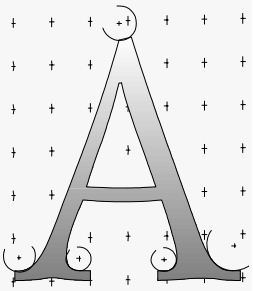
№ 35 (228) сентябрь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК



В 1999 году исполнилось 250 лет со дня рождения выдающегося французского астронома, автора "Небесной механики", математика и физика, одного из крупнейших ученых XVIII — XIX столетий Пьера Симона Лапласа (1749 — 1827)

О Пьере Симоне Лапласе читайте на стр. 32

Читайте в номере

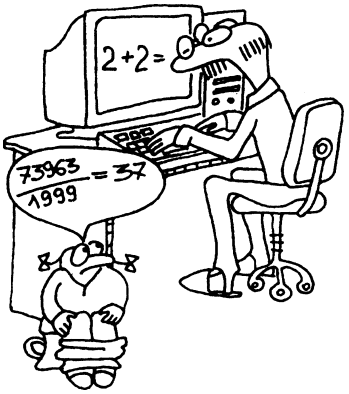
Алгоритмы над целыми числами

КОМПЬЮТЕРНЫЕ ИНСТРУМЕНТЫ В ОБРАЗОВАНИИ

В.А. Петров,
С.Н. Поздняков

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, которые люди знают с детства, 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 знания вышеперечисленных алгоритмов уже достаточно, чтобы объяснить несколько важных идей компьютерной математики, которые используются при программировании операций с целыми числами.

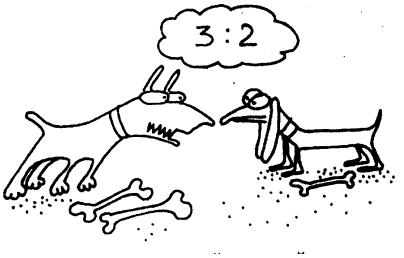


ДЕЛИМОСТЬ

Распространенной задачей является такая: определить, делится ли одно число на другое без остатка. Например, делим число 73 963 на 1999. Применяем алгоритм деления "уголком":

```
 73963 | 1999
-5997  | 37
-------|
 13993
-13993
-------
     0
```

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 73 963.

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 любого 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.

В № 33/99 мы опубликовали объявление о приеме школьников в Заочную школу современного программирования, которую организует журнал "Компьютерные инструменты в образовании". Надеемся, что материалы этой школы, которые мы планируем публиковать, будут интересны и нашим читателям.

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 его на простые множители вам понадобится таблица первых сотен или тысяч простых чисел, а построение этой таблицы — задача более трудоемкая, чем исходная.

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”, 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.

Таблица 1

Номер шага	1-е число	2-е число	3-е число	4-е 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

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 ≥ 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:

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

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

```
q:=a div b, r:=a mod b.
```

В дальнейшем мы будем использовать `a div b` и `a mod 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, с помощью этого метода тоже можно получить 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 числам a и b. Они являются членами представления числа $\frac{a}{b}$ в форме так называемой *цепной дроби:*

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

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

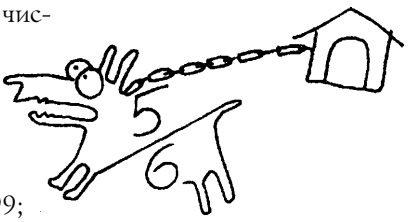
Число 1	Число 2	Частное
a	b	c_0
b	r	c_1
...	...	...
...	...	c_n

Например,

$$\frac{37}{24} = 1 + \frac{13}{24} = 1 + \frac{1}{24/13} = 1 + \frac{1}{1 + \frac{11}{13}} = \cdots =$$

$$= 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{5 + \frac{1}{2}}}}$$

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 выполне-

ния 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 следующей статье этой серии). 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
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 $30 \cdot 1 + 280 \cdot (-1) + 42 \cdot 6$, кратных числам набора 30, 280, 42 соответственно.

Таблица 2

Номер шага	1-е число	2-е число	3-е число	Соотношение
Шаг 1	30	**42**	**280**	
Шаг 2	**30**	42	**28**	$28 = 280 - 42 \cdot 6$
Шаг 3	**2**	**42**	28	$2 = 30 - 28 \cdot 1$
Шаг 4	**2**	0	**28**	↑
Шаг 5	2	0	0	НОД=2

В итоге получаем $2 = 30 - 28 \cdot 1 = 30 - (280 - 42 \cdot 6) \cdot 1 = 30 \cdot 1 - 280 \cdot 1 + 42 \cdot 6$.

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".**

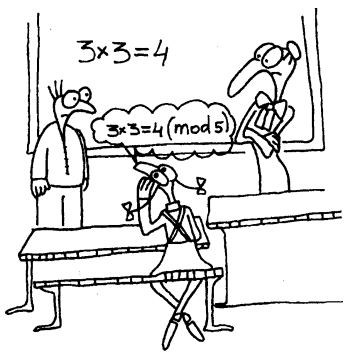
Из кучки камней двое играющих по очереди берут 1, 2 или 3 камня. Проигрывает тот, кто берет последний камень. Как играть второму, чтобы выиграть, если в кучке 17 камней?

Выигрышная стратегия второго игрока:

Второй должен брать всегда столько, чтобы вместе со своим 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 $r_1 r_2$ 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 $m = 5$ приведен ниже.

⊕	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

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.

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 $k < m_1, m_2, ..., m_n$ 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 $m_1, m_2, ..., m_n$, 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-е число набора
2-е число набора
...
n-е число набора

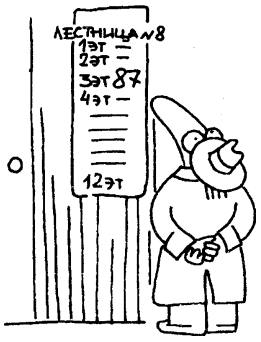
Формат вывода:
Количество операций m
Номер 1-го уменьшаемого Номер 1-го вычитаемого
Номер 2-го уменьшаемого Номер 2-го вычитаемого
...
Номер m-го уменьшаемого Номер m-го вычитаемого

Пример

Ввод:	*Вывод:*
3	5
10	1 3
6	1 2
4	2 3
	3 2

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):

ААААА,
ААААБ,
ААААВ,
...
КККККИ,
ККККК.

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 $n < 1\ 000\ 000$

Формат вывода:
Цифра с номером n

Пример

Ввод:	*Вывод:*
31	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 оканчивается тогда, когда в числе остается одна цифра. Например:

$123456 \to 23457 \to 3459 \to 462 \to 66 \to 12 \to 3$.

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:
Количество 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 дроби ($a, b < 1\ 000\ 000$).

Формат ввода:
Число a
Число b

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

Пример

Ввод:	*Вывод:*
5	3
9	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 ($a < 1\,000\,000$) 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-е значение b
2-е значение b
...
n-е значение b

ЗАДАЧА 9

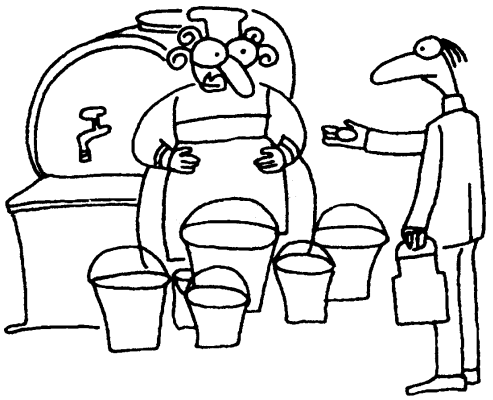
Уровень 1. Обобщить алгоритм Евклида с делением на 2 и вычитанием для наборов из более чем двух чисел. Применить обобщенный алгоритм к набору {72; 84; 132; 144}.

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 ($a < 1\,000\,000$) 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-я операция
2-я операция
...
m-я операция

Каждая операция представляет собой
+ номер ведра (долить в бочку)
или
— номер ведра (вычерпать из бочки)

Пример

Ввод:	*Вывод:*
2	3
3	+1
5	+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-е равенство
2-е равенство
...
n-е равенство

Каждое равенство имеет вид
число1 * число2 = число3

Формат вывода:
Число найденных модулей m
1-й модуль
2-й модуль
...
n-й модуль

Пример

Ввод:	*Вывод:*
1	2
$3 * 5 = 3$	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 ($a, b, p < 1\,000\,000$).

Формат ввода:
Модуль p
Число a
Число b

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

Пример

Ввод:	*Вывод:*
5	4
2	
3	
4	

ЗАДАЧА 14 (повышенной трудности)

***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:
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 0
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
2 4 6 8 2 4 6 8 2 4 6 8 2 4 6 8 2 4 6 8 2 4 6 8
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0

Вывод:
Да

ЖЕЛАЕМ УСПЕХОВ В РЕШЕНИИ ЗАДАЧ!

Интернет-школа программирования СПбГИТМО

Уважаемые коллеги!

В прошлом году кафедрой компьютерных технологий Санкт-петербургского государственного института точной механики и оптики (технического университета) и федеральной университетской компьютерной сетью RUNNet проводились занятия в Интернет-школе программирования.

Приглашаем вас и ваших учеников в нашу школу на новый учебный год 1999/2000.

Новый цикл обучения вновь будет ориентирован на достаточно способных учащихся 10—11-х классов, которые, как мы надеемся, уже имеют неплохую математическую подготовку, интересуются информатикой, выступают на олимпиадах и хотят в ближайшем будущем стать студентами "компьютерных" кафедр и факультетов ведущих российских вузов.

Основным содержанием работы Интернет-школы станет углубленная подготовка занимающихся в рамках тех разделов курса информатики, которые тематически можно объединить названием "Алгоритмы обработки данных".

Соответственно, предполагается, с учетом направленности занятий, что будущие ученики Интернет-школы уже знакомы с основами программирования. Занятия в нашей школе включают, помимо изучения теоретического материала, проведение дистанционного тестирования решений, присылаемых участниками школы. Обращаем ваше внимание, что при проведении тестирований мы будем использовать только компиляторы Borland для языков Паскаль и Си.

При организации учебного процесса в Интернет-школе используется опыт, накопленный при проведении российских школьных олимпиад по информатике и европейских региональных студенческих олимпиад чемпионата мира по программированию, а также при подготовке участников этих олимпиад.

Обо всех деталях, касающихся порядка зачисления и дальнейшей деятельности, вы узнаете, прислав заявку электронной почтой на наш адрес: **ses@spb.runnet.ru**. Просим в ней указать, "кто вы", "откуда", "сколько учеников и каких классов вы предполагаете делегировать в Интернет-школу".

Вашу заявку мы рассчитываем получить в течение двух-трех недель после выхода этого номера газеты "Информатика".

Ждем ваших заявок!

Кафедра компьютерных технологий СПбГИТМО

Информатика в этюдах

В.М. Казиев

В последнее время междисциплинарные аспекты, проблемы фундаментализации, гармонизации, гуманитаризации и гуманизации образования становятся более актуальными. В то же время преподавание информатики классически тяготеет к технократическому варианту "преподавания основ вычислительной техники, программирования и пользовательского интерфейса" (мы ни в коей мере не умаляем значимость такого подхода на определенном уровне подготовки и для определенных категорий обучаемых), что уже не адекватно стоящим перед наукой "Информатика" задачам (а мы эту науку воспринимаем в первую очередь как теоретическую, системно-математическую [1] и междисциплинарную).

При решении задач образовательной информатики в соответствии с указанной парадигмой важное значение приобретает обучение на развивающих, междисциплинарных специально дидактически подобранных, так называемых развивающих, задачах (см. также [2—6]), или этюдах (здесь сыграла свою роль замечательная книга "Этюды для программистов" Ч.Уэзерелла).

Предлагаемые этюды важны и необходимы при достижении не только музыкального и художественного мастерства, но при достижении высокого уровня информационной культуры.

Этюды разбиты по темам и снабжены подробными решениями. Приведен необходимый для понимания и решения рассматриваемых задач теоретический минимум, который, впрочем, не является "заменителем" учебников и лекций. Задачи и решения могут быть использованы для обучения, проведения олимпиад. С этой же целью сформулированы и темы для этюдов.

Данный материал был апробирован при обучении в школе-лицее КБГУ, на первом курсе специальности "прикладная математика" КБГУ, на олимпиадах школьников в КБР, и, как показывает опыт, материал вполне достаточен для приобретения устойчивых полных знаний по теоретической (математической) информатике — основы для перехода к программированию, новым информационным технологиям, пользовательскому интерфейсу.

Замечания, отзывы, пожелания и просьбы выслать другие задачи указанного цикла (а он постоянно пополняется автором) присылать по электронной почте: **kaziev@ns.kbsu.ru**.

На web-странице автора **(http://access.nalnet.ru/~svd/imoas)** можно найти ряд других полезных задач и материалов.

1. Информация: кодирование, шифрование, измерение. Информационно-логические задачи

Теоретический минимум

Алфавит — конечное множество знаков для представления сообщений, т.е. цепочек из этих знаков (слов), образуемых по некоторым, вполне определенным правилам. Примерами алфавитов являются:

- алфавит арифметики — множество из десяти цифр, знаков арифметических операций и десятичной точки (запятой);
- алфавит из букв русского языка, ошибочно часто называемый алфавитом русского языка (для этого необходимо добавить к ним знаки препинания);
- алфавит русского языка — множество знаков русского языка, знаков препинания и пробела;
- алфавит китайских иероглифов;
- алфавит из точек и тире в азбуке Морзе и др.

Информация — это некоторая последовательность сведений, знаний, сообщений, выражаемых с помощью некоторого алфавита символов, жестов, звуков. Информация — это содержание сообщения, сообщение — форма проявления информации. Формы облачения информации в сообщения различны, например, для живых существ — сигналы, жесты, а для различных технических устройств — сигналы.

***Основные свойства информации (сообщений)*:**

- полнота;
- ясность или корректность интерпретации, приема — передачи;
- понятность (получателю информации);
- сжимаемость (информации);
- достоверность (сведений).

Сообщения измеряются в байтах, килобайтах, мегабайтах, гигабайтах, терабайтах и петабайтах. Любые сообщения кодируются в ЭВМ в цифровом виде, с помощью алфавита из нулей и единиц, записываются и реализуются в ЭВМ в битах.

Соотношения между единицами измерения сообщений:

1 *бит* (от слова ***bi****nary digi****t*** — двоичная единица) — это 0 или 1,

1 *байт* = 8 битов, например, байт вида 10001110,

1 *килобайт* (1 *К*) = 1024 байт, или 2^{10} байт, или 2^{13} бит,

1 *мегабайт* (1 *М*) = 1024 *К*, или 2^{10} *К*, или 2^{20} байт, или 2^{23} бит,

1 *гигабайт* (1 *Г*) = 1024 *М* = 2^{10} *М* = 2^{20} *К* = = 2^{30} байт = 2^{33} бит,

1 *терабайт* (1 *Т*) = 1024 *Г* = 2^{20} *М* = 2^{30} *К* = = 2^{40} байт = 2^{43} бит,

1 *петабайт* (1 *П*) = 2^{10} *Т* = 2^{20} *Г* = 2^{20} *М* = 2^{40} *К* = = 2^{50} байт = 2^{53} бит.

Для измерения информации используются другие, более сложные принципы и методы, например, мера информации по К.Шеннону.

Нестрогим образом и упрощенно рассуждая, далее мы будем отождествлять такие различные понятия, как сообщение и информация.

Любая информация может быть воспринята и обработана принимающей стороной только в виде некоторых закодированных сообщений на языке сигналов, жестов, символов, звуков и т.д.

Кодирование — процесс преобразования букв (слов) одного алфавита в буквы (слова) другого алфавита. Например, при кодировании кодом Цезаря каждый символ алфавита, на котором записано сообщение, заменяется другим символом этого же алфавита, отстоящим от него на заданное число символов.

Правила кодирования (шифрования) не могут быть произвольными. Они должны быть такими, чтобы зашифрованное сообщение можно было бы прочесть (расшифровать). Однотипные правила (типа правила Цезаря) можно объединять в классы. Если внутри класса правил выбран (определен) некоторый параметр (числовой, табличный и т.д.), изменяя значения которого можно перебрать (варьировать) все правила, то такой параметр называется ***шифровальным ключом.***

Имеются две большие группы шифров: шифры перестановки и шифры замены. ***Шифр перестановки*** изменяет только порядок следования символов исходного сообщения. ***Шифр замены*** заменяет каждый символ сообщения на другие символы, не изменяя порядка их следования.

Восстановление некоторого зашифрованного сообщения по ключу — ***расшифрование***, а восстановление (прочтение) его без известного ключа — ***вскрытие (взлом)*** шифра.

Кодирование и шифрование — различные процедуры. При кодировании нет секретного ключа (достаточно знать соответствие символов), а при шифровании необходимо наличие ключа (нужно знать как соответствие — шифр, так и ключ к шифру). Кодирование не ставит основной целью недоступность для чтения, а ставит целью более сжатое, компактное и быстрое представление и преобразование текста. Шифрование же ставит целью сделать сообщение недоступным для чтения без обладания ключом к шифру.

При внутреннем представлении сообщений в памяти ЭВМ все символы кодируются байтами. Количество всех различных символов, которые можно при этом представить в ЭВМ, равно $2^8 = 256$.

Этюды с решениями

1. Можно ли сконструировать ЭВМ с быстродействием 10^{20} операций в секунду на принципах переноса электронов (на электронных принципах переноса информации)? Ответ подтвердить убедительными вычислениями.

Решение. Самая большая скорость переноса электронов — скорость света в вакууме: $v = 3 \cdot 10^{10}$ см/сек. При выполнении одной операции свет (электрон) должен пройти расстояние, не меньшее диаметра атома водорода $s = 10^{-8}$ см. Время этого пути равно:

$$t = s/v = 10^{-8}\ см/(3 \cdot 10^{10}\ см/сек.) = = 10^{-18}/3\ сек.$$

Следовательно, в идеальном варианте (если когда-либо удастся технически реализовать указанные выше характеристики) быстродействие компьютеров, разрабатываемых на принципах переноса электронов, не может превышать $3 \cdot 10^{18}$ операций в секунду. Итак, ЭВМ с быстродействием 10^{20} операций в секунду реализовать на указанных принципах нельзя.

2. Приведите примеры лексикографически упорядоченной и неупорядоченной последовательностей.

Решение. Слова "*арба*", "*баня*", "*кино*", "*лето*", "*тема*" лексикографически упорядочены, если в качестве входного алфавита взять алфавит русского языка. Слова двоичного алфавита "1010", "0111", "0101", "0010" лексикографически не упорядочены.

3. Описать словесно и математически закон формирования чисел данного ряда (начиная с третьего): 1, 1, 2, 5, 29, 866, ...

Решение. Нетрудно заметить, что начальный отрезок похож на ряд Фибоначчи, но это обманчиво. На самом деле каждый член ряда (с третьего) получается не сложением двух предыдущих, а сложением их квадратов. Математически это записывается в виде:

$$x_n = (x_{n-1})^2 + (x_{n-2})^2\ ,\ \ n = 3, 4, \ldots\ .$$

4. Найти правило шифровки текста, если из текста "*АРГУМЕНТФУНКЦИИ*" получен текст "*БСДФНЖОУХФОЛЧКК*".

Решение. Сравнивая число символов исходного и зашифрованного текста (они равны) и символы, стоящие на одинаковых местах в исходном и шифрованном тексте, замечаем, что использован шифр Цезаря: каждый символ кодируется следующим за ним по порядку (определенном в алфавите) символом, причем последний символ кодируется первым символом алфавита, т.е. алфавит "закольцован".

5. Найти все решения — различные между собой цифры *S, A, P, B, D*, отличные от цифры 4, если: $SP4 + APD = DBAB$. Решение найти за минимальное число рассуждений, если одно рассуждение — простое утвердительное повествовательное предложение (высказывание, предикат).

Решение. Наши рассуждения таковы:

1) так как при сложении двух десятичных цифр возможен перенос в старший разряд только 1, то однозначно определяем $D = 1$;
2) тогда из младшего разряда получаем однозначно $B = 5$;
3) из 2-го и 3-го разрядов получаем две возможности:
   а) $P + P = A$ и $S + A - 10 = 5$ $(2P = A, S + A = 15)$;
   б) $P + P - 10 = A$ и $S + A + 1 - 10 = 5$ $(2P = A + 10, S + A = 14)$;

если рассматривать случай **а**, то отсюда следует, что A — четно, т.е. $A = 2$, 6 или 8, а S — нечетно и $S > 5$, т.е. $S = 7$ или 9, т.е. подходят только значения $A = 6$, $S = 9$ (иначе получаем $P = 4$ при $A = 8$ и $S = 7$, что невозможно — цифра 4 явно уже присутствует в примере). Аналогично рассматривается случай **б**.

6. Найти и записать закономерность образования последовательности A вида:
1, 10, 11, 100, 111, 1000, 1111, 10 000, ...
Рассмотреть отдельно случаи:
а) числа заданы в десятичной системе;
б) числа заданы в двоичной системе.

Решение. Рассмотрим случай **а**, так как случай **б** рассмотреть легче. Из сравнения членов $A[i]$ $(i = 1, 2, ...)$ последовательности, стоящих на четных местах и на нечетных местах (отдельно), видно, что:

1) элемент на нечетном месте получается из элемента на предыдущем нечетном месте дописыванием к нему единицы;
2) каждый элемент на четном месте получается из элемента на предыдущем четном месте дописыванием справа к нему нуля.

Этот словесно описанный закон можно записать в аналитическом виде. Получим отдельно для случаев **1** или **2**:

$A[2n] = 10 \cdot A[2n - 2]$, $A[2n - 1] = 10 \cdot A[2n - 1] + 1$, $n = 1, 2, ...$.

Можно записать формулу, объединяющую обе эти формулы в одну:

$A[2n + m] = 10 \cdot A[2n + n - 2] + m$, где $m = 0$ или $m = 1$.

Существует лучшая, но менее наглядная форма записи:

$A[2n + mod(n, 2)] = 10 \cdot A[2n + mod(n, 2) - 2] + mod(n, 2)$.

7. *Петя, Вася, Дима, Коля* и *Гена* хотят (каждый в отдельности) пойти в кинотеатры, где они еще не были. При этом:
— *Петя* не был в "*Победе*", "*Востоке*", "*Авроре*", "*Мире*";
— *Вася* был в "*Авроре*", "*Мире*";
— *Дима* не был лишь в "*Авроре*";
— *Коля* не был в "*Победе*", "*Мире*";
— *Гена* был в "*Востоке*", "*Мире*", "*Юности*".

Найти с помощью как можно меньшего числа рассуждений, кто и в какой кинотеатр должен пойти. За одно рассуждение можно условно принять одно повествовательное предложение (простое высказывание) относительно кинотеатра или юноши.

Решение. Эту задачу решим без составления таблицы (для краткости). Первое рассуждение: *Дима* не был только в "*Авроре*", поэтому он идет в "*Аврору*". Второе рассуждение: так как *Гена* не был только в "*Победе*", то он идет в "*Победу*". Третье рассуждение: так как *Коля* не может уже идти в "*Победу*", то он идет в "*Мир*". Четвертое рассуждение: так как *Петя* не может пойти уже в "*Победу*", "*Аврору*" или в "*Мир*", то он идет в "*Восток*". Пятое рассуждение: *Васе* однозначно остается "*Юность*". Минимальность этих рассуждений обоснована (но не доказана формально!) тем, что юношей и кинотеатров было пять, следовательно, при уменьшении количества рассуждений (если это вообще возможно) какое-то рассуждение станет уже сложным.

8. Друзья *X, Y, Z, U, V* должны поехать в разные города *А, Б, В, Г, Д, Е*. При этом: *X* может ехать только в *А, Б, Д*; *Y* может ехать только в *Б* и *Г*; *Z* может ехать только один и только в *В*; *U* не может ехать вместе с *Y*; *V* может ехать только с *X* или *Z*, но не в *Д*. В каком городе мог быть каждый из них, если оказалось, что вдвоем они не были ни в одном городе?

Решение. Составим таблицу возможностей каждого, пометив города, куда может ехать каждый, знаком "+", а куда не может — знаком "—":

	X	*Y*	*Z*	*U*	*V*
А	+	—	—	+	+
Б	+	+	—	—	+
В	—	—	+	+	+
Г	—	+	—	—	—
Д	+	—	—	+	—
Е	—	—	—	+	—

Из анализа этой таблицы следует:
1) *Z* может ехать только в *В*;
2) *U* должен ехать только в *Е* (иначе в *Е* никто не едет);
3) *Y* должен ехать только в *Г* (иначе в *Г* никто не едет);
4) *X* должен ехать только в *Д* (так как *U* в *Д* уже не может ехать);
5) *V* может ехать только в *А* или в *Б* (т.е. задача имеет два решения).

9. Имеются 192 монеты, из которых одна фальшивая (легче). Сколько взвешиваний на чашечных весах нужно произвести, чтобы определить ее?

Решение. Если положить на чашки весов равное количество монет, то получим 3 возможности:
а) чашки уравновешены;
б) левая чашка ниже;
в) правая чашка ниже.

Таким образом, каждое взвешивание дает количество информации $I = \log_2 3$ и, следовательно, для определения фальшивой монеты нужно сделать не менее k взвешиваний, где k удовлетворяет условию $\log_2 3^{k-1} \geq \log_2 192$. Отсюда $k - 1 \geq (6 + \log_2 3)/\log_2 3$, или приблизительно $k \geq 5{,}89$. Следовательно, нам необходимо сделать не менее $\mathrm{int}(k + 0{,}5) = 6$ взвешиваний (достаточно шести взвешиваний).

10. ДНК человека можно представить себе как некоторое слово в четырехбуквенном алфавите, где каждой буквой помечается звено цепи ДНК, или нуклеотид. Сколько информации (в битах) содержит ДНК, если в нем содержится примерно $1{,}5 \cdot 10^{23}$ нуклеотидов?

Решение. На один нуклеотид приходится $\log_2(4) = 2$ (бит) информации. Следовательно, ДНК в организме человека позволяет хранить $3 \cdot 10^{23}$ бит информации. В этой связи заметим, что человек за среднюю продолжительность жизни использует около 5—6% нейронов (нервных клеток мозга — "ячеек ОЗУ человека").

11. Найти и записать формулой закон формирования нижеприведенной последовательности: *AB, AAB, ABB, AAAB, ABBB, ...* .

Решение. Словесное описание закона имеет вид: к слову, стоящему на очередном нечетном месте, дописывается с конца символ "*B*", а к слову, стоящему на очередном четном месте слева, дописывается символ "*A*". "Формульная" запись закона:

$$X_{2n+1} = X_{2n-1} + 'B', \quad X_{2n} = 'A' + X_{2n-2},$$
$$n = 1, 2, 3, \ldots$$

Здесь операция "+" означает конкатенацию (присоединение текста к тексту справа), а X_n — элемент последовательности, находящийся на n-м месте.

Темы для этюдов

1. Сколько различных символов в сообщениях (один символ — 1 байт):
11100010000011111111000011100011,
11111000111110001110001000000011,
10111000000011111011000011100010.

2. Лазерный диск (*CD*) вмещает 600 *M*. Сколько страниц учебника можно записать на 1 диск, если на одной странице учебника можно записать 50 строк по 40 букв (символов) в строке? Сколько бит информации помещается на одной странице учебника указанного формата?

3. Запишите кратко и полно условие существования треугольника с заданными вершинами $A(a; b)$, $B(c; d)$, $C(m; n)$ (желательно с помощью минимума условий). Запишите условие попадания заданной точки $M(x; y)$ в этот треугольник. Сколько бит занимает запись вашего условия?

4. Сколько осмысленных (т.е. имеющих однозначный смысл) предложений можно составить из слов данного предложения: "Поздним вечером переполненный автобус вез вахтенных рабочих домой".

5. Сколько всего байт необходимо для "запоминания одного экрана" в памяти ЭВМ, если каждая точка может быть одного из 8 (16, 32, 256, 2^n) различных цветов. Каждый цвет кодируется одним байтом, а экран дисплея может вмещать 1024×640 точек?

6. Найти и записать словесно и "формульно" какой-либо (по возможности простой) закон формирования чисел в последовательностях:
а) 10, 11, 100, 121, 1000, 1331, ... ;
б) 1, 3, 15, 105, 945, ...

7. Текст "*Сегодня хороший день?*" закодирован цифрами в виде

"10020304050607000904011120005020613".

Расшифровать код и определить код и символ, помеченный знаком "?". Ответ нужно обосновать за минимум утвердительных рассуждений.

8. Найти закон формирования последовательности и записать 2 следующих члена последовательности: *ABCDEF, FBCDEA, FECDBA*, ...

9. Найти всевозможные решения (наборы цифр) *S, M, A, P, B, C*:
а) *SAM* + *MAP* = *MPPM*;
б) *A6A* — *BB* = *CA2*.
Решение найти за минимум рассуждений; одно рассуждение — одно утвердительное высказывание относительно одного разряда чисел.

10. *Петя, Миша, Ваня, Коля, Дима* должны одновременно поехать в города *Нальчик, Москва, Серпухов, Тольятти, Норильск*. При этом:
— *Петя* должен ехать только в *Нальчик, Москву* или *Норильск*;
— *Миша* должен ехать только в *Москву* или *Тольятти*;
— *Ваня* должен ехать только в *Серпухов* или *Тольятти*;
— *Коля* может ехать в любой город;
— *Дима* не может ехать вместе с *Ваней* или *Петей* в *Москву*.

В каком городе мог быть каждый, если оказалось, что они не нарушили ни одного из этих условий? Ответ обосновать минимумом рассуждений — повествовательных утверждений об одном из городов или об одном из путешественников.

11. *Петя, Вася, Дима, Коля* и *Гена* хотят (каждый по отдельности) поехать в одну из перечисленных ниже стран, где они не были. При этом:
— *Петя* был в *Китае* и *Англии*;
— *Вася* был в *Китае* и во *Франции*;
— *Дима* хочет поехать только в *США*;
— *Коля* не был нигде;
— *Гена* был во *Франции* и *Алжире*.
Найти минимумом простых рассуждений, кто и куда должен поехать в соответствии с их пожеланиями. За простое рассуждение принять одно повествовательное предложение относительно страны и юноши.

Литература

1. *В.Казиев.* Информатика. Часть 1. Теория; Часть 2. Задачи; Часть 3. Лабораторный практикум. Нальчик, 1997.
2. *В.Казиев.* Развивающие задачи. ИНФО, № 3, 1997.
3. *В.Казиев.* Развивающие задачи. ИНФО, № 2, 1998.
4. *В.Казиев.* Развивающие задачи. Информатика в уроках и задачах (приложение к ИНФО), № 2, 1998.
5. *В.Казиев.* Развивающие задачи. Информатика в уроках и задачах (приложение к ИНФО), № 1, 1999.
6. *В.Казиев.* Информатика в примерах и задачах. Нальчик, 1998.

Продолжение следует

"Почему многие из тех, кто связан с компьютерами, упорно не хотят научиться работать на клавиатуре, используя слепой десятипальцевый метод, почему?"

В.В. Шахиджанян

На сервере **http://1001.vdv.ru** вы не найдете ответа на этот вопрос. Но зато сможете познакомиться с замечательной программой «Соло на клавиатуре", с помощью которой, по мнению авторов, гарантированно осваивается слепой десятипальцевый метод набора на клавиатуре компьютера.

УЗЕЛКИ НА ПАМЯТЬ

Компьютерный фокус

Д.М. Златопольский

С помощью компьютера можно не только доказывать теоремы [1], но и демонстрировать фокусы. В основе одного из компьютерных фокусов лежит известная игра "Отгадай число" [2], в которой один из двух участников задумывает целое число из некоторого интервала, а второй пытается отгадать это число, делая несколько попыток. В случае несовпадения задуманного числа и числа-ответа первый участник сообщает, какое из них больше, после чего играющий вновь называет число — и т.д. до отгадывания. Как известно, чтобы отгадать задуманное число за минимально возможное число попыток, следует использовать принцип бинарного поиска (деления интервала поиска пополам).

Фокус состоит в следующем. Фокусник — человек, сидящий у компьютера, — просит зрителей задумать и назвать любое целое число. Компьютер должен это число отгадать, выдавая на экран свои предположения. Зрители, глядя на эти предположения, называют любые слова, например, свои имена, которые фокусник вводит в компьютер с клавиатуры. Компьютер отгадывает число, причем делает это за минимально возможное число попыток.

Секрет фокуса: человек, набирая на клавиатуре очередное названное слово, незаметно для зрителей вводит в компьютер информацию о том, справа или слева в числовом ряду находится задуманное число от последнего предположения машины. Компьютер, вернее, программа, в него заложенная, — это то, что в цирке называется подсадкой (цирковой "рыжий"), а фокусник общается с компьютером, как дрессировщик с собакой, "знающей" арифметику.

Передать в программу указанную чуть выше информацию можно различными способами. В следующей программе (на ***школьном алгоритмическом языке*** [3, 4]) это делается с помощью пробела, который вводится или не вводится в конце слова.

Основные величины, используемые в программе:

`min` — левая граница интервала, в котором расположено искомое число;
`max` — то же, правая граница;
`mid` — середина этого интервала;
`p` — номер попытки;
`name` — имя одного из зрителей;
`yn` — имеет значение `"y"`, если число угадано, а иначе — `"n"`.

```
алг Фокус
нач цел min, max, mid, p, лит c, yn
  вывод нс, "Задумайте целое число в интервале от 1 до 99"
  вывод нс, "Сейчас я его отгадаю за несколько попыток!"
  нц | Приостановка программы до нажатия любой клавиши
  кц при inkey()<>""
  min:=0; max:=100
  p:=1
  нц  |Цикл "угадывания" числа
    mid:=(max+min)/2
    вывод  нс
    вывод  нс, p, "-я попытка. Вы задумали число", mid
    вывод нс, "Я прав?"
    нц
    | yn:= getkey()     | Ждем нажатия клавиши y или n
    кц при yn="y"  или yn="n"
    если yn="y"
      то
      вывод нс, " ДА !! "
      иначе
      вывод нс, " НЕТ !! "
    все
    нц | Приостановка программы до нажатия любой клавиши
    кц при inkey()<>""
    если yn="n"  |Если ответ неверный
      то
      вывод нс, "Кто это сказал? Как Вас зовут?"
      ввод нс, name
      если name[длин(name)]=" "   |если последний символ – пробел
        то
        min:=mid
        иначе
        max:=mid
      все
```

```
    | | | p:=p+1
    | все
    кц при yn="y"
    вывод нс, "Число отгадано!"
    вывод нс, "Количество попыток –", p
    вывод нс, "Какой я умный!"
    нц | Приостановка программы до нажатия любой клавиши
    кц при inkey()<>""
кон
```

Примечание. Чтобы при определении задуманного числа увеличить значение min, в конец имени зрителя необходимо ввести пробел, для уменьшения значения max он не нужен.

Как показывает опыт, более эффектным является использование другого способа передачи необходимой информации в программу, а именно — выдерживание различных пауз между вводом, например, первого и второго символов имени. Для ввода при этом следует использовать средства, контролирующие нажатие одной клавиши (в школьном алгоритмическом языке это функция getkey(), в Паскале — readkey, в Бейсике — INKEY$, в Си — getch()), а также процедуры и функции, отслеживающие время.

В представленных ниже программах используются следующие основные величины (кроме употреблявшихся в приведенном варианте программы величин min, max, mid и p):

i — порядковый номер символа в имени зрителя;
t1 — время ввода первого символа;
t2 — то же для второго;
t — промежуток времени между вводом первого и второго символов;
wait — предельное значение промежутка времени между вводом первого и второго символов, в зависимости от которого происходит изменение границ интервала поиска min и max. Это значение подбирается опытным путем.

Передачу в программу информации, в зависимости от которой происходит изменение значений min и max, можно проводить по следующему правилу: если t > wait, то задуманное число больше последнего предположения компьютера (то есть должно быть увеличено значение min), в противном случае — меньше (должно быть уменьшено значение max).

Приведем соответствующие программы.

Школьный алгоритмический язык

Обращаем внимание на то, что результатом функции getkey() является литерная величина, значением которой может быть и символическое обозначение специальных клавиш (Enter, функциональных, клавиш перемещения курсора и др.).

```
алг Фокус
нач цел min, max, mid, p, t1, t2, t, wait, i, лит c, yn
    вывод нс, "Задумайте целое число в интервале от 1 до 99"
    вывод нс, "Сейчас я его отгадаю за несколько попыток !"
    нц | Приостановка программы до нажатия любой клавиши
    кц при inkey()<>""
    min:=0; max:=100
    wait:=100       |Значение  подбирается опытным путем
    p:=1
    нц  |Цикл "угадывания" числа
      mid:=(max+min)/2
      вывод  нс
      вывод  нс, p, "-я попытка. Вы задумали число", mid
      вывод нс, "Я прав?"
      нц
        yn:= getkey()     |Ждем нажатия клавиши y или n
      кц при yn="y"  или yn="n"
      если yn="y"
        то
        вывод нс, " ДА !! "
        иначе
        вывод нс, " НЕТ !! "
      все
```

```
    нц | Приостановка программы до нажатия любой клавиши
    кц при inkey()<>""
    если yn="n"  |Если ответ неверный
      то
      вывод нс, "Кто это сказал? Как Вас зовут?"
      вывод нс
      i:=1
      нц  | Цикл ввода имени
        c:= getkey()     | Ждем нажатия клавиши
        если c<>"Enter"
          то
           вывод c       | Повторяем символ на экране
        все
        выбор            | Определяем значения t1, t2, t
          при i=1: t1:=время
          при i=2: t2:=время
          t:=t2-t1
        все
        i:=i+1
      кц при c="Enter"
      |меняем границы интервала в зависимости от значений t и wait
      если t>wait
        то
        min:=mid
        иначе
        max:=mid
      все
      p:=p+1
    все
  кц при yn="y"
  вывод нс, "Число отгадано!"
  вывод нс, "Количество попыток -", p
  вывод нс, "Какой я умный!"
  нц | Приостановка программы до нажатия любой клавиши
  кц при inkey()<>""
кон
```

Язык Паскаль [5]

```
Uses crt, dos;
Var
min, max, mid, p , i: byte;
t1, t2, t:word;
h, m, s, s100:word;
e,c, YN: char;
BEGIN
clrscr;
writeln('Задумайте целое число в интервале от 1 до 99');
writeln('Сейчас я его отгадаю за несколько попыток !');
e:=ReadKey;{Приостановка программы до нажатия любой клавиши}
min:=0; max:=100;
p:=1;
repeat          {Цикл "угадывания" числа}
 mid:=(max+min) div 2;
 writeln (p, '-я попытка. Вы задумали число ', mid);
 writeln ( 'Я прав?');
 repeat {Ввод Y или N}
  YN:=UpCase(ReadKey);
 until (YN='Y') or (YN='N');
 if YN='Y' then writeln('ДА !!') else writeln('НЕТ !!');
 e:=ReadKey;{Приостановка программы до нажатия любой клавиши}
 if YN='N' then begin{Число не угадано}
  writeln ('Кто это сказал? Как Вас зовут?');
  i:=1;
  repeat        {Цикл ввода имени}
   c:= readkey;       {Ждем нажатия клавиши}
   if c<>#13 then  {Если не нажата клавиша "Enter"}
    begin
     write(c);    {повторяем символ на экране}
     case i of {определяем значения t1, t2, t}
```

```
      1: begin
          Gettime(h, m, s, s100);
           t1:=h*3600+m*60+s+s100 div 100
          end;
       2: begin
           Gettime(h, m, s, s100);
           t2:=h*3600+m*60+s+s100 div 100;
           t:=t2-t1
          end
     end; {case i}
    end;
   i:=i+1
  until c=#13; {Цикл ввода имени}
  {меняем границы интервала}
  if t>1 then min:=mid else max:=mid;
  p:=p+1;
                end;{Число не угадано}
 until YN='Y'; {Цикл "угадывания" числа}
 {Число отгадано}
 writeln;
 writeln ('Число отгадано!');
 writeln ('Количество попыток - ', p);
 writeln ('Какой я умный!');
 e:=ReadKey;{Приостановка программы до нажатия любой клавиши}
END.
```

Язык Бейсик (вариант QuickBasic [6])

Здесь следует учесть, что функция INKEY$ не ждет нажатия очередной клавиши. Поэтому первые два символа имени вводятся индивидуально. Условие завершения циклов ввода этих символов — символ должен быть буквой (для простоты, латинской).

```
DEFINT M-P
DEFLNG T
DEFSTR C, Y
CLS
PRINT "Задумайте целое число в интервале от 1 до 99"
PRINT "Сейчас я его отгадаю за несколько попыток !"
SLEEP 'Остановка программы до нажатия любой клавиши
min = 0: max = 100
p = 1
DO                   'Цикл "угадывания" числа
 mid = (max + min) \ 2
 PRINT
 PRINT p; "-я попытка. Вы задумали число "; mid
 PRINT "Я прав?"
 DO 'Ввод Y или N
  YN = UCASE$(INKEY$)
 LOOP UNTIL YN = "Y" OR YN = "N"
 IF YN = "Y" THEN PRINT "ДА !!" ELSE PRINT "НЕТ !!"
 SLEEP
 IF YN = "N" THEN 'Число не угадано
  PRINT "Кто это сказал? Как Вас зовут?"
  DO 'Ввод первого символа имени
   c = INKEY$: cu = UCASE$(c)
  LOOP UNTIL cu >= "A" AND cu <= "Z"
  PRINT c;
  t1 = TIMER
  DO 'Ввод второго символа имени
   c = INKEY$: cu = UCASE$(c)
  LOOP UNTIL cu >= "A" AND cu <= "Z"
  PRINT c;
  t2 = TIMER
  t = t2 - t1
  DO 'Ввод остальных символов имени
   c = INKEY$
   IF c <> "" THEN PRINT c;
  LOOP UNTIL c = CHR$(13) 'Ввод прекращается при нажатии Enter
  'меняем границы интервала
```

```
  IF t > 2 THEN min = mid ELSE max = mid
  p = p + 1
 END IF           'Число не угадано
LOOP UNTIL YN = "Y" 'Цикл "угадывания" числа
'Число отгадано
PRINT "Число отгадано!"
PRINT "Количество попыток - "; p
PRINT "Какой я умный!"
SLEEP
END
```

Язык Си [7]

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
#include<string.h>
void main()
{int min=0, max=100, mid, p=1 , i;
 char c,YN; long t,t1,t2;
 clrscr();
 printf("\nЗадумайте целое число в интервале от 1 до 99");
 printf("\nСейчас я его отгадаю за несколько попыток !");
 getch();/*Приостановка программы до нажатия любой клавиши*/
 do                  /*Цикл "угадывания" числа*/
  {mid=(max+min)/2;
   printf("\n");
   printf("\n%d -я попытка. Вы задумали число %d",p,mid);
   printf("\nЯ прав?");
   do /*Ввод Y или N*/
    {YN=getch(); strupr(&YN);}
   while (YN != 'Y' && YN != 'N');
   if (YN == 'Y')
    printf("\nДА !!\n"); else printf("\nНЕТ !!");
   getch();
   if (YN == 'N') /*Число не угадано*/
    {printf("\nКто это сказал? Как Вас зовут?\n");
     i=1;
     do
       {c=getch();      /*Цикл ввода имени*/
       /*Если нажата клавиша "Enter", то выход из цикла*/
       if (c=='\r') break;
        printf("%c",c);/*повторяем символ на экране*/
       /*определяем значения t1, t2, t*/
       if (i==1) time( &t1);
       if (i==2) {time( &t2); t=t2-t1;}
       i++;
      }
     while (c != '\r');/*Цикл ввода имени*/
     /*меняем границы интервала*/
     if (t>1) min=mid; else max=mid;
     p++;
    }             /*Число не угадано*/
  }
 while (YN == 'N'); /*Цикл "угадывания" числа*/
 /*Число отгадано*/
 printf("\nЧисло отгадано!");
 printf("\nКоличество попыток - %d", p);
 printf("\nКакой я умный!");
 getch();
}
```

ЛИТЕРАТУРА

1. *Д.М. Златопольский.* Все дороги ведут к... 153, или Компьютер доказывает теорему. Информатика, 1998, № 33.
2. *Д.М. Златопольский.* Алгоритмизация простейших игр. Информатика и образование, 1998, № 6.
3. *А.Г. Кушниренко, Г.В. Лебедев, Р.А. Сворень.* Основы информатики и вычислительной техники. М.: Просвещение, 1990.
4. *М.Г. Эпиктетов.* Почему школьный алгоритмический? Информатика, 1995, № 33.
5. *В.В. Фаронов.* Программирование на персональных ЭВМ в среде Турбо Паскаль. М.: Изд-во МГТУ, 1992.
6. *В.В. Зельднер.* QuickBasic 4.5. М.: ABF, 1994.
7. *М.И. Болски.* Язык программирования Си. М.: Радио и связь, 1988.

Программа на языке Си подготовлена редакцией

Элементы математической логики в курсе школьной информатики

Л.Л. Акуленко-Босова

Статьи данного цикла обобщают пятилетний опыт проведения факультативных занятий с учащимися 9-, 10-, 11-х классов общеобразовательной средней школы.

Материал адресован учителям информатики, математики, а также ученикам, которые захотят изучить эту тему самостоятельно.

1. Об истории логики

Термин логика происходит от древнегреческого *logos*, означающего "слово, мысль, понятие, рассуждение, закон".

Логика — наука, изучающая законы и формы мышления. В логике мышление рассматривается как инструмент познания окружающего мира.

Основоположником логики считают Аристотеля, жившего в 384—322 гг. до н.э. В своих книгах ("Категории", "Первая аналитика", "Вторая аналитика" и др.) Аристотель подверг анализу человеческое мышление и его формы: понятие, суждение, умозаключение. В определении Аристотеля логика представляет собой науку о выводе одних умозаключений из других сообразно их логической форме. В соответствии с этим логику Аристотеля называют формальной.

В своих трудах Аристотель впервые обосновал один из важнейших разделов логики — **учение о суждениях и силлогизмах.**

В аристотелевской логике рассматривались четыре вида суждений:

1. **Общеутвердительные** — "все *S* суть *P*". Например, "Все рыбы — животные", "Все квадраты — прямоугольники".
2. **Общеотрицательные** — "Никакое *S* не есть *P*". Например, "Никакие рыбы не являются птицами", "Никакие треугольники не являются квадратами".
3. **Частноутвердительные** — "Некоторые *S* суть *P*". Например, "Некоторые прямоугольники — квадраты", "Некоторые люди — испанцы".
4. **Частноотрицательные** — "Некоторые *S* не суть *P*". Например, "Некоторые грибы несъедобны", "Некоторые треугольники не равнобедренны".

Аристотелевские **силлогизмы** представляют собой схемы логических выводов, состоящих из трех суждений одного из четырех перечисленных видов: два первых суждения — посылки, третье — заключение.

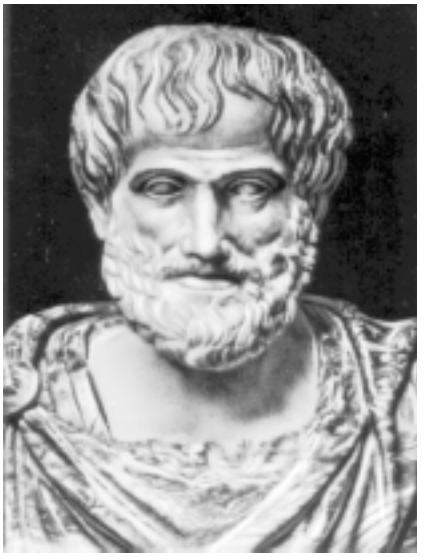
Вот простой пример:

"Все птицы — животные",
"Все воробьи — птицы", следовательно,

"Все воробьи — животные".

В силлогистике говорится о том, какие приемы рассуждения позволяют делать правильные умозаключения, а какие приводят к ложным выводам и поэтому недопустимы.

Родом Аристотель был из города Стагира на фракийском побережье полуострова Халькидика. Его отец был врачом и другом македонского царя Аминта II. Аристотель рос и учился вместе с сыном Аминта — будущим царем Филиппом II Македонским, и на протяжении всей жизни его судьба была тесно связана с македонским царским домом. В возрасте 18 лет Аристотель отправился в Афины к великому мыслителю Платону и провел в его школе около 20 лет. Он был самым способным из учеников Платона, глубоко усвоившим его знания и идеи, но далеко не всегда согласным со своим учителем. В 343 г. до н.э. царь Филипп приглашает Аристотеля стать наставником своего сына Александра. Когда через несколько лет Александр сам становится царем, знаменитым Александром Македонским, Аристотель возвращается в Афины и собирает вокруг себя учащуюся молодежь, которой читает курсы различных наук. В 323 г. до н.э. умер Александр Македонский и в Афинах победила антимакедонская партия. Аристотель, как друг и учитель Александра, вынужден был покинуть Афины. Год спустя он умер на острове Эвбея.



Приведем один шуточный пример, известный из глубокой древности, показывающий, что приемы логического мышления не столь просты, как это может показаться на первый взгляд.

Один житель острова Крит сказал: "Все критяне лжецы". Но ведь сам он критянин и, значит, лжец. Следовательно, он сказал неправду. Выходит, все критяне правдивы. Но тогда и он правдив и, соответственно, сказал правду. А если он сказал правду, то получается, что все критяне все-таки лжецы. Значит, и он лжец и поэтому сказал неправду. Посему все критяне правдивы. И он правдив, но все критяне лжецы. Тогда и он лжец...

Как же выбраться из заколдованного круга?

Такие рассуждения, приводящие к явно нелепым выводам, называются **софизмами**. Знание логики позволяет достаточно быстро раскрыть любой софизм, показать, где мы неправильно рассуждали, где применили неправильный прием умозаключения. Этот софизм мы раскроем дальше, на стр. 23.

Силлогистика Аристотеля зачастую называется просто традиционной формальной логикой, в отличие от современной формальной логики, возникшей в XIX в. и базирующейся на математических методах.

Первые идеи о "математизации" логики появились в XVII в. Так французский философ и математик Рене Декарт (1596—1650) считал, что человеческий разум может постигнуть истину, если будет исходить из достоверных положений, сводить сложные идеи к простым, переходить от известного и доказанного к неизвестному, избегая каких-либо пропусков в логических звеньях исследований. Таким образом, он рекомендовал в логике использовать общепринятые математические методы. В то время и другие ученые заметили, что выводы согласно определенным схемам напоминают математические выкладки при нахождении систем уравнений и неравенств. Особенно на этой стороне логических выводов настаивал великий немецкий философ и математик Готфрид Вильгельм Лейбниц (1646—1716), предложивший детальную программу логических исследований методами математики. Он писал: "Никто не должен бояться, что наблюдение над знаками уведет нас от вещей: напротив, оно приводит нас к сущности вещей".

Лейбниц предложил использовать в логике математическую символику и впервые высказал мысль о возможности применения в ней двоичной системы счисления. Так зарождалась математическая, или символическая, логика.

Логические изыскания Лейбница, существенно опередившие эпоху, оставались неизвестными до конца XIX столетия, когда они были найдены в архиве и опубликованы французским математиком Луи Кутюра. Логические исследования Лейбница были столь значительны, что и через 200 лет оказали существенное влияние на развитие математической логики.

Отцом математической логики по праву считается замечательный английский математик XIX столетия Джордж Буль (1815—1864), именем которого назван раздел математической логики — булева алгебра. Знаменитые труды Д.Буля по началам математической логики — "Математический анализ логики", "Исчисление логики" и "Исследование законов мысли" — появились в конце 40-х — начале 50-х гг. В них отразилось убеждение Буля о возможности изучения свойств математических операций, осуществляемых не обязательно над числами. Ученый говорил о символическом методе, который он применял как к изучению дифференцирования и интегрирования, так и к логическому выводу и к теоретико-вероятностным рассуждениям. Именно он построил один из разделов формальной логики в виде некоторой "алгебры", аналогичной алгебре чисел, но не сводящейся к ней.

Готфрид Вильгельм Лейбниц родился в г. Лейпциге (Саксония); его отец был профессором этики, а дед — профессором права Лейпцигского университета. В 1661 г. Лейбниц становится студентом и изучает философию, юриспруденцию и математику в университетах Лейпцига, Вены и Альтдорфа. В 1666 году он защищает сразу две диссертации на звание доцента — по юриспруденции и математике. Затем Лейбниц служит при дворах немецких князей в качестве юриста, находится на дипломатической службе. С 1676 г. и до самой смерти Лейбниц состоял советником и библиотекарем при дворе ганноверского герцога. На протяжении этих 40 лет Лейбниц вел научные исследования, публиковал научные труды, поддерживал переписку со всеми ведущими учеными эпохи.

Лейбниц был универсальным ученым, внесшим существенный вклад в философию, юриспруденцию, историю, физику и математику. Он является одним из создателей дифференциального и интегрального исчисления, комбинаторики, теории определителей. Значительна и научно-организаторская деятельность Лейбница — он был одним из основателей Прусской Академии наук в Берлине.

Большой вклад в становление и развитие математической логики внесли Августус де Морган (1806—1871), Уильям Стенли Джевонс (1835—1882), Платон Сергеевич Порецкий (1846—1907), Чарлз Сандерс Пирс (1839—1914) и др.

Математическая логика сегодня — раздел математики, логика, развиваемая математическими методами, играющая важную роль в вопросах обоснования математических теорий и нашедшая многочисленные приложения в вопросах конструирования и применения вычислительных машин.

В ЭВМ информация подвергается не только арифметической, но и логической обработке. Основу работы логических схем и устройств ЭВМ составляет специальный математический аппарат — раздел математической логики, называемый алгеброй логики.

2. Элементы алгебры логики. Высказывания и операции над ними

Высказывание — это любое предложение какого-либо языка (утверждение), содержание которого можно определить как истинное или ложное.

Примеры высказываний:

1) город Вашингтон — столица США (истинное высказывание);
2) число 2 является делителем числа 7 (ложное высказывание);
3) $3 + 5 = 2 \cdot 4$ (истинное высказывание);
4) $2 + 6 > 10$ (ложное высказывание);
5) II + VI > VIII (ложное высказывание);
6) сумма чисел 2 и 6 больше числа 8 (ложное высказывание);
7) *two plus six is eight* (истинное высказывание);
8) $F = m \cdot a$ (ложное высказывание);
9) *Na* — металл (истинное высказывание).

Высказывания, приведенные выше, являются простыми (элементарными). Простые высказывания будем обозначать заглавными латинскими буквами:

A = {Квадрат — это ромб};

B = {Волга впадает в Черное море}.

Элементарные высказывания являются кирпичиками, из которых с помощью логических операций строятся сложные высказывания. Сложные высказывания иногда называют **формулами логики высказываний**.

Примеры сложных высказываний:

1) число 6 четно или число 8 нечетно;
2) число 6 четно и число 8 нечетно;
3) таблетки от кашля "Кашлин" разрешены Минздравом РФ или капли от насморка "Мойнос" разрешены Минздравом РФ.

В алгебре логики, как и в обычной алгебре, вводится ряд операций.

Рассмотрим три основные логические операции.

КОНЪЮНКЦИЯ (лат. *conjunctio* — связываю), или логическое умножение;

— соответствует союзу и;

— обозначается символом &.

Джордж Буль родился в Линкольне (Англия) в семье мелкого торговца. Материальное положение его родителей было тяжелым, поэтому Джордж смог окончить только начальную школу для детей бедняков; в других учебных заведениях он не учился. Этим, может быть, отчасти и объясняется, что, не связанный традицией, он пошел в науке собственным путем. Буль самостоятельно изучил латынь, древнегреческий, немецкий и французский языки, изучил философские трактаты. С ранних лет Буль искал работу, оставляющую возможности для самообразования. После многих неудачных попыток Булю удалось открыть маленькую начальную школу, в которой он преподавал сам. Школьные учебники по математике привели его в ужас своей нестрогостью и нелогичностью, Буль вынужден был обратиться к сочинениям классиков науки и самостоятельно проштудировать обширные труды Лапласа и Лагранжа.

В связи с этими занятиями у него появились первые самостоятельные идеи. Результаты своих исследований Буль сообщал в письмах профессорам математики (Д.Грегори, А. де Моргану) знаменитого Кембриджского университета и вскоре получил известность как оригинально мыслящий математик. В 1849 г. в г. Корк (Ирландия) открылось новое высшее учебное заведение — Куинз колледж, по рекомендации коллег-математиков Буль получил здесь профессуру, которую сохранил до своей смерти в 1864 г. Только здесь он получил возможность не только обеспечить старость родителей, но и спокойно, без мыслей о хлебе насущном, заниматься наукой. Здесь же он женился на дочери профессора греческого языка Мери Эверест, которая много помогала Булю в работе и оставила после его смерти интересные воспоминания о своем муже; она стала матерью четырех дочерей Буля, одна из которых, Этель Лилиан Буль, в замужестве Войнич, — автор популярного в нашей стране романа "Овод".

Высказывание $A\&B$ истинно в том и только в том случае, когда одновременно истинны высказывания A и B.

Рассмотрим пример.

Пусть

A = {Учебник по информатике лежит на черном столе};

B = {Черный стол находится в кабинете № 13}.

Составим таблицу истинности.

Как это сделать?

Нам понадобятся три столбца — для значений A, B и $A\&B$. Строк нужно столько, сколько возможных сочетаний A и B. Ясно, что их четыре. Когда столбцы A и B заполнены, столбец $A\&B$ заполняется в соответствии с определением.

A	B	$A\&B$
0	0	0
0	1	0
1	0	0
1	1	1

Когда истинно высказывание $A\&B$?

ДИЗЪЮНКЦИЯ (лат. *disjunctio* — различаю), или логическое сложение;

— соответствует союзу или;

— обозначается символом $\vee$.

Высказывание $A\vee B$ ложно в том и только том случае, когда одновременно ложны высказывания A и B.

Рассмотрим пример и составим таблицу истинности.

Пусть

A = {Иванов учится на "5"};

B = {Иванов учится на "4"}.

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Когда ложно высказывание $A\vee B$?

Рассмотренные выше операции были двухместными (бинарными), т.е. выполнялись над двумя операндами (высказываниями). В алгебре логики определена и широко применяется и одноместная (унарная) операция.

ИНВЕРСИЯ (лат. *inversio* — переворачиваю), или отрицание;

— соответствует частице не;

— обозначается символом $\neg$ или надчеркиванием ($\overline{A}$).

Высказывание $\neg A$ истинно, когда A ложно, и ложно тогда, когда A истинно.

Таблица истинности инверсии:

A	$\neg A$
0	1
1	0

Рассмотрим пример.

Пусть

A = {Число 5 является делителем числа 30}.

Это истинное высказывание. Очевидно, новое высказывание

$\neg A$ = {Неверно, что число 5 является делителем числа 30} — ложно.

Высказывание B = {Число 57 делится на 4} — ложно.

А высказывание $\neg B$ = {неверно, что число 57 делится на 4} — истинно.

Здесь имеет смысл вспомнить приведенный выше софизм о критянах.

Необходимо проанализировать две взаимоисключающие возможности: либо критянин, о котором идет речь в софизме (обозначим его буквой X), правдив, либо он является лжецом. Рассмотрим обе возможности.

Допустим, что X — правдив. Тогда он сказал правду и все критяне (в том числе и он) лжецы. Приходим к противоречию.

Предположим, что X — лжец, а значит, он сказал неправду и высказывание "Все критяне — лжецы" — ложно. Но тогда истинным является отрицание этого высказывания, т.е. истинно высказывание "Не все критяне лжецы". Другими словами, среди критян имеются и правдивые люди, и лжецы (например, сам X). X сказал неправду, что и следовало ожидать от лжеца. Софизм разрешен.

Логические операции имеют следующий приоритет: действия в скобках, $\neg$, $\&$, $\vee$.

УПРАЖНЕНИЯ

1. Объясните, почему следующие предложения не являются высказываниями:

1. Какого цвета этот дом?
2. Число X не превосходит единицы.
3. $4X + 3$.
4. Посмотрите в окно.
5. Пейте томатный сок!
6. Эта тема скучна.
7. Валерий Леонтьев — самый популярный певец.

2. Даны два высказывания:

A = {Число 5 — простое},

B = {Число 4 — нечетное}.

Очевидно, что $A = 1$, $B = 0$.

В чем заключаются высказывания:
а) $\neg A$, б) $\neg B$, в) $A\&B$, г) $A \vee B$?
Какие из высказываний **а—г** истинны?

3. Рассмотрите следующие элементарные высказывания:
A = {Река Днепр впадает в Черное море},
B = {45 — простое число},
C = {Вена — столица Австрии},
D = {0 — натуральное число}.
Определите, какие из них истинные, а какие ложные.

4. По мишени произведено три выстрела. Рассмотрим высказывания
P_k = {мишень поражена k-м выстрелом}, $k = 1, 2, 3$.
Что означают следующие высказывания:
а) $P_1 \vee P_2 \vee P_3$;
б) $P_1 \,\&\, P_2 \,\&\, P_3$;
в) $\neg(P_1 \vee P_2 \vee P_3)$.

5. Найдите значения выражений:
а) $(1 \vee 1) \vee (1 \vee 0)$;
б) $((1 \vee 0) \vee 1) \vee 1$;
в) $(A \vee 1) \vee (B \vee 0)$;
г) $(0 \& 1) \& 1$;
д) $1 \& (1 \& 1) \& 1$;
е) $((1 \vee 0) \& (1 \& 1)) \& (0 \vee 1)$;
ж) $((1 \& A) \vee (B \& 0)) \vee 1$;
з) $((1 \& 1) \vee 0) \& (0 \vee 1)$;
и) $((0 \& 0) \vee 0) \& (1 \vee 1)$.

6. Даны три числа в различных системах счисления: $P = 23_{10}$, $B = 23_8$, $C = 1A_{16}$.
Переведите P, B и C в двоичную систему счисления и выполните поразрядно логические операции $(P \vee B) \,\&\, C$. Ответ дайте в десятичной системе счисления.

Решение
$P = 23_{10} = 10111_2$,
$B = 23_8 = 10011_2$,
$C = 1A_{16} = 11010_2$,
$P \vee B = 10111$, $(P \vee B) \,\&\, C = 10010$,
$10010_2 = 16 + 2 = 18_{10}$.

7. Поразрядное отрицание восьмиразрядного двоичного числа, записанное в десятичной системе счисления, равно 217. Определить исходное число в десятичной системе счисления.

Решение
$\neg A = 217_{10} = 128 + 64 + 16 + 8 + 1 = 11011001_2$.
$A = 00100110_2$.
$A = 32 + 4 + 2 = 38_{10}$.

8. Определите логическое произведение и логическую сумму всех двоичных чисел в диапазоне от 16_{10} до 22_{10}, включая границы. Ответ запишите в восьмеричной системе счисления.

Решение
Выпишем нужные числа друг под другом:
10000
10001
10010
10011
10100
10101
10110
Если в "вертикали" встречается хотя бы один 0, то и в логическом произведении на соответствующем месте стоит 0. Искомое произведение: $10000_2 = 20_8$.
Если в "вертикали" есть хотя бы одна 1, то и в логической сумме на соответствующем месте будет стоять 1. Искомая сумма: $10111_2 = 27_8$.

3. Таблицы истинности

Введенные нами три логические операции дают возможность из простых высказываний строить сложные. Всякое сложное высказывание принимает значение 1 или 0 в зависимости от значения простых высказываний, из которых оно построено.
Дадим точное определение таблицы истинности.
Таблицу, показывающую, какие значения принимает сложное высказывание при всех сочетаниях (наборах) значений входящих в него простых высказываний, называют **таблицей истинности** сложного высказывания.
Сложные высказывания часто называют **формулами логики высказываний**.
Построить таблицу истинности достаточно просто.

Алгоритм построения таблицы истинности:

1) подсчитать количество переменных n в формуле;
2) определить число строк в таблице $m = 2^n$;
3) подсчитать количество логических операций в формуле;
4) установить последовательность выполнения логических операций с учетом скобок и приоритетов;
5) определить количество столбцов в таблице: число переменных + число операций;
6) выписать наборы входных переменных с учетом того, что они представляют собой натуральный ряд n-разрядных двоичных чисел от 0 до $2^n - 1$;
7) провести заполнение таблицы истинности по столбикам, выполняя логические операции в соответствии с установленной в п. 4 последовательностью.

Пример

Для формулы $A \& (B \vee \overline{B} \& \overline{C})$ построить таблицу истинности.

A	B	C	$\overline{B}$	$\overline{C}$	$\overline{B} \& \overline{C}$	$B \vee \overline{B} \& \overline{C}$	$A \& (B \vee \overline{B} \& \overline{C})$
0	0	0	1	1	1	1	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	0
0	1	1	0	0	0	1	0
1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0
1	1	0	0	1	0	1	1
1	1	1	0	0	0	1	1

Дополнение 1

Наборы входных переменных во избежание ошибок иногда рекомендуют перечислять следующим образом:

а) определить количество наборов входных переменных;

б) разделить колонку значений первой переменной пополам и заполнить верхнюю часть колонки 0, а нижнюю — 1;

в) разделить колонку значений второй переменной на четыре части и заполнить каждую четверть чередующимися группами 0 или 1, начиная с группы 0;

г) продолжать деление колонок значений последующих переменных на 8, 16 и т.д. частей и заполнение их группами 0 или 1 до тех пор, пока группы 0 и 1 не будут состоять из одного символа.

Дополнение 2

Рассмотрим на примере составление другой таблицы истинности.

Под формулой $(X1\&X2) \vee (\neg X1 \vee X2)$ подпишем столбцы возможных значений под каждой из переменных ($X1$ и $X2$); последовательно (по приоритету операций) выпишем столбцы значений операций.

$(X1$	$\&$	$X2)$	$\vee$	$(\neg$	$X1$	$\vee$	$X2)$
0	0	0	1	1	0	1	0
0	0	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	1	1	1	0	1	1	1

Дополнение 3

Можно писать $\&X1X2$ вместо $X1\&X2$; $\vee X1X2$ вместо $X1 \vee X2$.

Тогда приведенную формулу можно записать как $\vee \&X1X2 \vee \neg X1X2$.

Упражнение

Составьте таблицы истинности для следующих формул логики высказываний:

а) $A\&(A \vee B \vee C)$;

б) $B \vee (B\&A)$;

в) $(\overline{B \vee A}) \& (A \& C)$;

г) $\neg B\&(A \vee C)$.

Продолжение следует

Третьи Соловейчиковские чтения

Ярмарка педагогических идей
Школа сотрудничества

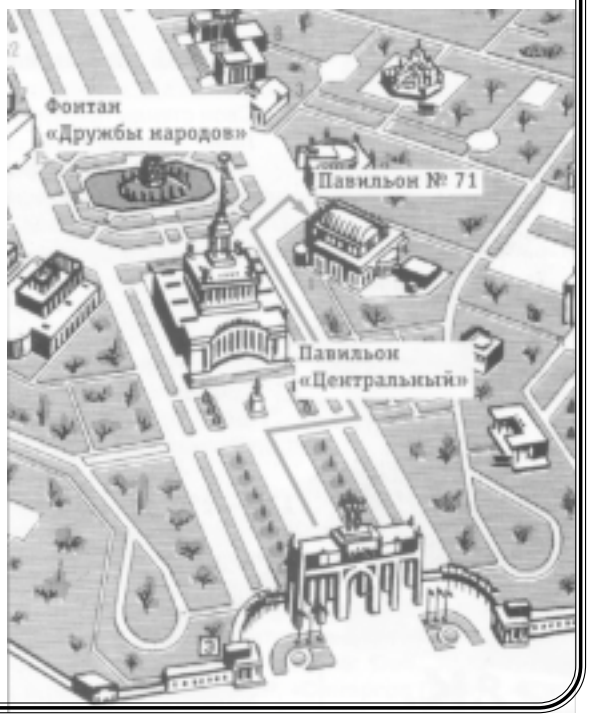
Дорогие коллеги!

Приглашаем вас, учителей и администрацию вашей школы, а также родителей ваших учеников посетить ЯРМАРКУ ПЕДАГОГИЧЕСКИХ ИДЕЙ. Она будет проходить 1 и 2 октября в ВВЦ в рамках Третьих Соловейчиковских чтений.

Мы будем рады встрече с вами и надеемся, что вам будет по-настоящему интересно.

На Третьих Соловейчиковских чтениях вы также сможете встретиться с сотрудниками редакции нашей газеты, узнать о планах публикаций, приобрести различные номера “Информатики”.

Ждем вас 1 и 2 октября
в павильоне 71 ВВЦ с 10 до 18 часов



Сетевая школа Роботландии. Заметки администратора

А.А. Дуванов

Четвертый учебный год открываются в Роботландии виртуальные аудитории для учителей и школьников, предлагая по выбору восемь различных обучающих курсов.

Хочется через газету, которая является одним из спонсоров сетевого Роботландского университета, поделиться опытом организации занятий и предоставить заинтересованным читателям информацию о наборе на новый учебный год.

Приметы роботландской школы

С точки зрения учителя, наш университет — это не совсем обычные по форме, но вполне привычные по административным итогам курсы повышения квалификации: после успешного обучения педагоги получают удостоверение как от образовательного учреждения "Роботландия+", так и от Российской Академии повышения квалификации и переподготовки работников образования.

Для детей занятия в рамках сетевого "колхоза" — это не только глубокое освоение любимых тем школьных предметов, но и спортивный азарт конкурсов, продуктивное общение со сверстниками: не "вообще" (в воздух), а по необходимости (по теме), в рамках общей курсовой деятельности.

Характерные черты нашей школы:

1. Особенность инструментальной среды курсов повышения квалификации для учителя: обучение совместно с детским коллективом.
2. Активная практическая деятельность.
3. Турнирный цикл обучения для детей.
4. Коллективная работа детей над творческим учебным проектом.
5. Дидактическая полезность турнирных разработок и приближенность "учебной" информатики к "большой".
6. Развитые горизонтальные связи (педагогическая и детская конференции, перекрестные проверки турнирных работ).
7. Постоянная обратная связь с университетом.
8. Основной канал связи — электронная почта.

Учителя проходят обучение на своих постоянных рабочих местах. Группа школьников, с которыми учителя занимаются на курсах, служит для них как бы частью инструментальной среды: полученные предметные знания и методические рекомендации непосредственно апробируются в детском коллективе.

Учитель и курируемый им детский коллектив образуют в рамках Университета сетевой узел — команду, которая связана с куратором и другими командами в едином учебном информационном пространстве.

К инструментальной среде университета относятся также:

— учебники Роботландии и методические пособия;
— программные средства Роботландии;
— постоянно действующая сетевая конференция;
— "горячая" информационная линия с куратором курса.

Активная деятельность учителей и детей лежит в основе обучения. Учебная информатика максимально приближена к "настоящей", "большой". В частности, конструкторские конкурсы имеют итогом сборники дидактических материалов, которые без всяких скидок можно использовать в обычном обучении на различных предметных уроках.

Как происходит обучение

От конкурса до конкурса. А в промежутках — обычные для факультатива занятия детей с учителем и не совсем обычные курсовые обсуждения в рамках постоянно действующей сетевой конференции.

Команды получают программные средства, учебники, методические пособия, календарные планы (все это в электронном виде) и приступают к работе.

Календарные планы ориентированы на одно аудиторное занятие в неделю. Это необходимый минимум. Детям придется много работать самостоятельно. Идеально, когда команда собирается два раза в неделю.

Процесс обучения можно изобразить такой схемой.

Из таких элементов строится цикл обучения: на фоне сетевой конференции дети изучают тему, потом готовятся к конкурсу (решают задания, подобные конкурсным, или изучают проекты, созданные детьми), затем — эмоциональный всплеск: конкурс! Конкурс — это всегда праздник с литрами адреналина в крови, турнирными таблицами, призами для победителей.

Перекрестные проверки

Большое внимание в нашей системе обучения уделяется перекрестной проверке детьми работ друг друга. Дети из Находки проверяют работы детей из Перми, Самары и Челябинска.

Возможно, первая реакция стороннего наблюдателя будет такой:

"Ага! Преподаватели Университета большие хитрецы — облегчают себе жизнь за счет детского труда!"

Это неправда. Преподаватели проверяют сами все конкурсные работы и на каждую пишут рецензию.

Мы на самом деле считаем перекрестную проверку важным элементом обучения, и многолетний практический опыт доказал это. Проверяя "чужие" работы, дети по-другому начинают относиться к своим решениям, и в итоге от такой "учительской" функции материал темы усваивается гораздо лучше.

Выполняя перекрестную проверку, дети:

— учатся пользоваться формальными оценочными критериями, порой по достаточно сложным формулам с различными весовыми коэффициентами;
— отслеживают мысль автора работы по описанию решения;
— учатся находить и "доказывать" ошибки;
— проникают в фактический материал темы на более высоком уровне (уровне педагога);
— знакомятся с разными подходами к решению одной и той же задачи;
— получают дополнительное ощущение коллективности труда в сетевом пространстве.

Практическая полезность детских проектов

Активная деятельность детей лежит в основе обучения в рамках нашего Университета. Причем мы стремимся приблизить учебную информатику к "настоящей", "большой". В частности, конструкторские конкурсы, как правило, имеют итогом сборник дидактических материалов, который, как уже говорилось, без всяких скидок можно использовать в обычной школе для обучения.

Проверить этот тезис очень легко. Перепишите примеры детских работ прошлого учебного года. Вероятно, эти материалы, скопированные из любопытства, вы захотите использовать в учебной работе сами или порекомендуете коллегам-предметникам.

Сборник работ по теме "Клеточка" (табличный классификатор):

ftp://ftp.botik.ru/rented/robot/univer/2/TAB.EXE (220 Kb)

Сборник работ по теме "Собеседник" (диалоговые программы и тесты):

ftp://ftp.botik.ru/rented/robot/univer/3/2/SOBSB.EXE (370 Kb)

Проект — эффективная форма учебной деятельности

Работа детей над достаточно большими и серьезными проектами как форма учебной деятельности, апробированная еще на фазе создания и внедрения курса информатики "Роботландия", доказала свою эффективность и в сетевом Университете.

Во-первых, обучение, с точки зрения ученика, уходит на второй план, являясь для него не мотивом, а поводом для получения конкретных, необходимых для выполнения проекта знаний и умений. Основной мотив — создание законченного компьютерного приложения, имеющего практическую ценность (компьютерная сказка, обучающая программа, справочник, тест, игра).

Во-вторых, ученик овладевает компьютерным инструментарием в комплексе, параллельно осваивая разные способы создания компьютерных объектов, имеет возможность проводить неотсроченные по времени аналогии, сравнивая типы создаваемых информационных объектов, интерфейсы программ, методы проектирования.

В-третьих, конечный продукт, созданный детьми в интегрированной среде конструирования, приближен по внешнему виду к профессиональным продуктам. Это достигается перекладыванием на систему основных операций по созданию интерфейса проектируемого приложения. Таким образом, ребенок получает внешне привлекательный продукт, не отвлекаясь от поставленной задачи на технические детали и затрачивая на техническую работу минимальное время.

В-четвертых, ученик во время работы осваивает технологии проектирования сложных систем. За основу предлагается технология "сверху вниз" (метод постепенной детализации).

В-пятых, появляется возможность естественным образом моделировать работу детей в творческих коллективах, работающих над одним большим проектом, наделяя членов коллектива функциональными обязанностями с учетом интересов ребенка.

Программирование в Университете

Есть, конечно, конкурсы другого рода. Их результаты чисто учебные, своего рода испытание в конце работы над темой. К таким состязаниям относятся турниры по программированию.

С задачами прошлых лет и их решениями вы можете познакомиться, скопировав книгу А.А. Дуванова "Роботландские сетевые турниры — 97, 98. Выпуск 1". Книга расположена по адресу:

ftp://ftp.botik.ru/rented/robot/kurs/sbornik1.arj (82 Kb)

Турнирные задачи по программированию этого года вы можете взять по адресу:

Задачи Кукарачи:

ftp://ftp.botik.ru/rented/robot/univer/3/1/TASKSCOC.ARJ (5 Kb)

Задачи Корректора:

ftp://ftp.botik.ru/rented/robot/univer/3/1/TASKSKOR.ARJ (5 Kb)

Кукарача и Корректор — это исполнители с примитивными средами и очень простым процедурным языком программирования. Однако задания, которые предлагаются детям, занимательны, достаточно сложны и часто связаны с обычными для программистов-профессионалов темами.

Помимо чисто прагматической цели — научить детей общим и частным методам программирования, Университет всячески пытается увлечь детей этой специфичной и очень эмоциональной сферой человеческой деятельности.

В шестом классе брат, студент радиотехнического института, подарил мне книгу Айсберга "Радио — это очень просто". Книга прочиталась с радостным увлечением, примерно с таким же, с каким читались в то время книги Фенимора Купера. И вот результат: четыре транзистора, катушка, диод, пара сопротивлений, конденсатор, динамик. Все это было размещено на деревянной дощечке как на постаменте и тщательно спаяно красивыми цветными проводками. И незабываемый шок, когда все эти металлические паучки и проволочки на деревяшке вдруг ожили и стали говорить человеческим голосом. Книга Айсберга воспринималась как научная фантастика, как уэллсовское описание машины времени. И вдруг: заработало! Это было волшебство, в это было невозможно поверить! Заводские радиоприемники, привычные с детства, ничуть не мешали остроте восприятия чуда. Они жили своей отдельной жизнью, как природа вокруг. А это чудо — рукотворное, сделанное собственными руками.

Пережитое потрясение запомнилось на всю жизнь. Однако потом удалось испытать подобное чувство еще раз: когда заработала первая программа. Оказалось, что программирование — это школа волшебства в чистом виде, без канифоли, паяльников, проводочков и дощечек. Голая мысль, продукт чистого разума, воплощенные в несколько строчек заклинаний на специальном языке для посвященных, позволили вдохнуть в железный ящик жизнь, и он начал вести себя разумно. С появлением персональных ЭВМ ощущение волшебства усилилось во много раз. Одно дело, когда "разум" программы читается по распечаткам с АЦПУ (алфавитно-цифровое печатающее устройство), и совсем другое, когда программа монтирует "душу" в железный ящик на столе и он, этот ящик, начинает общаться с тобой в "живом" диалоге.

Программирование представлено в Университете двумя курсами.

Первый, инструментально связанный со средами Кукарачи и Корректора, называется "Азы программирования". Но это не означает, что он для тех, кто не написал еще ни одной программы. На самом деле этот курс для ребят, которые уже знакомы с такими понятиями, как "исполнитель", "команда", "алгоритм", "ветвление", "цикл", и самостоятельно написали несколько простых программ. После того как человек напишет несколько программ, он начинает понимать, что "программирование — это обыкновенное волшебство" или "программирование — это непонятно, скучно и хуже математики". Так вот, "Азы" для тех ребят, которые уже поняли, что программирование — это то, что им нужно, или близки к этому. Другие (очень хорошие) ребята, которым не понравилось программирование, найдут для себя другое, интересное им, не менее полезное дело.

Задачи, которые решаются в примитивных средах Кукарачи и Корректора, совсем не просты. А главное, имеют прямое отношение к большой информатике и в силу этого способствуют формированию алгоритмического мышления будущих программистов.

В этом смысле название курса "Азы" означает лишь, что ребята сделают первые шаги в программировании, а совсем не то, что обучение закончится освоением алгоритмических структур, на базе которых будет построено несколько простых программ, бесцельно "гоняющих" роботов по среде. Программирование будет весьма содержательным. А по "заковыристости" задания вызывают повышенный интерес даже у студентов вузов.

В подтверждение серьезности "Азов" ниже приводится несколько тем, которые рассматриваются в теории и практике задач в средах роботландских исполнителей Кукарача и Корректор.

1. Рекурсивные приемы программирования и подводные камни рекурсии.
2. Нотация Бэкуса—Наура. Построение и анализ рекурсивных определений.
3. Лексический анализ выражений. Построение диаграмм переходов.
4. Программирование простых трансляторов. Польская запись арифметического выражения. Перевод выражения в польскую запись с последующим вычислением на стеке.
5. Практические приемы тестирования и отладки программ.

В этом году "Азы" дополнены "Буками программирования". Этот курс для ребят поопытнее (и, возможно, постарше). Писать программы можно на любом

языке (производственном или школьном), к которому тяготеет школьник (или учитель). Сам язык не является объектом изучения, а рассматривается как инструментарий. Куратор выбрал для занятий этого года следующие темы:

— целые числа;
— построение последовательностей;
— анализ последовательностей;
— элементы динамического программирования;
— элементы комбинаторики.

Изучение Интернет-технологий

Для нормальной работы в Университете каждая команда должна иметь электронный почтовый ящик. Этот информационный канал является основным для связи с куратором курса и другими командами.

Одного ящика на команду вполне достаточно для всех курсов, кроме курса "Введение в Интернет".

Дело в том, что дети, входящие в команду, должны активно использовать электронную почту для выполнения разных сетевых проектов. Один электронный ящик может оказаться серьезным препятствием для организации нормальных занятий по этому курсу.

Сетевой практикум предполагает учебную переписку, активное участие в курсовой конференции, учебные игры с роботландскими сетевыми исполнителями, различные конкурсы и турниры.

Как показал опыт последнего учебного года, все это работает хорошо только тогда, когда дети имеют постоянный доступ к сети. Если переписка и сетевая игра затягиваются на недели, азарт и интерес быстро угасают. Конечно, можно и при наличии одного почтового ящика организовать достаточно интенсивный доступ детей к сетевому пространству, но это сложно.

И еще одно важное замечание о подводных камнях курса "Введение в Интернет".

У нас сложилось впечатление, что команды, которые в прошлом году записались на этот курс и в разное время сошли с дистанции, записались случайно.

Вероятно, подвели модное слово "Интернет" и расчет на развлечения. Когда дело дошло до вдумчивой работы и стало понятно, что чистых "зрелищ" не будет, часть команд спасовала.

Понятно, когда ребята записываются на программирование. Это осознанный выбор. Совершенно ясно, что легкой жизни они не ждут. Знают, что предстоит большая работа для ума. И наградой им служит то особое, ни с чем не сравнимое чувство причастности к программистской Магии. Это просто потрясает. Без станков, напильников, пил и топоров, всего лишь несколькими десятками ударов по клавишам создать нечто такое, что начинает жить своей жизнью, воплощая в себе интеллект создателя. Постичь это можно только упорным трудом и особым "укладом головы".

Хотя и здесь бывают просчеты. А вот модное слово "Интернет" играет роль рекламной тети, зазывая на праздник с красивыми картинками, предполагая легкий пикничок с видом на Диснейленд.

В курсе "Введение в Интернет" нет хлопушек, петард и прочей веселой интернетовской бутафории.

Идет такая же вдумчивая работа, как и на других курсах, и удовольствие можно получить только в самой этой работе и в тех знаниях и умениях, которые позволяют на конкурсах занять призовые места и ощутить себя специалистом в изучаемой области.

От учителя не требуется практически никакой подготовки для работы на этом курсе. Нужно, чтобы он изначально умел принимать и читать электронные письма университета и посылать свои. Все остальные необходимые знания и умения, а также практические советы учитель получает в университете, занимаясь вместе с детьми по учебникам Университета и получая опережающие рекомендации и инструкции.

Несколько слов о детской учебной переписке. Все технические проблемы (правильное электронное письмо, прием-отправление писем, работа с разными кодировками, посылка вложений и извлечение их из тела письма, автоматическая подпись, цитирование, использование смайликов и т.д.) решаются довольно быстро и просто. Конечно, при условии хорошей организации доступа детей к электронному почтовому ящику.

Очень много усилий тратится на проблемы, связанные с логикой изложения текста сообщения и этикой электронной переписки.

Как научить детей формализовать свои мысли в текст и структурировать изложение? Эта проблема не имеет прямой связи с Интернетом, но тем не менее является важной педагогической, общегуманитарной задачей. На наш взгляд, гораздо более важной, чем технические проблемы электронного общения. В Университете этой проблеме уделяется повышенное внимание.

Краткое описание курсов Университета

Традиционно в 1999/2000 учебном году Университет откроет свои виртуальные классы для коллективных и индивидуальных учеников.

КОЛЛЕКТИВНЫЙ СТУДЕНТ — это группа детей, работающая под руководством одного или нескольких наставников.

ИНДИВИДУАЛЬНЫЙ СТУДЕНТ — это учитель, желающий пройти обучение индивидуально, без группы детей.

В составе Университета работают следующие годовые курсы.

I. ВВЕДЕНИЕ В ИНФОРМАТИКУ

(куратор — Ю.А. Первин)

Курс только для индивидуальных учеников — учителей, преподающих или собирающихся преподавать информатику в младших классах на базе программно-методической системы "Роботландия".

II. КОМПЬЮТЕРНОЕ КОНСТРУИРОВАНИЕ

Отделение 1. Конструирование 1

(куратор — Ю.А. Первин)

Курс ориентирован на руководимые школьными учителями группы детей 4—7-х классов, изучивших основы информатики. Создание компьютерных проектов, имеющих практическое применение на школьных уроках.

Отделение 2. Конструирование 2

(куратор — Ю.А. Первин)

Для детей 5—8-х классов. Конструирование с элементами программирования на базе программных сред "Конструктор Сказок" и "Собеседник".

III. АЗЫ И БУКИ ПРОГРАММИРОВАНИЯ

Отделение 1. Азы программирования

(куратор — А.А. Дуванов)

Для детей 5—8-х классов, изучивших основы информатики (исполнитель, алгоритм, первичные инструментальные навыки работы с компьютером). Введение в программирование для увлеченных детей.

Отделение 2. Буки программирования

(куратор — Я.Н. Зайдельман)

Курс наиболее эффективен для старшеклассников. Классические задачи и методы их решения. Составление и анализ алгоритмов не "по наитию", а "по науке". Олимпиадные задачи по программированию.

IV. ИНТЕРНЕТ-ТЕХНОЛОГИИ

Отделение 1. Введение в Интернет

(куратор — А.А. Дуванов)

Для детей 5—8-х классов. Введение в сетевые коммуникации на базе электронной почты. Коммуникативные проекты и конкурсы, работа с сетевыми исполнителями.

Отделение 2. Интернет-программирование

(куратор — А.А. Дуванов)

Для старшеклассников. Создание HTML-документов с элементами программирования на JavaScript. Основное направление курса связано не с созданием интернетовских страниц, а с проектированием HTML-приложений, пригодных для использования на школьных уроках.

V. Английский с компьютером

(куратор — Н.А. Прохорова)

Методические вопросы преподавания языка с использованием компьютера рассматриваются автором апробированного многолетней учительской практикой курса английского языка — Magic Land. Проектирование и реализация компьютерных упражнений к урокам.

Как стать студентом Университета

Нужно по адресу **kurs@robotland.users.botik.ru** до 1 октября 1999 года направить заявку по следующему образцу:

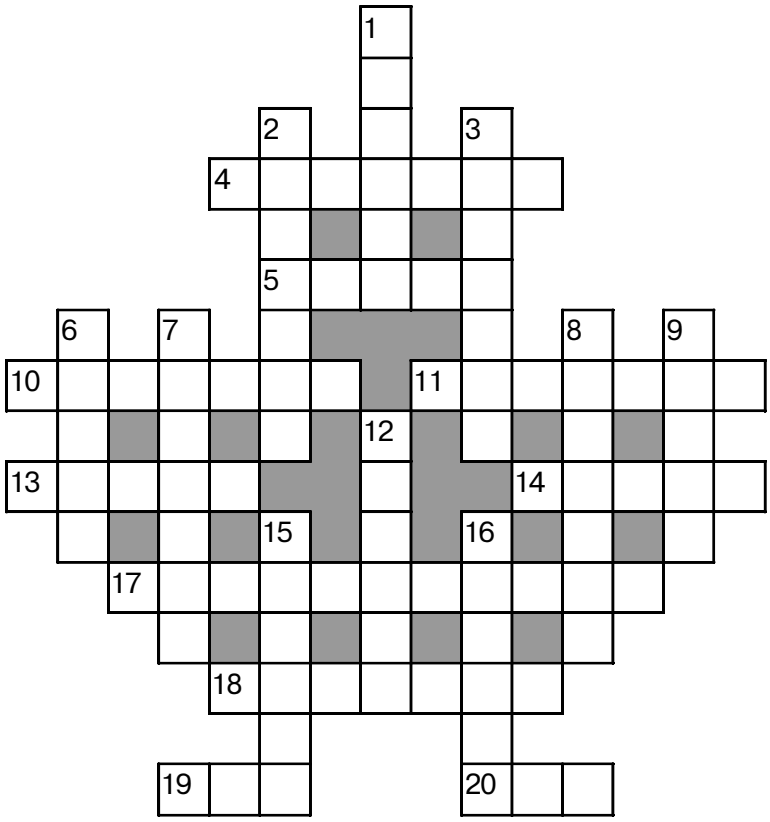
ЗАЯВКА НА УЧАСТИЕ В РОБОТЛАНДСКОМ УНИВЕРСИТЕТЕ

Предполагаемый курс (отделение):	Буки программирования
Форма обучения:	Коллективный ученик
Руководитель (обучаемый):	Иванов Петр Николаевич
Стаж работы в школе:	3 года
Директор школы:	Петров Семен Сергеевич
Электронный адрес:	ivanov@sch62.nsk.ru
Почтовый адрес школы:	603090, Новосибирск, ул. Жемчужная, 6, средняя школа 62
Телефон школы:	(3832)651220
Почтовый адрес для писем:	тот же
Число детей в группе и их возраст:	20 школьников, 8—9-е классы
Откуда получена информация:	газета "Информатика"

Для получения более подробной информации можно:
— написать по адресу **kurs@robotland.botik.ru** (администрация Университета),
— написать по адресу **auto@robotland.botik.ru** (автоответчик),
— заглянуть на сайт **http://www.botik.ru/~robot**.

Занятия в Университете начинаются 11 октября.

Кроссворд



По горизонтали:

4. Число или текст, представленные явно, т.е. значением, а не именем переменной. **5**. Пластина, на которой размещается микропроцессор или другие микросхемы. **10**. Основоположник отечественной вычислительной техники. **11**. Местоположение цифры в последовательности цифр числа. **13**. Один из необходимых элементов блок-схемы. **14**. Прямоугольник, ограничивающий окно, меню и т.п. **17**. Создание точной копии объекта (файла, блока). **18**. Множество символов. **19**. Один из последовательно сменяющих друг друга этапов выполнения алгоритма. **20**. Алгоритмический язык.

По вертикали:

1. Еженедельный источник новой информации. **2**. Устройство для отображения графической и текстовой информации на ЭВМ. **3**. Именованная группа файлов на гибком или жестком магнитном диске. **6**. Способ решения проблемы, задачи. **7**. Средство связи и передачи информации. **8**. Единица измерения объема информации. **9**. Счетная палочка на Руси. **12**. Последовательность символов, записанных в одну линию. **15**. Форма работы с ЭВМ, представляющая информацию в виде текстов или графических изображений. **16**. Язык программирования.

ОТВЕТЫ К КРОССВОРДУ

По горизонтали:

4. Литерал. **5**. Плата. **10**. Лебедев. **11**. Позиция. **13**. Конец. **14**. Рамка. **17**. Копирование. **18**. Алфавит. **19**. Шаг. **20**. Ада.

По вертикали:

1. Газета. **2**. Дисплей. **3**. Каталог. **6**. Метод. **7**. Телефон. **8**. Гигабит. **9**. Бирка. **12**. Строка. **15**. Диалог. **16**. Рапира.

Материал рубрики подготовлен **В.Г. Федориновым**

Пьер Симон Лаплас

Научное наследие Лапласа удивительно многогранно, причем самым большим его сочинением является “Небесная механика” в пяти томах. Этот трактат публиковался с перерывами в течение четверти столетия — с 1799-го по 1825 г., а сопутствовала ему развернутая популярная работа “Изложение системы мира”, вышедшая в 1796 г.

Неоценим вклад Лапласа в математику. Фундаментальными являются его работы по дифференциальным уравнениям, в частности, связанные с интегрированием методом “каскадов” уравнений в частных производных. Введенные им шаровые функции имеют разнообразное применение. В алгебре Лапласу принадлежит широко теперь известная теорема о представлении определителей при помощи сумм произведений дополнительных миноров — *теорема Лапласа*. Широко известны также *оператор Лапласа*, *преобразование Лапласа*, *интеграл Лапласа*, *уравнение Лапласа*.

Много внимания уделил Лаплас математической *теории вероятностей,* или теории случайностей, как называли ее в то время [1], от которой в последние десятилетия “отпочковался” целый ряд отраслей науки, в том числе *теория информации*. Заслуги его здесь чрезвычайно велики.

Теория вероятностей родилась из так называемых азартных игр, которые с незапамятных времен создавались именно так, чтобы в них исход опыта был независим от поддающихся наблюдению условий опыта, был чисто случайным. Само слово *азарт* (фр. *le hasard*) означает *случай*. Простейшая игра подобного типа, которую все знают, — игра в “орла и решку”. Если монета представляет собой совершенно правильный цилиндр с центром тяжести, совпадающим с ее геометрическим центром, то вероятность выпадения орла при одном бросании монеты такая же, как и для решки. Сумму вероятностей всех возможных *событий*[1] для какого-либо явления принимают за единицу. Если событие имеет вероятность, равную единице, то его надо считать достоверным, то есть таким, которое обязательно произойдет. В случае с монетой вероятность того, что при бросании ее не выпадут ни орел, ни решка, равна нулю (невозможное событие). Вероятность, что выпадет либо орел, либо решка, равна единице (достоверное событие). А вообще вероятность — это число, которое тем больше, чем более возможно событие [2].

В урне лежат сто шаров, из которых один черный, а остальные — белые. Какова вероятность того, что, вынимая наудачу один шар, мы вытянем именно черный? Понятно, что каждый шар имеет один шанс быть вынутым, а всего шансов в нашем примере — сто. Вероятность вытянуть белый шар равна девяносто девяти сотым, то есть очень близка к единице. Может, конечно, случиться, что первый же вынутый шар окажется черным, но наш математический расчет позволяет утверждать, что если подобный опыт будет продолжаться много сотен раз, то в каждых ста опытах черный шар будет вынут только один раз. Такого рода примеров, однако обычно более сложных, на практике встречается очень много. Пока не были изучены объективные законы, разные шарлатаны могли широко использовать “случай” и вовлекать в свое предприятие простодушных людей.

До Лапласа первые шаги в данной области сделали Б.Паскаль, П.Ферма, Х.Гюйгенс, Я.Бернулли, А.Муавр. Лаплас впервые дал стройное и систематическое изложение основ теории вероятностей, привел доказательство одной из форм центральной предельной теоремы *(интегральной теоремы Муавра — Лапласа)*, развил ряд замечательных приложений теории вероятностей к вопросам практики, в частности, к анализу ошибок наблюдений и измерений. Классический труд Лапласа “Аналитическая теория вероятностей” издавался трижды при его жизни — в 1812-м, 1814-м и 1820 гг.; в качестве введения к двум последним изданиям была помещена работа “Опыт философии теории вероятностей”, в которой в популярной форме разъясняются основные положения и значение этой теории.

В наше время теория вероятностей, опирающаяся на нее математическая статистика и ряд сформировавшихся на их основе дисциплин вводятся в качестве обязательных на ряде факультетов (причем далеко не только математических) высших учебных заведений, а иногда отдельные элементы теории вероятностей изучаются уже в школе. И надо помнить, что своими корнями все эти науки уходят в работы “великого геометра”[2] Пьера Симона Лапласа.

Литература

1. *Б.А. Воронцов-Вельяминов.* Лаплас. 2-е изд. М.: Наука, 1985.

2. *Е.С. Вентцель.* Теория вероятностей. 3-е изд. М.: Наука, 1964.

3. *Д.Я. Стройк.* Краткий очерк истории математики: Пер. с нем. 4-е изд. М.: Наука, 1984.

[1] Под событием в теории вероятностей понимается всякий факт, который в результате опыта может произойти или не произойти.

[2] Геометрией в XVIII веке во Франции называли математику вообще [3].

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

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

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

Учредитель: ООО “Чистые пруды”
Регистрационный номер 012868
Отпечатано в типографии ОАО ПО “Пресса-1”.
125865, ГСП, Москва, ул. Правды, 24.
Тираж 5000 экз.
Заказ №

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

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

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