас обсудим важное и неочевидное утверждение. Получив значение величины *сумма*56, мы знаем сумму, которую должны составлять пятая и шестая цифры, чтобы номер оказался “счастливым”. При этом количество возможных значений пятой цифры также известно (оно равно *макс_пят* — *мин_пят* + 1). Именно столько будет и “счастливых” номеров при известных значениях величин *мин_пят* и *макс_пят*.

Это значит, что можно отказаться от перебора всех возможных значений пятой цифры, расчета шестой и проверки всех цифр номера на его “счастливость”, а просто после расчета двух последних величин добавить значение *макс_пят* — *мин_пят* + 1 к предыдущему значению искомой величины *кол*.

Все сказанное выше позволяет сформировать вариант программы, решающей обсуждаемую задачу за минимальное время:

```
алг Число_счастливых_номеров_Вариант_6
нач цел кол, пер, вт, тр, чет, сумма123,
        сумма56,
    мин_чет, мин_пят, макс_чет, макс_пят
кол := 0
нц для пер от 0 до 9
  нц для вт от 0 до 9
    нц для тр от 0 до 9
      сумма123 := пер + вт + тр
    |Уточняем границы значений 4-й цифры
      если сумма123 < 10
        то
          макс_чет := сумма123
        иначе
          макс_чет := 9
      все
      если сумма123 > 18
        то
          мин_чет := сумма123 - 18
        иначе
          мин_чет := 0
      все
      нц для чет от мин_чет до макс_чет
    |Уточняем границы значений 5-й цифры
        сумма56 := сумма123 — чет
        если сумма56 < 10
          то
            макс_пят := сумма56
          иначе
            макс_пят := 9
        все
        если сумма56 > 9
          то
            мин_пят := сумма56 — 9
          иначе
            мин_пят := 0
        все
      |Учитываем все новые "счастливые"
      |номера (см. выше)
       кол := кол + макс_пят —
              мин_пят +  1
    нц
  нц
нц
вывод нс, кол
кон
```

Итак, в результате оптимизации программы мы получили вариант, решающий задачу подсчета количества “счастливых” билетов за время, примерно в четыре раза меньшее, чем первый вариант.

В заключение заметим следующее. Конечно, может возникнуть вопрос: “А какая разница, сколько времени будет выполняться программа — 10 сек. или 3 сек., главное, чтобы результат был!”. Во-первых, существуют задачи, в которых время выполнения программы является важным или даже определяющим фактором. Во-вторых, на олимпиадах по программированию часто ограничивают время решения задачи. В любом случае, как писал великий немецкий математик Карл Гаусс, “Достоинство науки требует, чтобы всемерно совер-

шенствовались средства, ведущие к достижению цели”…

### Задания для самостоятельной работы

1. Оказывается (и интуитивно понятно), что на числовой оси на отрезке от 0 до 999999 “счастливые” шестизначные номера расположены симметрично:

000000 и 999999
001001 и 998998
001010 и 998989
001100 и 998899
…
499976 и 500023
499985 и 500014
499994 и 500005

Это означает, что в приведенном в статье варианте 1 программы можно рассматривать только номера от 0 до 499999, то есть в два раза меньше, но при этом найденное значение *кол* должно быть удвоено. Разработайте соответствующий вариант программы.

2. Разработав любой из приведенных вариантов программы на языке программирования, которым вы владеете, определите количество “счастливых” шестизначных номеров.


3. Сравните время выполнения различных вариантов программ в системах программирования, которые вы применяете (авторы все расчеты проводили в системе программирования PascalABC.net). Для этого используйте процедуры и функции, возвращающие текущее время или время выполнения программы, предусмотренные в системе программирования. В системах программирования Турбо Паскаль и PascalABC.net это, соответственно, процедура GetTime и функция Milliseconds. Так как на продолжительность работы программы, кроме ее особенностей, влияют и другие факторы, для каждого варианта проведите серию расчетов и используйте минимальное значение в серии.

4. Разработав соответствующие программы, определите количество “счастливых” четырех- и восьмизначных номеров.

5. Используя программу Microsoft Excel или подобную, получите графическую иллюстрацию сравнения времени выполнения различных вариантов программ для шестизначных (или не только) номеров билетов.

Результаты присылайте в редакцию.

### Литература

1. Задачи о счастливых билетах. / “В мир информатики” № 191 (“Информатика”, № 10/2013).

## ЗАДАЧНИК

Задача, которую вы решаете, может быть очень скромной, но если она бросает вызов вашей любознательности и если вы решите ее собственными силами, то вы сможете испытать ведущее к открытию напряжение ума и насладиться радостью победы.

*Джордж Пойя*

## Ответы, решения, разъяснения к заданиям, опубликованным в майском выпуске “В мир информатики”

### Кроссворд

*Ответы*

По горизонтали: 2. Папка. 6. Шифр. 7. Бодо. 8. Константа. 10. Тире. 12. Норд. 13. Робот. 14. Дно. 15. Икс. 18. Процесс.

По вертикали: 1. Уши. 2. Принтер. 3. Пи. 4. Абонент. 5. Код. 9. Африка. 11. Абзац. 16. Три. 17. Ось.

*Правильные ответы представили*:

— Абдрахманова Валерия, Гафарова Екатерина, Заляева Карина и Клипперт Илона, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Волошин Марк, Гашимов Геннадий, Дощик Константин, Кармаев Константин, Кратик Роман, Лазуренко Глеб, Постников Владимир и Чунин Павел, средняя школа г. Пионерский Калининградской обл., учитель **Багрова О.А.**;

— Глушаков Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Денисова Елизавета и Долгополова Анастасия, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

— Диков Андрей, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Куракина Мария, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Ломакин Илья, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Рахимова Лейсан, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Рожков Тимур, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

### Головоломка “Лишние слова”

Напомним, что следовало указать, какое слово в каждом наборе не соответствует признаку, по которому сформирован набор.

*Ответы*

1. Иванов. 2. Маленький. 3. Землетрясение. 4. Постепенно. 5. Чешуя. 6. Понимать. 7. Голубой. 8. Сторожка. 9. Волнение. 10. Спокойствие. 11. Сало. 12. Светлый. 13. Дым. 14. Сирень. 15. Вечер. 16. Злой. 17. Чернила.

*Правильные ответы прислали*:

— Абдрахманова Валерия, Гафарова Екатерина, Заляева Карина и Клипперт Илона, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Авдеенко Анна и Зюзин Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Барановская Татьяна и Жукова Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Колесников Денис, в прошлом учебном году — ученик 2-го (!) класса школы № 14 г. Благовещенска Амурской обл., учитель **Рыбалко А.А.** (редакция решила наградить Дениса дипломом);

— Куракина Мария, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Рахимова Лейсан, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

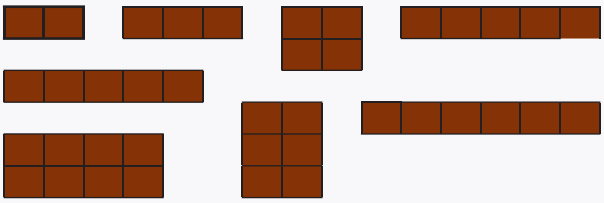
Отметим ответы Валерии Абдрахмановой и Илоны Клипперт, приведших обоснование каждого ответа.

## Задача “Игра «Шоколадка»”

Напомним условие: “Двое по очереди разламывают шоколадку размером 10 × 5 на 50 долек. За ход разрешается сделать прямолинейный разлом любого из имеющихся кусков вдоль углубления. Выигрывает тот, кто первым отломит последнюю дольку. Кто выиграет в эту игру — начинающий ее или делающий ход вторым?”

*Ответ*

Если проанализировать зависимость числа разломов $R$ от количества долек шоколадки $D$ (см. рисунок ниже), то можно установить, что $R = D - 1$.



Следовательно, в шоколадке 10 × 5 придется сделать 49 разломов, то есть, как бы не действовали участники игры, при 50 дольках шоколадки последний, 49-й, разлом всегда сделает тот, кто начал.

*Ответы представили*:

— Волков Владимир и Глушаков Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Диков Андрей, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

Программы решения задач, предложенных для самостоятельной работы в статьях “Решаем задачи о днях недели” и “Диофантовы даты”, прислали:

— Зубов Владислав, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Комарова Мария и Рожков Тимур, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Крысанов Виктор, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Хомутова Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Редакция решила наградить всех их дипломами. Поздравляем!

## В зоопарке

В зоопарке на трех закрытых со всех сторон клетках, в двух из которых находились тигр и леопард, висели таблички с надписями:

1) на первой клетке: “Здесь сидит тигр”;

2) на второй клетке: “Здесь находится леопард”;

3) на третьей клетке: “Тигр сидит во второй клетке”.

Работник зоопарка утверждает, что только одно из этих утверждений является истинным. Не ошибается ли он?

## Три одноклассника

Три одноклассника — Влад, Тимур и Юра, встретились спустя 10 лет после окончания школы. Выяснилось, что один из них стал врачом, другой — физиком, а третий — юристом. Один полюбил туризм, другой — бег, страсть третьего — регби. Юра сказал, что на туризм ему не хватает времени, хотя его сестра — единственный врач в семье, заядлый турист. Врач сказал, что он разделяет увлечение коллеги. Забавно, но у двоих из друзей в названиях их профессий и увлечений не встречается ни одна буква их имен. Определите, кто чем любит заниматься в свободное время и у кого какая профессия.

## Поэт-врун

Поэт Ляпис-Трубецкой говорит правду с полуночи до полудня и лжет в остальное время суток. Ежедневно он сочиняет стихи с 11-00 до 15-00. Сколько часов в сутках, когда он может заявлять: “Сейчас я сочиняю стихи!”?

## На новогоднем балу

Как-то раз четыре товарища (Петя, Боря, Леша и Коля) пошли со своими сестрами (Светой, Наташей, Олей и Леной) на школьный новогодний бал.

Определите, кто чей брат и кто с кем танцевал во время первого танца, если:

а) каждый из ребят не танцевал со своей сестрой;

б) Лена танцевала с Петей, а Света — с братом Наташи;

в) Оля танцевала с братом Светы, Боря — с сестрой Алеши, а Алеша — с сестрой Пети.

## Еще одна задача о домах в Зеленом городе[2]

В Зеленом городе 8200 домов: часть каменных, остальные — деревянные. 2500 домов стоят на правом берегу единственной речки этого города. На карту нанесли 6500 домов (все каменные дома и все дома на левом берегу реки). Что можно узнать о числе домов разного типа на каждом берегу реки?

Разбор аналогичной задачи был проведен в декабрьском выпуске журнала за 2014 год.

## Русские числительные

Французский школьник Жан-Луи изучает русские числительные. Он ищет все двузначные числа, которые записываются двумя словами, начинающимися на одну и ту же букву. Сколько таких чисел?

## Мамед и Самед

Летом Мамед и Самед очень любили есть вишню и черешню. Однажды мама купила бидон вишни (208 штук) и ведерко черешни (320 штук).

Известно, что Мамед ест одну вишенку за 5 секунд, а одну черешенку — за 8 секунд. Самед же одну вишенку ест за 6 секунд, а одну черешенку — за 4 секунды.

1. Какое минимальное время понадобится каждому брату на поедание ягод, если они разделят поровну и вишни, и черешни?

2. За какое минимальное время мальчики могут съесть все эти ягоды?

Брать в рот и есть сразу несколько ягод — неприлично и опасно ☺.

# MICROSOFT EXCEL УГЛУБЛЕННО

## Считаем “счастливые” билеты в среде электронных таблиц

В статье “Еще раз о «счастливых» билетах” в этом выпуске описаны различные методы подсчета числа так называемых “счастливых” билетов. Для решения этой задачи может быть применена также программа Microsoft Excel или аналогичная.

Можно использовать метод, основанный на сравнении сумм цифр левой и правой половин числа — номера билета. Проиллюстрируем его на примере четырехзначных номеров. Верхняя часть соответствующего листа показана на *рис.* 1.

В столбце A получим все номера билетов (от 0 до 9999), в столбцах B–E — их отдельные цифры, в столбцах F и G — суммы цифр, в столбце H — результат сравнения сумм цифр левой и правой половин номера. Искомое значение выведем в ячейке F1.

Теперь о том — как. Конечно, сделаем так, чтобы вручную вводить как можно меньше формул, ведь строк с числами — 10 000! (Последний символ — это не знак факториала ☺.)

О том, как получить большое число порядковых номеров, рассказано в статье [1]. Напомним один из возможных способов, наиболее подходящий в нашем случае (без выделения больших диапазонов ячеек), — использование прогрессии.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **ЧИСЛО «СЧАСТЛИВЫХ» БИЛЕТОВ** | | | | | | | |
| 2 | **Номер** | **Цифра 1** | **Цифра 2** | **Цифра 3** | **Цифра 4** | **Сумма12** | **Сумма34** | **Сравнение** |
| 3 | 0 | | | | | | | |
| 4 | 1 | | | | | | | |
| 5 | 2 | | | | | | | |
| 6 | 3 | | | | | | | |
| … | | | | | | | | |

*Рис.* 1

[2] Задачу предложила Е.А. Мирончик, учитель информатики лицея № 111 г. Новокузнецка Кемеровской обл.

В ячейке A3 наберем значение 0 (начальное) и введем его, оставшись в этой же ячейке (для этого после набора нужно нажать не на клавишу Enter, а щелкнуть мышью на кнопке с “галочкой” на панели инструментов, расположенной справа от адреса ячейки. Можно, конечно, после нажатия на клавишу Enter вернуть указатель активной ячейки на A3). Далее нужно выбрать вкладку **Главная**, в группе **Редактирование** щелкнуть на пиктограмме **Заполнить** (  ) и выбрать `Прогрессия`.

В появившемся диалоговом окне **Прогрессия** (*рис.* 2):

— в поле `Предельное значение` указать значение 9999;

— в группе радиокнопок `Расположение` указать значение `по столбцам`;

— нажать кнопку OK.

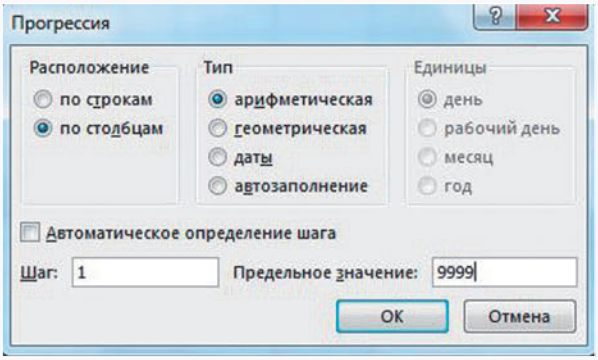



*Рис.* 2

Так как нам придется работать с последней заполненной строкой (10002), то нужно как-то пометить ее (для чего — станет понятно чуть позже). Перейти к последней заполненной строке можно, нажав комбинацию клавиш Ctrl + Shift + ↓ (эта комбинация выделяет все ячейки до первой пустой ячейки в указанную стрелкой сторону в данном столбце). Находясь в столбце A, сделаем это — окажемся на строке 10002 и на этой строке в столбцах B–H поставим какие-то “пометки”, например, букву *п* (от ***п****оследняя*) — см. *рис.* 3. Вернемся наверх, нажав комбинацию клавиш Ctrl + Home.

Теперь можно вводить необходимые формулы.

Для выделения отдельных цифр числа следует использовать функции ЦЕЛОЕ и ОСТАТ. Первая возвращает целую часть своего аргумента, вторая — остаток от деления первого аргумента на второй.

Итак, формулы:

— в ячейке B3: =ЦЕЛОЕ(A3/1000);

— в ячейке C3: =ОСТАТ(ЦЕЛОЕ(A3/100);10);

— в ячейке D3: =ОСТАТ(ЦЕЛОЕ(A3/10);10);

— в ячейке E3: =ОСТАТ(A3;10);

— в ячейке F3: =B3+C3;

— в ячейке G3: =D3+E3;

— в ячейке H3: =ЕСЛИ(F3=G3;1;0). (Если суммы цифр равны, то записываем 1, иначе — 0.)

А в остальных ячейках? Как и обещали, заполним и их, так сказать, “с минимальными затратами труда”.

Покажем все на примере формулы в ячейке B3:

1) копируем ее в буфер обмена;

2) выделяем все ячейки ниже нее до ячейки B10002, нажав Ctrl + Shift + ↓;

3) вставляем во все выделенные ячейки формулу из буфера обмена.

Остальные формулы копируются аналогично (можно также скопировать и сразу все ячейки из диапазона B3:H3).

После этого осталось только получить искомое значение количества “счастливых” билетов с четырехзначными номерами. Для этого в ячейку F1 следует ввести формулу =СУММ(H3:H10002).

## Задание для самостоятельной работы

1. Заполнив лист по описанной методике, найдите количество “счастливых” билетов с четырехзначными номерами.

2. Решите эту же задачу для шестизначных номеров.

**Примечание.** Можно учесть также особенность, указанную в задании 1 для самостоятельной работы в статье [1] (“счастливые” номера на числовой оси расположены симметрично).

Результаты присылайте в редакцию.

## Литература

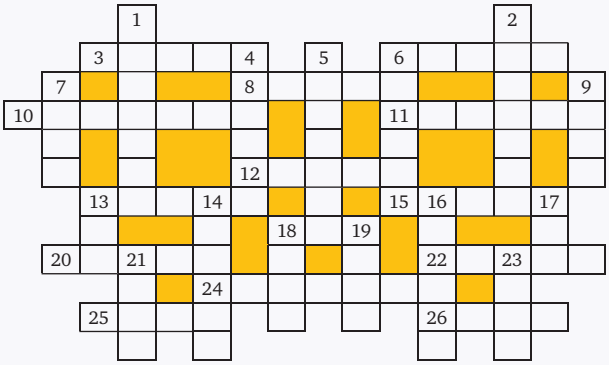
1. Заполнение ячеек порядковыми номерами. / “Информатика”, № 7–8/2015.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **ЧИСЛО «СЧАСТЛИВЫХ» БИЛЕТОВ** | | | | | | | |
| 2 | **Номер** | **Цифра 1** | **Цифра 2** | **Цифра 3** | **Цифра 4** | **Сумма12** | **Сумма34** | **Сравнение** |
| … | | | | | | | | |
| 10001 | 9998 | | | | | | | |
| 10002 | 9999 | п | п | п | п | п | п | п |
| 10003 | | | | | | | | |

*Рис.* 3

## Кроссворд

Решите, пожалуйста, кроссворд.



*По горизонтали*

3. Канал для передачи информации, а также инструмент графического редактора.

6. Второй экземпляр (файла и др.).

8. Величина, с помощью которой осуществляется счет.

10. Арифметическая операция.

11. Инструмент графического редактора.

12. Один из необходимых элементов блок-схемы алгоритма.

13. Поименованная группа файлов, объединенных по какому-то признаку.

15. Буква греческого алфавита.

18. Цифра шестнадцатеричной системы счисления.

20. Пропущенное слово в предложении “Ключ — условие, по которому проводится … записей в базе данных”.

21. Системный …

22. Овощное блюдо, которое любят программисты.

24. Фантастический роман польского писателя Станислава Лема, по которому режиссером Андреем Тарковским снят кинофильм.

25. Язык программирования для начинающих (и не только).

26. Популярный вид компьютерных программ.

*По вертикали*

1. Часть окна текстового редактора, используемая для установки полей, отступов и т.п.

2. Распечатка текста программы.

4. Элемент электронной таблицы.

5. В программировании — преднамеренное прерывание выполнения программы, при котором выполняется вызов отладчика (используется также понятие “точка …”).

6. Тип топологии локальной сети, а также “круг без внутреннего круга”.

7. Световое … (устройство ввода информации).

9. Жаргонное название результата обработки изображения на устройстве ввода информации в компьютер.

13. Положение в компьютерной игре, имитирующей древнеиндийскую игру.

14. Указатель места на экране.

16. В программировании — совокупность значений одного типа, имеющих общее имя.

17. Язык программирования, названный в честь первой женщины — программиста.

18. Так иногда называют удельный вес, отношение части к целому.

19. В парапсихологии: то же, что биополе.

23. Валюта, в которой получали зарплату итальянские программисты, а также музыкальный инструмент.

## Чему равно АС?

Если число **АС** больше числа **С** на 160, а число **СЕВА** больше числа **ЕВА** на 49 152, то чему равно число **АС**? Одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры. Все известные числа в условии — десятичные.

## Потомки

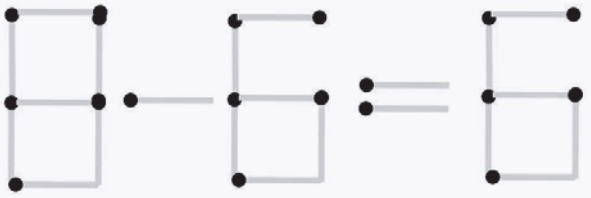
Среди потомков Петра Ивановича по мужской линии (сыновья, сыновья сыновей и т.д.) три Петра и пять Ивановичей. При каком наименьшем числе потомков это возможно? (Имена двух любых родных братьев, естественно, различны.)

## Кубик

На гранях кубика записано по одной цифре от 1 до 6. Если на этот кубик смотреть с одной стороны, то видны 1, 2 и 3, с другой стороны видны 4, 1 и 5, а с третьей стороны видны 2, 6 и 4. Какие цифры находятся с противоположных сторон цифр 1 и 2?

## Переложить спичку

Как, переложив одну спичку, получить верное равенство?



Задание предназначено для учащихся 1–7-х классов.

## ЯПОНСКИЙ УГОЛОК

友

### Судоку обычные…

Решите, пожалуйста, две обычные японские головоломки “судоку”:

1) простую:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 8 | 5 | 1 | | 9 | 6 | | | |
| | 2 | | | 8 | | | 6 | |
| 6 | | | 5 | | 4 | 2 | | 8 |
| | 3 | 7 | | 6 | 9 | | | |
| | | 4 | | 3 | | 9 | | |
| | | | 1 | 2 | | 4 | 5 | |
| 3 | | 8 | 6 | | 1 | | | 4 |
| | 4 | | | 5 | | | 3 | |
| | | | 3 | 4 | | 7 | 8 | 9 |

2) сложную:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 3 | | | 9 | | 7 |
| 1 | | | 9 | | | 4 | 3 | 8 |
| | 7 | | 4 | | | | 5 | |
| | 3 | 7 | | | 1 | 6 | 2 | |
| | | | | 9 | | | | |
| | 8 | | | | 4 | | 9 | 1 |
| 4 | | | | 8 | | | | |
| 6 | | | 2 | 4 | | | 7 | |
| | 5 | | | | | 3 | | |

### И не совсем…

Здесь задание усложнено: основные диагонали большого квадрата также должны содержать цифры от 1 до 9.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 3 | | 8 | 4 | | | | |
| | | | 9 | | | | | |
| | | 5 | | | | | | |
| 2 | 5 | | | | 7 | 4 | 8 | |
| | | 1 | | | | | 3 | |
| | 7 | 3 | | | | | | 1 |
| | 4 | | | | | | | |
| | | 8 | 6 | | | 9 | | |
| 9 | | | | | | | | |

Решения (можно не все) присылайте в редакцию.

## КРЕПКИЙ ОРЕШЕК

Напомним, что в данной рубрике редакция приводит разбор задач, решение которых вызвало трудности.

### Числовой ребус “КАПЛЯ + КАПЛЯ + КАПЛЯ = ОЗЕРКО”

Приведем начало решения.

*Решение*

Запишем ребус “в столбик”:

| | | | | | |
|---|---|---|---|---|---|
| | **К** | **А** | **П** | **Л** | **Я** |
| + | **К** | **А** | **П** | **Л** | **Я** |
| | **К** | **А** | **П** | **Л** | **Я** |
| **О** | **З** | **Е** | **Р** | **К** | **О** |

Видно, что **О** = 1 (при этом **Я** = 7) или **О** = 2 (при этом **Я** = 4).

Исследуем эти два случая.

1. **О** = 1, **Я** = 7

| | | | | | |
|---|---|---|---|---|---|
| | К | А | П | Л | 7 |
| + | К | А | П | Л | 7 |
| | К | А | П | Л | 7 |
| 1 | З | Е | Р | К | 1 |

Так как 10 < 3**К** < 20, то с учетом возможного переноса “в уме” из разряда тысяч возможные значения цифры **К**: 3, 4, 5 и 6. Кроме того, на цифру **К** должна оканчиваться сумма 3**Л** + 2 (см. разряд десятков). Анализ всех возможных значений цифры **К** и соответствующих значений цифры **Л** приведен в таблице:

| **К** | **К** – 2 | **Л** | **Вывод** |
|---|---|---|---|
| 3 | 1 | 7 | Невозможно, т.к. **Я** = 7 |
| 4 | 2 | 4 | Невозможно, т.к. **К** = **Л** |
| 5 | 3 | 1 | Невозможно, т.к. **О** = 1 |
| 6 | 4 | 8 | Возможно |

Запишем данные последнего варианта в ребус:

| | | | | | |
|---|---|---|---|---|---|
| | 6 | **А** | **П** | 8 | 7 |
| + | 6 | **А** | **П** | 8 | 7 |
| | 6 | **А** | **П** | 8 | 7 |
| 1 | **З** | **Е** | **Р** | 6 | 1 |

При этом **З** = 9 (“в уме” из разряда тысяч переходит 1).

Оставшиеся “неиспользованными” цифры применительно к значению П и соответствующему значению Р проанализированы в таблице:

| **П** | **Р** | **Вывод** |
|---|---|---|
| 0 | 2 | Возможно |
| 2 | 8 | Невозможно, т.к. **Л** = 8 |
| 3 | 1 | Невозможно, т.к. **О** = 1 |
| 4 | 4 | Невозможно, т.к. **П** = **Р** |
| 5 | 7 | Невозможно, т.к. **Я** = 7 |

Анализ единственного возможного варианта показывает, что он не может быть реализован оставшимися значениями цифр для букв **Л** и **Е**.

Значит, **О** = 2, **Я** = 4. Полный анализ этого варианта проведите самостоятельно и пришлите ответ в редакцию.

Редакция благодарит читателей, приславших правильный ответ:

— Донникову Анну, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Чернову Ксению, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Якименко Степана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

## ЭТО ПОЛЕЗНО ЗНАТЬ

### Компьютер и осанка

*“Не сутулься!” — кому из подростков родители и учителя не говорили такие слова! Но мало кто прислушивается к словам старших и следит за своей осанкой. По оценкам медиков до 80% детей и подростков имеют проблемы со спиной. И не последнюю роль в этом играют компьютеры. Хотя правильнее сказать: не компьютеры, а длительное неподвижное времяпрепровождение перед ними.*

**Человек прямоходящий.** Нашим далеким предкам понадобилась не одна тысяча лет, чтобы выпрямиться в полный рост. Сегодняшние школьники теряют навык прямохождения, словно пытаясь вернуться в первобытные времена. Конечно, на четвереньки мы еще не опустились, но сгорбленная спина, сутулость, опущенные плечи и голова — очень распространенные явления. Когда мы встаем из-за стола, большинство из нас не распрямляются до конца, причем не только в области грудного отдела, но и тазобедренного. В результате даже у стоящего человека таз оказывается отставлен назад, а ноги слегка согнуты в коленях и повернуты внутрь.

Правильная осанка появляется не сразу, человек рождается с прямой спиной без всяких изгибов. Изгибы появляются уже в процессе роста и эксплуатации позвоночника. Правильные они будут или нет — зависит от человека.

Между тем позвоночник — это наш стержень, который позволяет ходить прямо, но часто не выполняет своих функций из-за неправильного образа жизни. Когда какой-то его отдел блокируется, то нагрузка на другие его отделы распределяется неравномерно.

**Согбенные работой.** Казалось бы, за компьютером сидишь, а не стоишь, позвоночник не нагружаешь. Откуда тогда берется сутулость?

Во-первых, причина в неправильном расположении компьютера, которое провоцирует неправильное сидение за ним. Человека словно перекашивает, нагрузка на мышцы оказывается неравномерной.

Вторая причина — слабость мышц позвоночника. Если человек не приучен держать спину ровно, то и в сидячем положении мышцам нужна будет опора и поддержка.

И третья причина — длительное нахождение в одном и том же положении без движения. Даже при правильной осанке подросток не сможет держать спину прямо в течение нескольких часов. Это, кстати, объясняет, почему в школе дети резко начинают сутулиться. При долгом однообразном положении и отсутствии движения мышцы устают — не меньше, чем при интенсивной активности, и перестают выполнять поддерживающую функцию. Так развиваются остеохондроз и сколиоз.

**Ровная спина.** Итак, что же такое правильная осанка? Если говорить упрощенно, то уши должны располагаться точно в плоскости плеч, а плечи — точно над бедрами. При этом плечи находятся на одной горизонтальной линии, одно плечо не может быть выше другого. При опущенных руках локти должны попадать точно в изгиб талии; если они выше талии — вы слишком поднимаете плечи; если ниже или торчат в стороны — скорее всего вы сутулитесь. Ребра не должны слишком выдаваться вперед, находясь точно над подвздошными костями.

**Зачем нужна хорошая осанка?** Хорошая осанка полезна для вашего здоровья по разным причинам. Стоя и сидя правильно, вы препятствуете деформации мышечной ткани и предотвращаете боль в мышцах спины, шеи. Хорошая осанка также помогает дышать глубоко — животом, а не поверхностно.

Физических преимуществ много, но есть и другие преимущества. Когда вы стоите правильно, тело как бы выравнивается, вы выглядите стройнее и в целом лучше. А благодаря этому вы и чувствуете себя увереннее.

**Хорошо сидим!** В первую очередь проблемы начинаются из-за того, что человек слишком долго находится за компьютером. Вообще любая фиксированная поза в течение продолжительного времени вредна для организма, так как ведет к застою крови в органах и затеканию мышц.

Если же, сидя за компьютером, вы еще и сутулитесь, то нагрузка на позвоночник, плечевые сухожилия и мышцы плеча увеличивается, а мышцы растягиваются. Сгорбленное положение может стать причиной грыжи межпозвоночных дисков поясничного и шейного отделов и синдрома запястного канала (о нем мы расскажем в следующих номерах).

Если монитор расположен слишком далеко от глаз, то приходится вытягивать шею вперед. В итоге увеличивается нагрузка на мышцы основания головы и шеи, сдавливаются сосуды шеи, ухудшая

кровоснабжение головы. Если при этом человек работает с бумажным документом, то ему приходится периодически откидывать голову назад, чтобы заглянуть в книгу или тетрадь. Это усиливает прогиб шейного отдела позвоночника. Впоследствии это может стать причиной головной боли и боли в руках, поскольку нервы, отходящие от спинного мозга в области шеи, протягиваются до кончиков пальцев.

**В правильном положении.** Итак, чтобы всех этих неприятностей не происходило, посмотрим, как оборудовано компьютерное место для школьника и какие ошибки при этом чаще всего допускаем.

*Ошибка 1.* Компьютерный столик.

В большинстве случаев он не очень годится для работы: ведь на его столешнице практически не остается места, чтобы положить руки. Значит, локти будут постоянно свисать, а опираться о стол и, таким образом, поддерживать спину будут запястья. Долго они не выдержат, и спина постепенно начнет скрючиваться.

Поэтому компьютерный стол должен быть достаточно широким, на котором места хватит и монитору, и тетрадкам с учебниками, и локтям.

*Ошибка 2.* Стул обычный.

Если компьютерный стол многие считают необходимым, то компьютерному креслу отводят второстепенную роль, полагая, что вполне сойдет и обычный стул. И это в корне неверно. Специальные компьютерные стулья имеют анатомическую конфигурацию и повторяют все изгибы спины. При выборе следует обращать внимание на следующие параметры.

Жесткость сиденья — она должна быть средняя. Стул должен регулироваться по высоте и степени наклона. Иметь широкие подлокотники.

*Ошибка 3.* Монитор в дальнем углу.

Поскольку у школьников функции компьютерного и письменного стола выполняет, как правило, один и тот же стол, то обычно монитор устанавливается в самом дальнем углу столешницы, вплотную к стене так, чтобы расчистить место под тетради и учебники. В результате как раз и происходит то, о чем мы писали выше. Вы вытягиваете шею, чтобы увидеть изображение на мониторе, затем отклоняетесь назад, заставляя мышцы плечевого пояса беспрестанно напрягаться.

Поэтому монитор должен находиться примерно в 50 см от глаз. Чем больше монитор, тем больше должно быть расстояние.

*Ошибка 4.* Выбираем ноутбук.

Он и места занимает чуть-чуть, и кажется более современным гаджетом. Но медики уверены: с точки зрения здоровой осанки это устройство — самое неполезное. Поскольку при работе за ним нарушаются абсолютно все правила: и руки на весу, и ноги поджаты, и спина скрючена.

Если все же у вас есть ноутбук или планшетный компьютер, то дома у него должно быть свое стационарное место согласно всем гигиеническим требованиям.

**Правильная поза за компьютером.** Сев за компьютер, мысленно проведите линию позади спины: на одной прямой должны оказаться затылок, лопатки и копчик. Плечи можно расслабить — не стоит сидеть, словно аршин проглотили.

Руки согнуть под прямым углом и положить на клавиатуру, причем предплечья (часть руки от запястья до локтя) на две трети должны лежать на столе.

Подлокотники на кресле необходимо отрегулировать таким образом, чтобы они не мешали работе и оставляли плечи расслабленными.

Сидеть нужно, занимая сиденье почти целиком, а не на его краешке.

Ступни при этом находятся полностью на полу. Если сидеть, закинув ногу на ногу или подложив одну под другую, то происходит перераспределение нагрузки и постепенное смещение позвонков.

Правильная осанка дается непросто. С непривычки могут начать побаливать спина, область шеи и плеч. Это естественно — ведь мышцы еще не привыкли к такому положению. Со временем, если все время сидеть правильно, неприятные ощущения пройдут.

**Гимнастика за компьютером.** Не надо сидеть за компьютером часами, не вставая. Каким бы важным делом вы ни занимались, каждые полчаса обязательно делайте перерыв. Во время перерыва очень полезно проделать несложные упражнения для спины.

- Выполняется стоя. Положить руки на пояс, медленно прогнуться назад. Вернуться в исходное положение.
- Выполняется стоя или сидя. Повороты головы: сначала в одну сторону, затем в другую.
- Затем наклоны головой: вперед — назад. При наклоне вперед постараться коснуться подбородком груди. Затем наклоны наискосок — к левому плечу — к правому плечу. Тянуться сильнее, чтобы почувствовать, как тянутся мышцы шеи с противоположной стороны.
- Сцепить руки за спиной в замок и потянуться.
- Выполняется сидя или стоя. Вытянуть руки, сцепленные в замок, перед собой и потянуться, прогнув спину и втянув живот. Затем, не расцепляя замок, поднять руки вверх и потянуться, чуть отводя их назад.

Эти упражнения прекрасно помогают снять напряжение с затекших мышц.

И, самое главное, за учебой и работой не забывать, что залог правильной осанки — движение. Ведь организму нужна постоянная и разнообразная физическая нагрузка.

**Упражнения для осанки.** Для профилактики неправильной осанки и ее исправления следует 2–3 раза в неделю делать небольшой комплекс упражнений.

• Встать у стены так, чтобы к стене плотно прижимались затылок, лопатки, ягодицы и пятки.

• Сохраняя такую позицию, поочередно поднимать ноги, сгибая в коленях и пытаясь их подтянуть к груди.

• Не “отлепляясь” от стены, делать боковые наклоны.

• Все так же плотно прижимаясь к стене, присесть и встать.

Каждое упражнение следует проделывать до 10 раз.

Некоторые упражнения выполняются лежа на жесткой поверхности — в идеале на полу.

• Из положения лежа на спине поочередно поднять и опустить прямые ноги.

• Также лежа на спине, крутить “велосипед”.

• Лечь на живот для выполнения упражнения “рыбка”: отвести руки назад и приподнять ноги, прогнув спину.

*По материалам журнала “Потенциал”*
*(автор статьи — Е.В. Бабичева,*
*редактор журнала “Здоровье”)*

## ВНИМАНИЕ! КОНКУРС!

### Конкурс № 113

В качестве заданий этого конкурса предлагаем выполнить задания, предложенные для самостоятельной работы в статьях “Еще раз о «счастливых» билетах” и “Считаем «счастливые» билеты в среде электронных таблиц”.

Ответы (можно не ко всем заданиям) отправьте в редакцию до 15 ноября по адресу: 121165, Москва, ул. Киевская, д. 24, “Первое сентября”, “Информатика” или по электронной почте: *vmi@1september.ru*.

## ЦИФРОВОЙ МИР

### Люди водят машину хуже, чем компьютерный алгоритм

В конце 2014 года фирма Google представила первый полноценный прототип управляемого компьютером электромобиля, информация о разработке и тестировании которого публиковалась с весны 2014 года.

Он имел все нужные компоненты, которые необходимы самоуправляемому автомобилю для беспрепятственного “выхода” на дороги общего пользования. Впрочем, с момента презентации первой версии автономного автомобиля в его внешнем виде мало что изменилось. Наиболее существенным отличием прототипа самоуправляемого автомобиля Google была лазерная система LIDAR на крыше, которая является “глазами” транспортного средства. Ее размеры значительно уменьшились, система преобразилась в нечто наподобие полицейской “мигалки”.

Начиная с анонса самоуправляемого автомобиля, инженеры Google работали с несколькими прототипами с целью тестирования отдельных компонентов и систем, включая тормоза или датчики. Все наработки, полученные в ходе тестирования отдельных компонентов, были использованы для создания рабочего прототипа.

Тестовые поездки автомобиля стартовали в новогодние праздники, а в начале 2015 года он вышел на дороги общего пользования. За рулем авто на всякий случай находится водитель, который может взять на себя управление в случае возникновения чрезвычайной ситуации. Но генеральный директор фирмы-разработчика Tesla Motors пообещал, что в течение пяти лет в его электромобилях появится система “полного автопилота”.

Летом этого года фирма Google сообщила, что за июнь 2015 года ее автономные автомобили попадали в аварию два раза, и оба ДТП произошли по вине человека.

В общей же сложности за шесть лет с момента выхода первого автономного электромобиля Google на дороги беспилотный транспорт попал в аварии 14 раз. Но все инциденты были спровоцированы так называемым “человеческим фактором” — невнимательностью водителей-людей. Ни единой ошибки компьютерной системы, которая могла бы послужить причиной ДТП, за время испытаний не зафиксировано.

В большинстве случаев в беспилотные автомобили на светофорах и шоссе врезались другие участники движения. В четырех случаях вина за аварию лежит на водителях беспилотных электромобилей Google, которые брали управление в свои руки.

За все время испытаний машины Google на автопилоте преодолели более 2,7 млн км. В происшествиях, зарегистрированных с момента выхода на дорогу первого автономного электромобиля, не пострадал ни один человек. Сами машины получили незначительные повреждения.

Педагогический университет
**«Первое сентября»**

Лицензия Департамента образования г. Москвы 77 01 № 0007183, рег. № 036377 от 23.07.2015

# ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

(с учетом требований ФГОС)

С 1 ноября ведется прием заявок на второй поток 2015/16 учебного года

## образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – 108 УЧЕБНЫХ ЧАСОВ

  Стоимость – 4990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – 72 УЧЕБНЫХ ЧАСА

  Стоимость – от 3990 руб.

По окончании выдается удостоверение о повышении квалификации установленного образца

Перечень курсов и подробности – на сайте edu.1september.ru

Пожалуйста, обратите внимание:

заявки на обучение подаются только из Личного кабинета, который можно открыть на любом сайте портала www.1september.ru

журнал

# Информатика – Первое сентября

## 1-е полугодие 2016 года

## ПОДПИСКА

на сайте www.1september.ru и в почтовых отделениях РФ


| Индекс | Название издания | Периодичн. в полугодие | 1 месяц | | 6 месяцев | |
|---|---|---|---|---|---|---|
| | | | Каталожная цена (руб.) | Подписная цена (руб.) | Каталожная цена (руб.) | Подписная цена (руб.) |
| Название блока в разделе «Журналы» | **ПЕРВОЕ СЕНТЯБРЯ. ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА** (499)249-31-38 | | | | | |
| 79066 | **Информатика – Первое сентября. Бумажная версия** *В июне не выходит. Подписка на июнь не принимается* *(-) 160 г 64 стр.* | 5 | 440.00 | | 2200.00 | |
| 12684 | **Информатика – Первое сентября. Электронная версия на CD (полная копия бумажной версии)** *В июне не выходит. Подписка на июнь не принимается* (-) *75* г | 5 | 160.00 | | 800.00 | |
| сайт 1september.ru | **Информатика – Первое сентября. Электронная версия** | 5 | – | | – | 500.00 |

Подписку принимают во всех отделениях связи Российской Федерации, а также на сайте www.1september.ru

При подключении школы к проекту «Школа цифрового века» (см. digital.1september.ru) каждый учитель получает доступ ко всем журналам Издательского дома «Первое сентября». Стоимость подключения школы на год – 6 тыс. рублей независимо от количества учителей

При оформлении подписки на сайте оплата производится по квитанции в отделении банка или электронными платежами on-line

VISA MasterCard QIWI WebMoney