# ЗАОЧНАЯ ШКОЛА СОВРЕМЕННОГО ПРОГРАММИРОВАНИЯ

## НАПУТСТВИЕ УЧАЩИМСЯ

*Сейчас наше школьное программистское образование стоит перед трудным выбором. Компьютеры стали настолько важны для обычных пользователей, а их использование стало настолько простым, что они приблизились к обычному бытовому наукоемкому оборудованию. Подобно тому, как нам не требуется знать подробностей устройства телевизора, утюга, водяного фильтра, фотоаппарата, мы можем не знать сколько-нибудь глубоко и подробностей устройства компьютера и его программ.*

*Отсюда делается вывод, что и учить этому не обязательно. Достаточно учить пользоваться готовыми пакетами и, в первую очередь, конечно же, продукцией воспеваемого газетами Microsoft'а. Такая точка зрения была ясно высказана еще знаменитым недорослем в пьесе Д.Фонвизина. Помните, что он говорил? "Зачем учить географию, если есть извозчики".*

*В использовании компьютеров помощью "извозчиков" не обойтись. И в отличие от других технических достижений, входящих в быт, здесь недостаточно иметь сколько-то техников и инженеров. Прогресс вычислительных средств повлек за собой невиданное изменение других наук и отраслей промышленности (той же географии). Открылся громадный рынок интересного и, кстати, хорошо оплачиваемого труда, — хорошие программисты всем нужны. Можно сказать без преувеличения, что тот, кто научился программировать, обеспечил себя занятием на всю жизнь. Занятием интересным, занятием с громадным выбором областей приложения, — в технике, в гуманитарных науках, в искусстве, в медицине. Нужно отметить, что программирование хорошо согласуется с лучшими традициями русского и советского образования. Не случайно так много наших способных программистов находят себе работу в заграничных фирмах.*

*Создаваемая школа собирается привлечь ребят к программированию как интересной области творческой деятельности, показать им, хотя бы на некоторых примерах, замечательный мир, спрятанный внутри "простой железной коробки".*

*Мир программирования влияет и на развитие математики и на состояние школьной математики. Я помню, как много времени на уроках математики в мои школьные годы уделялось теме "Приведение тригонометрических выражений к виду, удобному для логарифмирования". Уже тогда было непонятно, зачем это все знать каждому школьнику (много позже я узнал, что появление темы относится к далекому прошлому, когда люди измеряли Землю и не имели почти никаких вычислительных средств за исключением хорошей таблицы логарифмов). Но теперь-то зачем нужно рассказывать это всем? Потребности в математике изменились. И про эти изменения в нашей школе также расскажут.*

***Романовский Иосиф Владимирович, доктор физ.-мат. наук, профессор СПбГУ, в течение многих лет - член жюри городских и Всероссийских олимпиад по информатике.***

# ПАМЯТКА УЧАЩЕМУСЯ

Обучение искусству программирования требует изучения таких дисциплин, как логика, дискретная математика, прикладная теория алгоритмов и др., которым в школьном курсе уделяется совсем немного времени, а во многих школах их не изучают вовсе. Школа современного программирования ставит своей целью восполнить этот пробел, а также приобщить школьников, интересующихся профессией программиста, к решению содержательных задач по программированию. Предполагается, что в школе будут обучаться как те ребята, которые владеют теми или иными языками программирования, так и те, которые не имеют о них пока никакого представления.

В соответствии с этим, в школу принимаются ребята 5-11 классов, а предлагаемые задачи имеют два уровня (***Уровень 1*** не требует умения программировать). Каждый обучающийся может выбрать тот уровень, который окажется ему по силам. В процессе обучения нужно будет решать много задач, различных по трудности. Задачи в своем большинстве будут носить исследовательский характер. Все подписчики журнала найдут условия задач в очередном номере журнала, начиная с первого. Тем, кто не является подписчиком, задания будут высылаться по электронной или обычной почте. На решение предлагаемых задач отводится один месяц (до опубликования и разбора решений в следующем номере журнала). Решения будут приниматься на дискетах и в письменном виде (по адресу: 191025, Санкт-Петербург, Марата 25, Заочная школа современного программирования, телефон для справок (812) 164-13-55) и по электронной почте (school@aec.neva.ru). Обращаем внимание, что участниками школы могут быть как отдельные школьники, так и коллективы (классы, кружки и пр.). Мы будем благодарны всем, кто сообщит нам о проблемах и предложениях, возникших в процессе выполнения заданий и пересылки решений.

## УСЛОВИЯ, КОТОРЫМ ДОЛЖНЫ УДОВЛЕТВОРЯТЬ ПРИСЫЛАЕМЫЕ РЕШЕНИЯ.

Программы (***Уровень 2***) проверяются автоматической системой тестирования, поэтому необходимо точно следовать указанным в условиях форматам ввода/вывода, а также указанным ниже правилам.

Все программы должны быть скомпилированы под DOS и должны использовать стандартный ввод/вывод. Для Паскаля это процедуры read/readln и write/writeln, для Си - функции scanf и printf, для Си++ - объекты cin и cout, для Бейсика - операторы INPUT и PRINT. Нельзя использовать модули или библиотеки, которые перенаправляют стандартный ввод/вывод (например, для Паскаля - модуль CRT). При одинаковых входных данных программа должна выдавать одинаковый результат.

Названия исполнимых файлов с программой:

<первые три буквы фамилии>_<задача>.exe

*Например*, pet_2.exe, sid_3.exe.

Все тексты должны быть набраны в формате ASCII.

Тексты программ будут читаться проверяющими, поэтому следует использовать поясняющие комментарии и уделять внимание структуре программы, а также названиям идентификаторов.

*Примечание:* те, кто не может прислать решения в электронном виде, присылают подробно и аккуратно оформленные решения на бумаге.

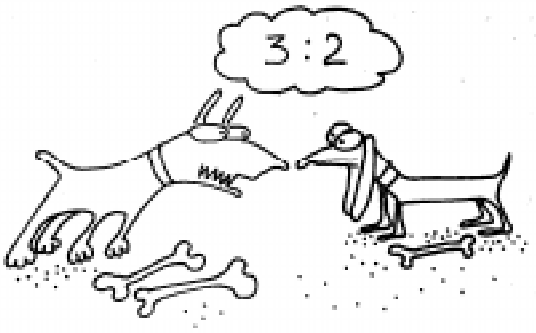
*Петров Виктор Александрович*
*Поздняков Сергей Николаевич*

# ЗАНЯТИЕ 1. АЛГОРИТМЫ НАД ЦЕЛЫМИ ЧИСЛАМИ

С первого класса учителя “программируют” своих учеников на выполнение алгоритмов с целыми числами. Сначала это алгоритмы “поразрядного” сложения и вычитания, потом алгоритмы умножения “столбиком” и деления “уголком”, потом алгоритмы перевода дробей в десятичную запись и обратно. (Были даже времена, когда ученики выполняли алгоритм извлечения квадратного корня).

Алгоритмы, которыми владеют люди, вполне удовлетворяют их при решении бытовых задач, например, определения стоимости покупки или подсчета квартплаты. В то же время вряд ли человек справится с делением 100-значного числа на 50-значное: не очевидно также, что привычный способ записи чисел является лучшим для всех возникающих задач. Тем не менее, владения вышеперечисленными алгоритмами уже достаточно, чтобы объяснить несколько важных идей компьютерной математики, которые используются при программировании операций с целыми числами.


## ДЕЛИМОСТЬ




Распространенной задачей является такая: определить, делится ли одно число на другое без остатка. Например, число 73963 на 1999. Применяем алгоритм деления “уголком”:

$$\begin{array}{r|l} \dot{7}\dot{3}963 & 1999 \\ \underline{5997\phantom{0}} & \underline{37\phantom{0}} \\ 13993 & \\ \underline{13993} & \\ 0 & \end{array}$$

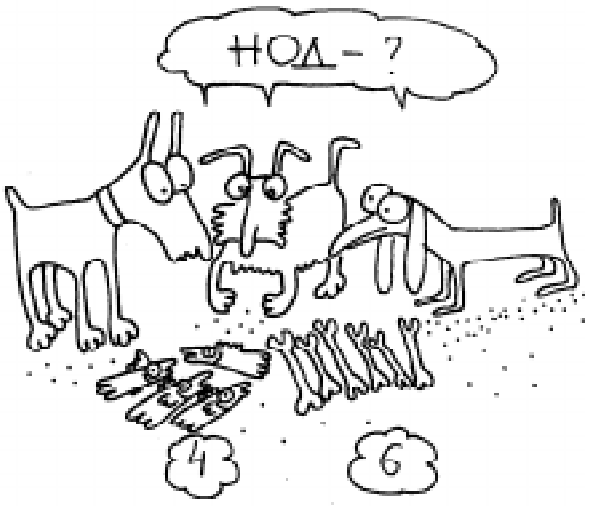
и получаем частное 37 и остаток 0. Делаем заключение о делимости. Если требуется проверка результата, то умножаем частное 37 на делитель 1999 (используя алгоритм умножения “столбиком”) и убеждаемся, что результат равен делимому 73963.

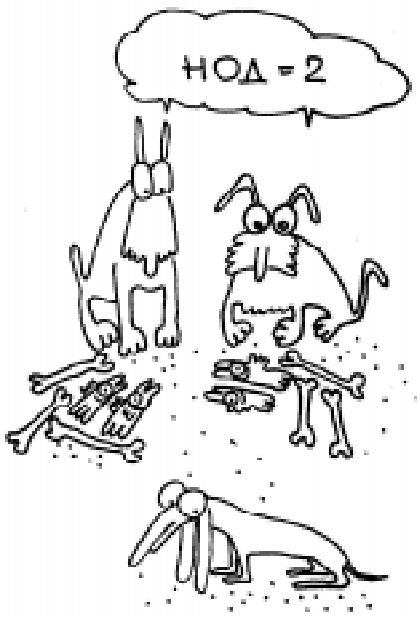
Теперь нетрудно понять, почему делимость определяется через операцию умножения.

**Определение.** Говорят, что число $a$ делится на число $b$ (или что $a$ кратно $b$), если можно подобрать такое число $c$, чтобы выполнялось равенство $a=bc$.

($a$, $b$, $c$ – целые, $b\neq 0$)

Одна из часто встречающихся задач состоит в том, чтобы узнать, имеет ли данный набор чисел общий делитель, отличный от 1 (как говорят, нетривиальный делитель, так как 1 является делителем лю-




бого набора чисел). Числа, которые не имеют нетривиального общего делителя, называются *взаимно простыми.*

Обычно сразу ищут наибольший общий делитель (НОД), который делится на все остальные делители и, таким образом, содержит их "в себе".

*Например*, набор чисел 72, 84, 132, 144 имеет общие делители 2, 3, 4, 6, 12. Наибольший общий делитель равен 12 (пишется НОД=12). Он делится на все общие делители.

Некоторые из вас скажут: "мы решали эту задачу в школе, мы знаем что делать: нужно разложить каждое из чисел на простые множители". Но представьте, что число достаточно большое, ну хотя бы больше миллиарда, тогда для разложения вам понадобится таблица первых сотен или тысяч простых чисел, а построение этой таблицы задача более трудоемкая, чем исходная задача.

Сегодня мы познакомим вас с очень простым и эффективным методом нахождения наибольшего общего делителя.

## АЛГОРИТМ ЕВКЛИДА С ВЫЧИТАНИЕМ

Идея метода проста. ***Из набора выбираются любые два ненулевых числа, и большее из них*** (или любое, если числа равны) ***заменяется разностью этих чисел***. Этот процесс повторяется до тех пор, пока не останется одно ненулевое число. Это число и будет наибольшим общим делителем исходного набора, состоящего из натуральных чисел.

В таблице 1 как пример представлен протокол работы одного из возможных алгоритмов, основанных на методе вычитания, для чисел 72, 84, 132, 144.

*Замечание.* Когда мы употребляем слово "алгоритм",

| | 1-ое число | 2-ое число | 3-е число | 4-ое число | Сумма чисел |
|---|---|---|---|---|---|
| Шаг 1 | **72** | 84 | **132** | 144 | 432 |
| Шаг 2 | **72** | **84** | 60 | 144 | 360 |
| Шаг 3 | 72 | **12** | 60 | **144** | 288 |
| Шаг 4 | 72 | 12 | **60** | **132** | 276 |
| Шаг 5 | **72** | 12 | 60 | **72** | 216 |
| Шаг 6 | **72** | 12 | **60** | 0 | 144 |
| Шаг 7 | **12** | **12** | 60 | 0 | 84 |
| Шаг 8 | **12** | 0 | **60** | 0 | 72 |
| Шаг 9 | **12** | 0 | **48** | 0 | 60 |
| Шаг 10 | **12** | 0 | **36** | 0 | 48 |
| Шаг 11 | **12** | 0 | **24** | 0 | 36 |
| Шаг 12 | **12** | 0 | **12** | 0 | 24 |
| Шаг 13 | 12 | 0 | 0 | 0 | 12 |

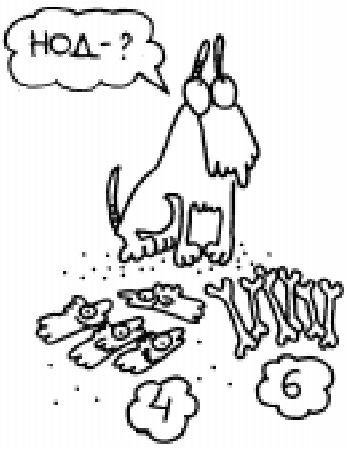
**Таблица 1.**

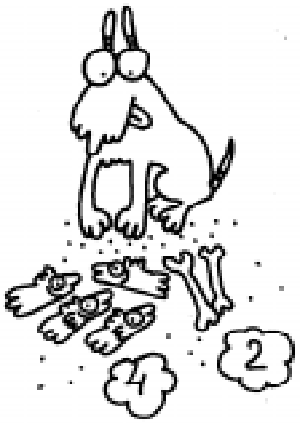
мы подразумеваем, что все выполняемые операции определяются однозначно, и для любых допустимых входных данных за конечное время порождается результат (выходные данные). Интересно, что, формулируя метод вычитаний, можно было бы сказать "выберем два случайных положительных числа набора", и тогда, благодаря наличию датчиков псевдослучайных чисел в программном обеспечении компьютера, метод вычитаний превращается в алгоритм.

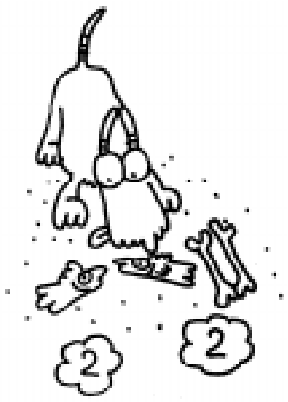
Выбирая для вычитания различные пары, можно получить разные алгоритмы. Все эти алгоритмы будут решать поставленную задачу. Для того чтобы в этом убедиться, необходимо обосновать корректность алгоритмов, то есть доказать, что каждый такой алгоритм обязательно закончит свою работу, что полученный в результате работы алгоритма набор будет содержать только одно ненулевое число и что это число будет наибольшим общим делителем исходного набора.

***Обоснование корректности алгоритма*** - очень важный этап для программирования.

Доказать корректность изложенного метода несложно.




**Доказательство.**

1. Для доказательства заметим, что если числа $a_1$ и $a_2$ делятся на $b$, то и их разность делится на $b$.

Следовательно, указанная операция над набором чисел сохраняет общие делители набора. Аналогично можно доказать, что эта операция не добавляет новых общих делителей. Мы доказали, что на каждом шаге алгоритма *множество общих делителей не меняется*.

2. Далее заметим, что при указанных преобразованиях *числа набора остаются неотрицательными.* Действительно, вычитая из положительного числа меньшее (или равное ему), мы получим неотрицательное число. Поэтому операцию вычитания можно осуществлять до тех пор, пока в наборе есть хотя бы два положительных числа.

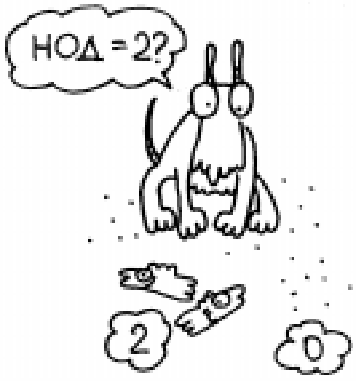
3. Наконец, *процесс изменения набора обязательно закончится,* так как после каждого вычитания сумма чисел набора уменьшается. В то же время эта сумма всегда есть положительное целое число, следовательно, процесс не может длиться бесконечно (очевидно, что число шагов не превышает суммы чисел исходного набора). ■

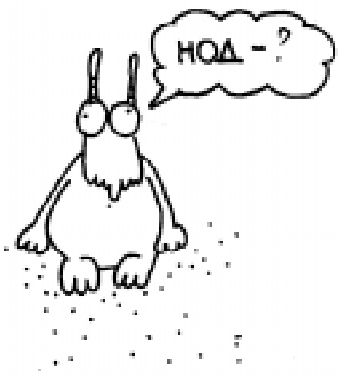
## ДЕЛЕНИЕ С ОСТАТКОМ

Рассмотрим такой ***алгоритм*** для двух положительных целых чисел $a$ и $b$:

ПОКА $a-b\geq 0$

ДЕЛАТЬ заменять $a$ на $a-b$



*Например*, для чисел $a$=37 и $b$=11 результат этого алгоритма $a$=4 (последовательные значения $a$: 37, 26, 15, 4).

Вы наверняка знакомы с другим алгоритмом, дающим такой же результат: алгоритмом деления целых чисел с остатком. Результат работы этого алгоритма можно проверить так: умножить частное на делитель и прибавить остаток. Должно получиться делимое.

**Утверждение.** Для натуральных $a$ и $b$ (делимого и делителя) единственным образом находятся числа $q$ и $r$ (частное и остаток), обладающие следующими свойствами:

$$a = bq + r, \qquad 0 \le r < b$$

В популярных языках программирования имеются операции для нахождения частного и остатка от деления целых чисел. Например, в языке Pascal эти операции записываются так:

$q:=a \operatorname{div} b$ $r:=a \bmod b$.

В дальнейшем мы будем использовать $a \operatorname{div} b$ и $a \bmod b$ для обозначения частного и остатка при делении $a$ на $b$.

Условие $0 \le r < b$ говорит о том, что остаток всегда неотрицателен и меньше делителя. При $r=0$ получается определение делимости чисел.

## АЛГОРИТМ ЕВКЛИДА С ДЕЛЕНИЕМ

Рассмотрим снова задачу о нахождении наибольшего общего делителя набора натуральных чисел. Преобразование набора будем осуществлять по новому правилу: ***возьмем два ненулевых числа из набора и большее из них*** (или любое в случае равенства) ***заменим остатком от деления на меньшее***. Далее будем действовать так же, как и в методе вычитаний. Оказывается, этот метод тоже порождает НОД исходного набора чисел.

Если чисел в наборе два, то альтернативы выбора пар нет. Тогда оба метода (вычитаний и делений с остатком) однозначно определяют все действия, какими бы ни были исходные положительные целые числа, поэтому для набора из двух чисел эти методы являются алгоритмами.

Интересный смысл имеют частные $c_0$, $c_1$, ..., $c_n$, которые получаются в процессе применения алгоритма Евклида к чмслам $a$ и $b$. Они являются членами представления числа $\frac{a}{b}$ в форме так называемой *цепной дроби*:

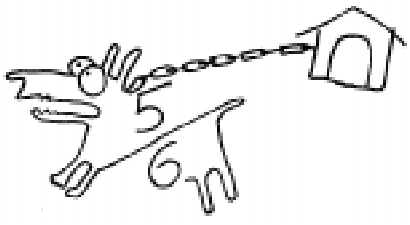
$$\frac{a}{b} = \frac{bc_0 + r}{b} = c_0 + \frac{r}{b} = c_0 + \frac{1}{b/r} = \ldots$$

$$= c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \ldots \cfrac{1}{c_{n-1} + \cfrac{1}{c_n}}}}$$

| Число 1 | Число 2 | Частное |
|---|---|---|
| $a$ | $b$ | $c_0$ |
| $r$ | $b$ | $c_1$ |
| ... | ... | ... |
| ... | ... | $c_n$ |

*Например,* $\frac{37}{24} = 1 + \frac{13}{24} = 1 + \frac{1}{24/13} =$

$$= 1 + \cfrac{1}{1 + \frac{11}{13}} = \ldots = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{5 + \frac{1}{2}}}}$$



***Алгоритм Евклида относится к числу "быстрых" алгоритмов.*** На каждом шаге этого алгоритма большее число уменьшается более чем вдвое (например, для чисел {1999; 1000} после первого шага число 1999 будет заменено на 999 (то есть 1999 уменьшится более чем вдвое), если

же взять пары {1999; 1001} или {1999, 999}, то после первого шага получим соответственно 998 для первой пары и 1 для второй; как видно, вариации второго числа только уменьшают остаток).

Долгое время алгоритм Евклида был самым эффективным способом отыскания наибольшего общего делителя, однако с появлением электронно-вычислительных машин ситуация изменилась (алгоритм Евклида, как нетрудно понять, появился задолго до вычислительных машин). Учет специфических особенностей выполнения арифметических операций компьютером позволил построить более эффективную (для программной реализации) версию алгоритма Евклида. Действительно, для оценки трудоемкости алгоритма правильно не просто считать общее число операций, но и учитывать время, которое требуется для выполнения этих операций. Так, например, операция умножения включает в качестве промежуточных операций несколько сложений, а операция деления – несколько умножений и вычитаний.

В вычислительных машинах реализована так называемая двоичная арифметика (о ней будет рассказано в одном из следующих занятий). В этой арифметике очень легко умножать и делить числа на 2. Это достигается простым сдвигом цифр (в обычной арифметике простым добавлением и удалением нуля осуществляется умножение и деление на 10).

Предлагаем вам проверить и обосновать корректность следующего алгоритма нахождения наибольшего общего делителя, который благодаря своим особенностям, допускает более эффективную реализацию на компьютере, нежели описанные выше алгоритмы.

**Алгоритм.** Пусть $a$ и $b$ натуральные числа.

ПОЛОЖИТЬ НОД=1;
ПОКА оба числа $a$ и $b$ четные
ДЕЛАТЬ поделить каждое из них на 2 и
умножить НОД на 2;
ВСЕ-ПОКА

| | 1-ое число | 2-ое число | 3-е число | Соотношение |
|---|---|---|---|---|
| Шаг 1 | 30 | **42** | **280** | |
| Шаг 2 | **30** | 42 | **28** | 28=280-42*6 |
| Шаг 3 | **2** | **42** | 28 | 2= 30-28*1 |
| Шаг 3 | **2** | 0 | **28** | ... ↑ |
| Шаг 5 | 2 | 0 | 0 | НОД=2 |

В итоге получаем
2=30-28*1=30-(280-42*6)*1=30*1-280*1+42*6

**Таблица 2.**

ПОКА оба числа $a$ и $b$ отличны от нуля
ДЕЛАТЬ
ЕСЛИ одно из чисел $a$, $b$ четное
ТО поделить его на 2
ИНАЧЕ заменить большее из чисел $a$, $b$ их разностью;
{Комментарий: в процессе выполнения последнего цикла одно из чисел $a$, $b$ обязательно будет нечетным}
ВСЕ-ПОКА
выбрать из получившихся чисел $a$, $b$ ненулевое и домножить на него НОД

В ряде задач бывает нужно ***представить некоторое целое число в виде суммы чисел, кратных заданным***. Понятно, что для решения этой задачи необходимо, чтобы это число делилось на наибольший общий делитель заданных чисел (так как каждое слагаемое такого представления делится на него). Оказывается, это условие является также и достаточным, то есть любое число, делящееся на наибольший общий делитель, можно представить в таком виде!

**Утверждение.** Наибольший общий делитель набора чисел можно представить в виде суммы чисел, кратных числам набора.

Алгоритм Евклида (использующий деление с остатком) позволяет явным образом найти это представление. А именно, ***каждый раз при замене числа a на его остаток по модулю b нужно записать равенство*** $r=a-bq$. В конце мы получим такое равенство для наибольшего общего делителя. А теперь только осталось ***подставить эти равенства друг в друга, начиная с последнего!***

*Пример* нахождения такого представления для чисел 30, 42 и 280 показан в таблице 2. Наибольший общий делитель набора чисел, равный 2, представлен в виде суммы чисел 30*1+280*(-1)+42*6, кратных числам набора 30, 280, 42, соответственно.

## АРИФМЕТИКА ОСТАТКОВ

Представим себе, что нас интересуют не сами числа, а их остатки от деления на некоторое число $m$ (говорят, "остатки по модулю $m$"). Такая ситуация встречается довольно часто.

*Пример.*

**Игра “в камушки”.**

Из кучки камней двое играющих по очереди берут 1, 2 или 3 камня. Проигрывает тот, кто берет последний камень. Как играть второму, чтобы выиграть, если в кучке 17 камней ?

***Выигрышная стратегия второго игрока:***

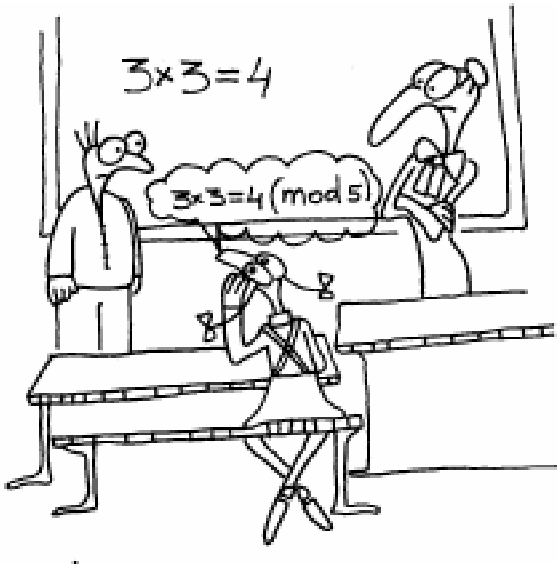
Второй должен брать всегда столько, чтобы вместе с его противником взять 4 камня. Так как остаток от деления 17 на 4 равен 1, последний камень достанется первому игроку.

Если нам известны остатки $r_1$ и $r_2$ чисел $a$ и $b$ по модулю $m$, то, даже не зная самих чисел, можно определить остатки суммы $a+b$ или произведения $ab$ по модулю $m$. Для этого надо сложить или перемножить заданные остатки и затем найти остатки от деления $r_1+r_2$ или $r_1r_2$ на $m$. Сформулируйте самостоятельно алгоритм для нахождения остатка разности.

Теперь забудем о самих числах и будем работать только с их остатками от деления на $m$. Получается замечательная арифметика – в ней всего $m$ чисел:

0, 1, 2, 3, ..., $m$-1.

Мы можем построить таблицы сложения и умножения, которых будет достаточно для перемножения любых чисел этой замечательной арифметики. Нам не понадобятся ни алгоритмы “поразрядного” сложения, ни умножения “столбиком”! Такую арифметику мы будем называть модульной, а число $m$ - ее модулем.




*Пример* таблиц сложения и умножения для $m$=5 на рисунке 1.

Особенно интересна модульная арифметика для простого числа $p$. Так, в ней можно определить деление остатков. А именно, если $a$ и $b$ – остатки по модулю $p$, $b\neq 0$, то существует такой остаток $c$, что $a=bc$; естественно его обозначить через $a/b$. Таким образом, остатки по модулю $p$ “так же хороши”, как и рациональные числа: для них можно определить действия сложения, вычитания, умножения и деления, причем они обладают привычными свойствами (говоря математическим языком, они образуют поле).

Важность модульной арифметики для вычислений определяется следующей теоремой.

| ⊕ | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | 0 | 1 | 2 | 3 | 4 |
| **1** | 1 | 2 | 3 | 4 | 0 |
| **2** | 2 | 3 | 4 | 0 | 1 |
| **3** | 3 | 4 | 0 | 1 | 2 |
| **4** | 4 | 0 | 1 | 2 | 3 |

Таблица сложения по модулю 5.

| ⊛ | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 2 | 3 | 4 |
| **2** | 0 | 2 | 4 | 1 | 3 |
| **3** | 0 | 3 | 1 | 4 | 2 |
| **4** | 0 | 4 | 3 | 2 | 1 |

Таблица умножения по модулю 5.

**Рисунок 1.**

**Китайская теорема об остатках.** Если заданы натуральные попарно взаимно простые числа $m_1$, $m_2$, ..., $m_n$ и целые числа $k_1$, $k_2$, ..., $k_n$ такие, что при любом $i$ $0 \le k_i < m_i$, то существует единственное число $k < m_1 m_2 \dots m_n$ такое, что для всех $i$ остаток от деления $k$ на $m_i$ равен $k_i$.

Смысл этой теоремы в том, что натуральные числа однозначно представляются своими остатками по некоторой системе модулей.

Как говорят в таких случаях, имеет место взаимно-однозначное соответствие между числами, меньшими $m_1 m_2 \dots m_n$, и наборами их остатков по модулям чисел $m_i$. Если учесть, что это соответствие сохраняется при арифметических действиях, мы получаем мощное средство для быстрых вычислений с большими числами!

На практике в качестве $m_i$ обычно берут несколько первых простых чисел.

# ЗАДАЧИ

**Задача 1.**

***Уровень 1.*** Рассмотрим алгоритм нахождения НОД, построенный на основе метода вычитаний с таким уточнением: “на каждом шаге алгоритма из наибольшего числа набора вычитается следующее по величине или равное”.

а) Примените этот алгоритм к набору из примера: {72; 84; 132; 144}.

б) Приведите пример набора из четырех различных чисел, для которого этот алгоритм является лучшим из всех алгоритмов, основанных на методе “вычитаний”.

в) Приведите пример набора из четырех различных чисел, для которого этот алгоритм не является лучшим.

**Задача 2.**

Предложите алгоритм выбора пар для вычитания, обеспечивающий возможно меньшее число шагов в обсуждаемом методе нахождения НОД.

***Уровень 1.*** Опишите алгоритм. Продемонстрируйте его работу для набора чисел из пункта в) задачи 1.

***Уровень 2.*** Напишите программу, которая по заданному набору из $n$ чисел определяет последовательность выбора пар, приводящую к результату за возможно меньшее число шагов.

Формат ввода:

Количество чисел $n \le 10$

1-ое число набора

2-ое число набора

...

$n$-ое число набора

Формат вывода:

Количество операций $m$

Номер 1-го уменьшаемого Номер 1-го вычитаемого

Номер 2-го уменьшаемого Номер 2-го вычитаемого

...

Номер $m$-го уменьшаемого Номер $m$-го вычитаемого

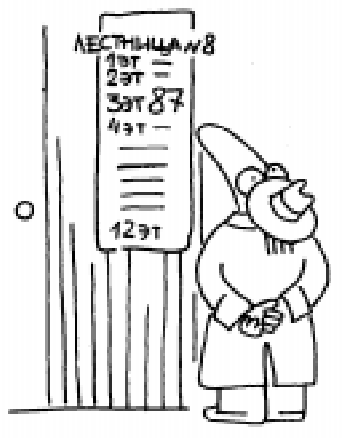
*Пример.*

Ввод:

```
3
10
6
4
```

Вывод:

```
 5
13
12
23
32
23
```




**Задача 3.**

***Уровень 1.*** Ваш знакомый живет в стандартном двенадцатиэтажном доме в квартире 87. На каком этаже может располагаться его квартира? (На лестничной

площадке одно и то же число квартир). Вообще, как быстро определить, может ли квартира с данным номером находиться на данном этаже?

**Задача 4.**

***Уровень 1.*** Язык племени мумбу-юмбу состоит из шестибуквенных слов, составленных из букв {А, Б, В, Г, Д, Е, Ж, З, И, К}. В Оксфорде издан полный словарь слов этого языка (упорядоченных по алфавиту:

АААААА
АААААБ
АААААВ
...
КККККИ
КККККК.

На каждой странице словаря помещается 15 слов.

На каких страницах и в каких строках находятся слова ДЕКАДА и ЗАБАВА?


**Задача 5.**

Найдите цифру с номером $n$ в последовательности
01234567891011121314151617181920...
записанных подряд натуральных чисел.

***Уровень 1.*** Опишите алгоритм. Найдите цифру с номером 1999.

***Уровень 2.*** Напишите программу.

Формат ввода:

Число $n < 1000000$

Формат вывода:

Цифра с номером $n$

*Пример.*

Ввод:

31

Вывод:

2

**Задача 6.**

Для любого натурального числа алгоритм совершает следующие операции: отделяет от числа первую цифру и прибавляет ее к числу из оставшихся цифр. Процесс продолжается до тех пор, пока в числе останется одна цифра. Например: 123456→23457→3459→462→66→12→3.

***Уровень 1.*** Определить результат работы алгоритма для чисел вида $\underbrace{88\ldots8}_{n}$ при различных значениях $n$.

***Уровень 2.*** Напишите программу, которая выдает все промежуточные результаты для чисел из не более чем 50 цифр.

Формат ввода:

Число $n$ цифр заданного числа
1-я цифра
2-я цифра
...
$n$-я цифра.

Формат вывода:

Первый промежуточный результат
Второй промежуточный результат
...
Итоговая цифра.

**Задача 7.**

Известно, что любую дробь $\frac{a}{b}$, где $a$ и $b$ - натуральные числа, можно представить в виде цепной дроби.

***Уровень 1.*** Опишите алгоритм. Представьте в виде цепной дроби $\frac{17}{239}$.

***Уровень 2.*** Напишите программу, которая по данным числам $a$ и $b$ представляет дробь $\frac{a}{b}$ в виде цепной ($a, b < 1000000$).

Формат ввода:

Число $a$
Число $b$

Формат вывода:
Количество числителей $n$
Число $c_0$
Число $c_1$
...
Число $c_n$

*Пример.*
Ввод:
5
9

Вывод:
3
0
1
1
4

**Задача 8.**

***Уровень 1.*** Пусть большее из чисел набора $\{a,b\}$ равно 1999. Какое максимальное число шагов может сделать алгоритм Евклида с делением для такого набора в худшем случае? Каким при этом будет второе число набора? Приведите пример такого числа. Сколько таких чисел?

***Уровень 2.*** Придумайте алгоритм для нахождения по числу $a$ ($a$<1000000) всех чисел $b$, меньших $a$ и таких, для которых алгоритм Евклида делает максимальное число шагов.

Формат ввода:
Число $a$

Формат вывода:
Число шагов
Количество $n$ различных значений $b$
1-ое значение $b$
2-ое значение $b$
...
$n$-ое значение $b$

**Задача 9.**

***Уровень 1.*** Обобщить алгоритм Евклида с делением на 2 и вычитанием для наборов из более чем двух чисел. Применить обобщенный алгоритм к набору {72; 84; 132; 144}.

***Уровень 2.*** Доказать, что если одно из чисел $a$, $b$ четное, а другое нечетное, то следующее преобразование сохраняет множество общих делителей $a$ и $b$:
ЕСЛИ ровно одно из чисел $a$, $b$ четное
ТО поделить его на 2
ИНАЧЕ заменить большее из чисел $a$, $b$
их разностью

**Задача 10.**

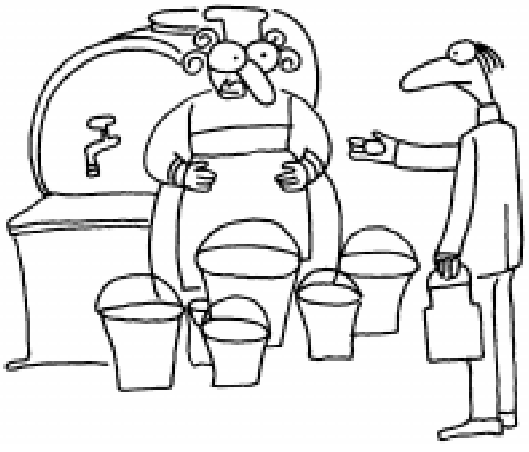
Есть несколько ведер с различными емкостями. Разрешается этими ведрами добавлять или вычерпывать воду из бочки. Сначала бочка пуста.

***Уровень 1.*** Необходимо налить в бочку 1 литр воды, имея в распоряжении ведра емкостью 17, 32 и 45 литров. Как это сделать поскорее (за возможно меньшее количество переливаний)? Сколько при этом будет перелито воды? Можно ли выполнить эту работу, чтобы общее количество перелитой воды было еще меньше?

***Уровень 2.*** Написать программу, которая выдает последовательность действий для наливания в бочку $a$ ($a$<1000000) литров воды.

Формат ввода:
Число ведер $n \le 10$
Емкость 1-го ведра
Емкость 2-го ведра
...
Емкость $n$-го ведра
Нужное число литров $a$

Формат вывода:
НЕЛЬЗЯ
или
Число операций $m$
1-ая операция



2-ая операция
...
*m*-я операция
Каждая операция представляет собой
+номер ведра (долить в бочку)
или
-номер ведра (вычерпать из бочки)

*Пример.*

Ввод:

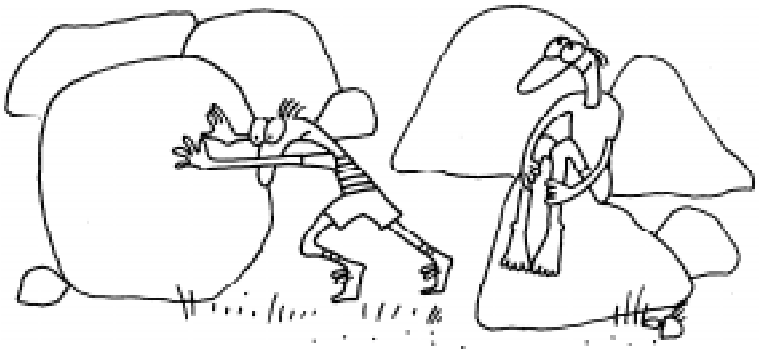
2
3
5
1

Вывод:

3
+1
+1
-2

**Задача 11.**

***Уровень 1.*** Из кучки камней двое играющих по очереди берут 1, 2 или 3 камня. Проигрывает тот, кто берет последний камень. Предположим, что всего камней *N*. Кто из игроков имеет выигрышную стратегию? Опишите ее. А если выигрывает взявший последний камень?



**Задача 12.**

***Уровень 1.*** Алиса, попав в Страну Чудес, забыла таблицу умножения. Она говорит:
семью семь будет ... пятнадцать, девятью девять будет ... тринадцать.

Определите в какой модульной арифметике такие правила умножения справедливы. Сколько всего таких арифметик?

***Уровень 2.*** Напишите программу, которая по введенным примерам таблицы умножения определяет, существует ли модульная арифметика, в которых эти примеры имеют место. Чем больше таких модулей найдет Ваша программа и чем быстрее она это сделает, тем лучше.


Формат ввода:
Число равенств $n \le 10$
1-ое равенство
2-ое равенство
...
*n*-ое равенство
Каждое равенство имеет вид
число1*число2=
=число3

Формат вывода:
Число найденных модулей *m*
1-ый модуль
2-ой модуль
...
*m*-ый модуль

*Пример.*

Ввод:

1
3*5 = 3

Вывод:

2
6
12

**Задача 13.**

По заданным остаткам *a*, *b* по простому модулю *p* найти *a/b* (в решении этой задачи пригодится представление наибольшего общего делителя суммой чисел, кратных числам набора).

***Уровень 1.*** Опишите алгоритм. Найдите 17/23 по модулю 239.

***Уровень 2.*** Напишите программу вычисления частного *a/b* ($a,b,p<1000000$).

Формат ввода:
Модуль *p*
Число *a*

Число $b$
Формат вывода:
Число $a/b$ (по модулю $p$)

*Пример.*
Ввод:
5
2
3
Вывод:
4

**Задача 14*.**

***Уровень 2.*** Напишите программу, которая по заданным не более чем 25-значным числам $a$, $b$, $c$ и $d$ проверяет, верно ли, что $ab = cd$.

*Указание*. Проверьте равенство для остатков от деления чисел $a$, $b$, $c$ и $d$ на все простые числа до 100. (Так как произведение всех простых чисел до 100 более чем 25-значное, по китайской теореме об остатках этого достаточно, чтобы обосновать исходное утверждение.)

Формат ввода:
Число $a$
Число $b$
Число $c$
Число $d$
Формат вывода:
Да или Нет

*Пример.*
Ввод:
1234123412341234123412340
8888888888888888888888
2468246824682468246824682468
4444444444444444444444440
Вывод:
Да

## ЖЕЛАЕМ УСПЕХОВ В РЕШЕНИИ ЗАДАЧ!

***Петров Виктор Александрович, студент мат.-мех. факультета СПбГУ.***
***Поздняков Сергей Николаевич, доцент СПбГЭТУ.***

***НАШИ АВТОРЫ***