«Мир информатики»
Журнал для тех, у кого информатика – любимый школьный предмет
Выпуск № 92, сентябрь 2024 г.

Семинар

Логические операции над числами

Вы, конечно, знаете о логических операциях конъюнкции (или логического умножения), дизъюнкции (логического сложения) и других. А известно ли вам, что в компьютере они используются не только при работе со сложными логическими выражениями при формировании запросов к базам данных, в условных операторах в программах, в функции ЕСЛИ в электронной таблице Microsoft Excel и т.п., но и применительно к числам? Выполняются эти операции в процессоре компьютера (поэтому их называют также логическими командами) над числами, представленным в двоичном виде. Рассмотрим те логические команды, которые выполняются над двумя числами (говорят, что у них — два операнда):

1) **AND** (русский вариант — **И**);
2) **OR** (**ИЛИ**);
3) **XOR** (от английского *eXclusive OR* — **исключающее ИЛИ**).

В отличие от арифметических операций над двумя операндами, логические команды являются *поразрядными*. Например, при сложении двух двоичных цифр возможен перенос в старший разряд, а при логических операциях все разряды рассматриваются изолированно друг от друга. Разумеется, действия над всеми разрядами выполняются параллельно и одновременно. Описанные операции называют также “битовыми” или “побитовыми”

Чтобы было легче понять, в чем заключаются указанные логические операции в процессоре, условимся называть их первый операнд “данными”, а второй — “маской”. Правила выполнения логических операций в каждом разряде представлены в таблице:

X (данные)	**Y (маска)**	**X AND Y**	**X OR Y**	**X XOR Y**
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Вам эта таблица ничего не напоминает? Да, конечно, она аналогична таблицам истинности для логических операций над величинами логического типа (с той разницей, что здесь операндами являются двоичные цифры).

Например, при $X = 101011$ и $Y = 1101$ имеем X **AND** $Y = 1001$, X **OR** $Y = 101111$, X **XOR** $Y = 100110$.

Конечно, возможна и поразрядная операция **NOT** (**НЕ**). У неё операнд один.

В языке программирования Python знаки битовых операций следующие:

И	&
ИЛИ	|
Исключающее ИЛИ	^
НЕ	~

Операция **НЕ** для числа *x* соответствует –(*x* + 1): ~5 даёт –6.

Задания для самостоятельной работы

1. Рассчитайте значения:
а) 1101101 **AND** 10101;
б) 1011011 **OR** 11001;
в) 1001101 **XOR** 10111.

2. Определите:
1) чему равен результат операции **И** при нулевой маске?

2) чему равен результат операции **ИЛИ** при маске:
а) из всех единиц?
б) в виде нуля?

3) какая операция и с какой маской позволяет выделить (получить) младший разряд двоичного числа?

4) какой получится результат, если к какому-то числу дважды применить операцию исключающего **ИЛИ**?

5) какой получится результат, если операцию исключающего **ИЛИ** применить к двум одинаковым числам?

6) какой получится результат, если к какому-то числу применить операцию исключающего **ИЛИ** с маской из всех единиц?

Во всех задачах принять, что длина маски равна длине данных.

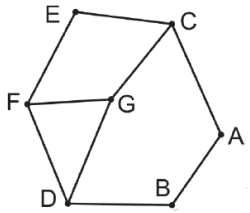
В заключение заметим, что в демонстрационных вариантах ЕГЭ по информатике нескольких последних лет представлено задание 13, в котором используется поразрядная операция «исключающее ИЛИ» (поразрядная конъюнкция).

Решение задания 1 демонстрационного варианта ЕГЭ по информатике 2025 года

Условие [1]

На рисунке схема дорог *N*-ского района изображена в виде графа, в таблице содержатся сведения о протяжённости каждой из этих дорог (в километрах).

		Номер пункта						
		1	2	3	4	5	6	7
Номер пункта	1				30	3		5
	2				21		13	
	3					39	53	2
	4	30	21					
	5	3		39			8	
	6		13	53		8		
	7	5		2				

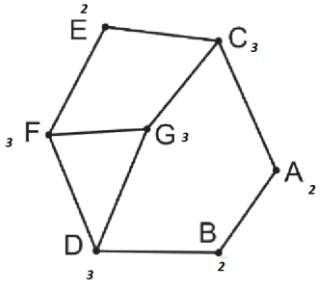


Так как таблицу и схему рисовали независимо друг от друга, нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какова сумма протяжённостей дорог из пункта *D* в пункт *G* и из пункта *A* в пункт *C*.

В ответе запишите целое число.

Решение

Для каждой вершины в графе следует определить и записать её *степень* – количество ребер, в которыми она связана

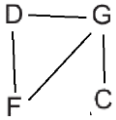


а в таблице на бумаге – аналогичную информацию (без расстояний):

	1	2	3	4	5	6	7	
1				●	●		●	*3*
2				●		●		*2*
3					●	●	●	*3*
4	●	●						*2*
5	●		●			●		*3*
6		●	●		●			*3*
7	●		●					*2*
	3	*2*	*3*	*2*	*3*	*3*	*2*	

Обратите внимание на новую строку, в которой указаны степени номеров в первой строке.

Итак, нас интересуют 4 вершины (пункта): *D*, *G*, *A* и *C*. На графе вершины *D*, *G* и *C* имеют степень 3. Их связи:

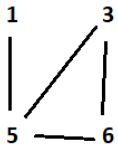


В таблице пункты со степенью 3 имеют номера 1, 3, 5 и 6. Выпишем их:

1 3

5 6

и по таблице отметим их связи:



Сравнение схем связи на графе и на таблице показывает, что пункты *D*, *G* и *C* это номера 6, 5 и 1 в таблице.

Осталось в таблице найти номер вершины *A*. Её степень – 2. Такую же степень имеют также вершины *B* и *E*. Как определить номер вершины *A*? – Она связана с вершинами с степенью 2 и 3. Но такая же особенность у вершины *B*.

В таблице степень 2 имеют пункты 2, 4 и 7. Исследуем их связи в таблице.

Пункт 2 связан с номерам 4 и 5, степень которых, согласно последней строке, – 2 и 3.

Пункт 4 связан с номерам 1 и 2, степень которых – 3 и 2.

Пункт 7 связан с номерам 1 и 2, степень которых – 3 и 3, то есть вершина *A* не может иметь в таблице номер 7.

Как определить номер (2 или 4) интересующей нас вершины *A*, установите самостоятельно.

Итак, расстояние от пункта *D* до пункта *G* – это расстояние в таблице между номерами 5 и 6 (то есть 8), а от пункта *A* до пункта *C* – 30.

Ответ: 38.

Решение задания 3 демонстрационного варианта ЕГЭ по информатике 2025 года

Условие [1][1]

Используя информацию из приведённой базы данных, определите общую массу (в кг) всех видов зефира, полученных магазинами, расположенными на проспекте Революции, за период со 2 по 10 августа включительно.

В ответе запишите только число.

Решение

Прежде всего обратим внимание на слово «полученных» и фразу «всех видов зефира», а также на единицу измерения ответа (кг).

Для нахождения ответа нужно знать:

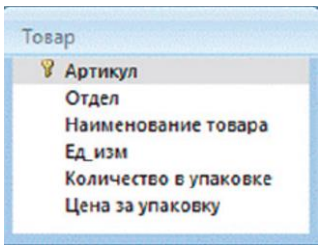
1) количество упаковок зефира каждого вида, поступивших в магазины, расположенные на проспекте Революции;

2) вес одной упаковки каждого вида.

Где находится нужная информация?

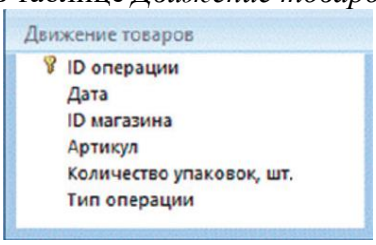
Вес одной упаковки записан в поле *Количество в упаковке* в таблице *Товар*:

[1] Полное условие см. [1].



Для нахождения веса одной упаковки того или иного товара (*Количество в упаковке*) надо знать его артикул.

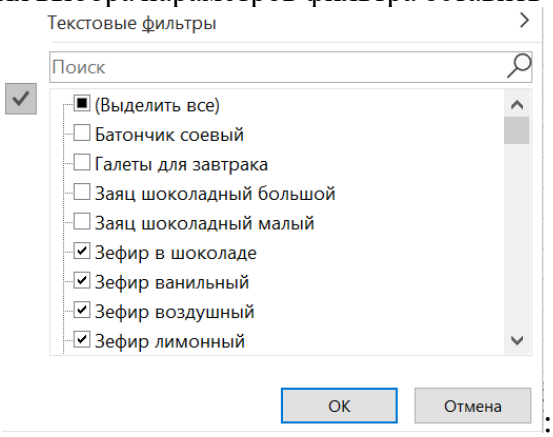
Второе нужное значение (количество упаковок того) или иного товара, поступившего в тот или иной магазин, указано в таблице *Движение товаров*:



Что надо знать для его нахождения? – Артикул товара и ID нужных магазинов. Первый записан в таблице *Товар*, а ID нужных – в таблице *Магазины*.

Итак, сначала[2] надо узнать артикулы всех видов зефира (а заодно и вес одной упаковки каждого вида) и перечень магазинов, расположенных на проспекте Революции.

1. Для определения артикулов и веса упаковок:
   - активировать лист «Товар»;
   - включить фильтр по столбцу «Наименование»;
   - в окне для выбора параметров фильтра оставить «галочку» на всех видах зефира:



:

Результат фильтрации:

Артикул	Отдел	Наименование товара	Ед_изм	Количество в упаковке	Цена за упаковку
4	Конфеты	Зефир в шоколаде	грамм	250	264
5	Конфеты	Зефир ванильный	грамм	800	239
6	Конфеты	Зефир воздушный	грамм	500	179
7	Конфеты	Зефир лимонный	грамм	1000	299

Надо выписать все значения артикулов и весов одной упаковки (обратите внимание на единицу измерения!).

2. Для отбора нужных магазинов следует:
   - активировать лист «Магазин»;
   - включить фильтр по столбцу «Адрес»;

[2] До этого из условия надо выписать (или запомнить) период, за который надо провести расчёты.

- в окне для выбора параметров фильтра оставить «галочку» только на соответствующих названиях улиц.

Результат отбора:

ID магазина	Район	Адрес
M10	Октябрьский	просп. Революции, 1
M15	Октябрьский	просп. Революции, 29

После этого можно определить количества упаковок каждого вида зефира этих магазинах:

- активировать лист «Движение товаров»;
- включить фильтры:
  - по столбцу «Артикул» (со значениями 4, 5, 6 и 7);
  - по столбцу «ID магазина» (со значениями M10 и M15);
  - по столбцу «Тип операции» (со значением Поступление);
  - по столбцу «Дата» (со значениями 02, 03, 07 и 09 августа 2023);

Результат фильтрации:

A	B	C	D	E	F
ID операци	Дата	ID магазина	Артикул	Количество упаковок, ш	Тип операции
112	02.08.2023	M10	4	200	Поступление
113	02.08.2023	M10	5	200	Поступление
114	02.08.2023	M10	6	200	Поступление
115	02.08.2023	M10	7	200	Поступление
148	02.08.2023	M15	4	200	Поступление
149	02.08.2023	M15	5	200	Поступление
150	02.08.2023	M15	6	200	Поступление
151	02.08.2023	M15	7	200	Поступление
2272	09.08.2023	M10	4	300	Поступление
2273	09.08.2023	M10	5	300	Поступление
2274	09.08.2023	M10	6	300	Поступление
2275	09.08.2023	M10	7	300	Поступление
2308	09.08.2023	M15	4	300	Поступление
2309	09.08.2023	M15	5	300	Поступление
2310	09.08.2023	M15	6	300	Поступление
2311	09.08.2023	M15	7	300	Поступление

Как определить искомое общее значение общей массы всех видов зефира? – Нужно каждое количество упаковок умножить на массу одной упаковки данного вида зефира, сложить все произведения и перевести найденную сумму в килограммы. Лучший способ решения этой задачи:

1) отсортировать данные по столбцу D (Артикул);

2) в столбце G записать массу упаковок зефира с тем или иным ID:

	A	B	C	D	E	F	G
1	ID операци	Дата	ID магазина	Артикул	Количество упаковок, ш	Тип операции	
113	112	02.08.2023	M10	4	200	Поступление	250
114	148	02.08.2023	M15	4	200	Поступление	250
115	2272	09.08.2023	M10	4	300	Поступление	250
116	2308	09.08.2023	M15	4	300	Поступление	250
149	113	02.08.2023	M10	5	200	Поступление	800
150	149	02.08.2023	M15	5	200	Поступление	800
151	2273	09.08.2023	M10	5	300	Поступление	800
152	2309	09.08.2023	M15	5	300	Поступление	800
2273	114	02.08.2023	M10	6	200	Поступление	500
2274	150	02.08.2023	M15	6	200	Поступление	500
2275	2274	09.08.2023	M10	6	300	Поступление	500
2276	2310	09.08.2023	M15	6	300	Поступление	500
2309	115	02.08.2023	M10	7	200	Поступление	1000
2310	151	02.08.2023	M15	7	200	Поступление	1000
2311	2275	09.08.2023	M10	7	300	Поступление	1000
2312	2311	09.08.2023	M15	7	300	Поступление	1000

после чего в одной из свободных ячеек ввести формулу с функцией СУММПРОИЗВ:

=СУММПРОИЗВ(E113:E2312;G113:G2312)/1000

Примечание

Диапазоны ячеек в формуле вручную вводить не следует – используйте их выделение мышью.

Решение задания 11 демонстрационного варианта ЕГЭ по информатике 2025 года

Условие [1]

На предприятии каждой изготовленной детали присваивают серийный номер, содержащий десятичные цифры, 52 латинские буквы (с учётом регистра) и символы из 963-символьного специального алфавита. В базе данных для хранения каждого серийного номера отведено одинаковое и минимально возможное число байт. При этом используется посимвольное кодирование серийных номеров, все символы кодируются одинаковым и минимально возможным числом бит. Известно, что для хранения 2000 серийных номеров отведено не более 693 Кбайт памяти. Определите максимально возможную длину серийного номера. В ответе запишите только целое число.

Решение

Сколько бит требуется для кодирования одного символа серийного номера? Общее число различных символов (мощность алфавита кодирования) составляет 10 (цифры) + 52 (латинские буквы с учётом регистра) + 963 (символы специального алфавита) = 1025.

Зависимость количества бит K для кодирования от мощности алфавита следующая:

если M есть степень двойки

то

$K = log_2 M$

иначе

$K = \lfloor log_2 M \rfloor + 1$

где символы $\lfloor \quad \rfloor$ обозначают целую часть числа, записанного между ними.

В данном случае $K = 11$ бит (десяти бит недостаточно, так как $2^{10} = 1024$).

M	256	512	1024	2048	4096	8192
K, бит	8	9	10	11	12	13

Это значит, что для хранения одного серийного номера из x символов требуется $\frac{x \times 11}{8}$ байт. Обратим внимание на то, что это количество может быть нецелым, в то время как для хранения данных в памяти отводится целое количество байт.

Для хранения 2000 номеров понадобится $\frac{x \times 11}{8} \times 2000$ байт.

Так как для 2000 номеров отведено не более 693 Кбайт памяти, то можем записать неравенство:

$$\frac{x \times 11}{8} \times 2000 \leq 693 \times 1024 \qquad (*),$$

откуда $x \leq 258{,}048$, то есть максимально возможная длина серийного номера получилась равной 258. Казалось бы, – всё правильно. Но это не так! Проверим найденное значение, используя, например, электронную таблицу:

	A	B	C	D	E
1	Длина номера, x	Кол-во байт для одного номера, $\frac{x \times 11}{8}$	Требуемое целое кол-во байт	Кол-во байт для 2000 номеров	Отведено, байт (693 × 1024)
2	258	354,75	355	710000	709632
3					

Видно, что при длине серийного номера 258 символов количество байт, которое будут занимать 2000 номеров, превышает отведённый объём памяти. Это связано с отмеченной чуть выше особенностью выделения памяти целыми байтами.

При меньшем значении длины серийного номера отведённый объём не превышается:

	A	B	C	D	E
1	Длина номера, x	Кол-во байт для одного номера, $\frac{x \times 11}{8}$	Требуемое целое кол-во байт	Кол-во байт для 2000 номеров	Отведено, байт (693 × 1024)
2	258	354,75	355	710000	709632
3	257	353,375	354	708000	709632

Итак, *ответ*: 257.

Возникает вопрос: «Всегда ли нужно при решении аналогичных задач уменьшать найденное по неравенству (*) значение на 1?» – Нет, не всегда. В качестве примера рассмотрим такое же задание, как в [1], но в котором известно, что для хранения 1000 серийных номеров отведено не более 347 Кбайт памяти. В этом случае неравенство, аналогичное неравенству (*), будет иметь вид:

$$\frac{x \times 11}{8} \times 1000 \leq 347 \times 1024,$$

откуда $x \leq 258{,}048$, то есть максимально возможная длина серийного номера получилась равной 258. Проверим это значение:

	A	B	C	D	E
1	Длина номера, x	Кол-во байт для одного номера, $\frac{x \times 11}{8}$	Требуемое целое кол-во байт	Кол-во байт для 1000 номеров	Отведено, байт (347 × 1024)
2	258	354,75	355	355000	355328
3					

Итак, при длине серийного номера 258 символов количество байт, которое будут занимать 1000 номеров, не превышает отведённый объем памяти, то есть ответ в данном случае: 258.

Задания для самостоятельного выполнения

1. При регистрации в компьютерной системе каждому объекту присваивается идентификатор, содержащий только десятичные цифры и символы из 1234-символьного специального алфавита. В базе данных для хранения каждого идентификатора отведено одинаковое и минимально возможное целое число байт. При этом используется посимвольное кодирование идентификаторов, все символы кодируются одинаковым и минимально возможным количеством бит. Известно, что для хранения 65 536 идентификаторов выделено 2050 Кбайт памяти. Укажите максимально допустимую длину идентификатора пользователя.

2 [2]. На предприятии каждой изготовленной детали присваивается серийный номер, содержащий десятичные цифры и символы из 2030-символьного специального алфавита. В базе данных для хранения каждого серийного номера отведено одинаковое и минимально возможное число байт. При этом используется посимвольное кодирование, все символы кодируются одинаковым и минимально возможным числом бит. Известно, что для хранения 318 серийных номеров должно быть использовано более 67 Кбайт памяти. Определите минимально возможную длину серийного номера. В ответе запишите только целое число.

3 [2]. При регистрации в компьютерной системе каждому пользователю выдаётся идентификатор, состоящий из цифр, больших и малых символов латинского алфавита. В базе данных для хранения сведений о каждом пользователе отведено одинаковое и минимально возможное целое число байт. При этом используется посимвольное кодирование паролей, все символы кодируются одинаковым и минимально возможным количеством бит. Кроме собственно пароля идентификатора, в системе хранятся дополнительные сведения о каждом

пользователе, для чего выделено 32 байта; это число одно и то же для всех пользователей. Для хранения сведений о 314 пользователях потребовалось 12 874 байт. Определите максимально допустимую длину идентификатора в символах. В ответе запишите только целое число.

Решение задания 17 демонстрационного варианта ЕГЭ по информатике 2025 года

Условие

В файле содержится последовательность натуральных чисел. Её элементы могут принимать целые значения от 1 до 100 000 включительно. Определите количество пар последовательности, в которых остаток от деления хотя бы одного из элементов на 16 равен минимальному элементу последовательности. В ответе запишите количество найденных пар, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Решение

Так как искомые значения зависят от минимального элемента последовательности, то определить их во время чтения данных из файла невозможно. Необходимо предварительно записать все числа последовательности в массив, после чего обрабатывать его.

Итак, основные этапы программы решения задачи:

1) чтение данных из прилагаемого к заданию файла и запись их в массив *м*;

2) определение минимального элемента массива *мин*;

3) рассмотрение всех пар элементов массива и поиск пар чисел, в которых хотя бы для одного числа остаток от его деления на 16 равен минимальному элементу последовательности. Каждую найденную пару:

- учесть в общем количестве таких пар *кп*;
- для неё определить сумму значений её чисел и сравнить с максимальной суммой *макс_сум* таких пар, найденных ранее. Если текущая сумма больше, то её запомнить в качестве нового значения максимальной суммы *макс_сум*;

4) вывод ответов.

Этап 1. Чтение чисел и запись их в файл

Так как количество чисел в условии не задано, то сначала следует узнать это количество, открыв прилагаемый к заданию файл с помощью программы Блокнот и, перейдя с помощью мыши на последнюю строку файла, узнать её номер в строке состояния программы:

Можно также, не переходя на последнюю строку, выделить все строки (**Ctrl** + **A**), скопировать их (**Ctrl** + **C**) и вставить на лист электронной таблицы, после чего узнать номер последней заполненной на листе строки, перейдя к ней (**Ctrl** + ↓).

Итак, фрагмент программы с чтением чисел из файла и записью их в массив:

```
ф := открыть на чтение("demo_2025_17 txt") | Файловая переменная
| Читаем и обрабатываем строки
нц для i от 1 до 10000
  ввод ф, ст  | ст - строковое представление числа в файле
  |Преобразовываем его число и записываем в очередной элемент массива
  м[i] := лит_в_цел(ст, успех)
кц
закрыть(ф)
```

Можно также не узнавать количество чисел в файле, а при чтении использовать оператор цикла с условием и счетчик *n* количества прочитанных чисел:

```
n := 0
нц пока не конец файла(ф)
  n := n + 1
  ввод ф, ст
  м[n] := лит_в_цел(ст, успех)
кц
```

В программе на языке Python целесообразно при заполнении массива использовать метод `append()`.

Этап 2. Поиск минимального элемента массива

```
мин := 100001 | Начальное значение искомого минимума
нц для i от 1 до 10000
  если м[i] < мин
    то
      мин := м[i]
  все
кц
```

Этап 3

```
кп := 0         | Количество подходящих пар чисел
макс_сум := 0   | Начальное значение максимальной суммы
нц для i от 1 до 9999
  если mod(м[i], 16) = мин или mod(м[i + 1], 16) = мин
    то    | Встретилась пара чисел, удовлетворяющая условию
      | Учитываем её в общем количестве пар
      кп := кп + 1
      | Сравниваем сумму значений с максимальной суммой
      если м[i] + м[i + 1] > макс_сум
        то
          макс_сум := м[i] + м[i + 1]
      все
  все
кц
```

Этап 4

```
вывод нс, кп, " ", макс_сум
```

Примечание

Чтобы не писать в программе значения 10000 (дважды), 10001 и 9999, можно использовать константу

```
к = 10000
```

и записывать, соответственно, `к`, `к + 1` и `к - 1`.

Литература

1. Демонстрационный вариант контрольных измерительных материалов Единого государственного экзамена 2025 года по информатике (Демоверсии, спецификации, кодификаторы 2024. Информатика). https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-5

2. Сайт К.Ю. Полякова. https://kpolyakov.spb.ru/school/ege.htm

Проект «Возведение в степень в компьютере»

Предлагаем читателям выполнить межпредметный проект (математика + информатика), связанный с различными методам возведения в степень в компьютере. Описываются основные вопросы проекта, для выполнения которого необходимы определенные знания математики и навыки программирования. При описании компьютерных программ, которые должны быть разработаны, логика действий в них описывается на школьном алгоритмическом языке (система программирования КуМир). Русский синтаксис этого языка и большое число комментариев к программам делает приводимые методику и программы максимально понятными, и читатель сможет разработать аналогичные программы на языке программирования, которым владеет.

Казалось бы, задача возведения в степень в компьютере решается просто – чтобы возвести число a в степень n при целом положительном значении n надо умножить число a само на себя n – 1 раз. Прежде чем представлять соответствующую программу на школьном алгоритмическом языке, заметим, что в ней используется переменная величина *произв*, накапливающая промежуточные значения произведений, а искомая величина степени имеет имя *ст*.

```
алг многократное_умножение
нач цел n, i, вещ a, произв, ст
  ввод a
  ввод n
  произв := a | Начальное значение переменной произв
  нц для i от 1 до n - 1
    произв := произв * a
  кц
  ст := произв | Искомое значение степени
  вывод ст
кон
```

Но проблема заключается в том, что при возведении больших чисел в очень большие степени вычисления могут занять много времени. Поэтому были разработаны алгоритмы возведения в степень, позволяющие компьютерам проводить эту операцию с большими числами быстрее. Конечно, возникает вопрос: «Зачем нужны алгоритмы быстрого возведения в степень?». Ответ – они нужны при решении большого количества задач. Например, криптографические алгоритмы, которые используются для шифрования и расшифровки данных, оперируют числами длиной в тысячи бит и применяют операцию возведения в степень. В машинном обучении быстрое возведение в степень используется при обучении и тестировании моделей: статистических, нейросетевых, обработке изображений и др. Программы для защиты паролей и других конфиденциальных данных тоже не обходятся без возведения в степень. И этот список можно продолжать дальше.

Достаточно очевидный вариант алгоритма с меньшим количеством операций умножения (а значит, и ускорения расчётов) основан на следующих рассуждениях.

Если n – чётное число, то a^n можно определить как произведение $a^{n/2} \times a^{n/2}$. (например, $a^{100} = a^{n/50} \times a^{n/50}$). Получается, что для любого чётного n мы можем свести задачу к вдвое меньшей степени, потратив всего одну операцию умножения.

Если же n нечётно, то верно следующее: $a^n = a^{n-1} \times a$, то есть можно перейти к степени (n – 1), которая уже будет чётной.

Итак, мы нашли следующую зависимость для подсчёта a^n:

если n – чётное число,

то

$a^n = a^{n/2} \times a^{n/2}$

иначе

$a^n = a^{n-1} \times a$

Так как за каждые два перехода значение n гарантированно уменьшается хотя бы в два раза, всего будет не более $2 \times \log_2 n$ переходов, прежде чем мы придём к $n = 1$. Каждый переход требует ровно одно умножение. Для сравнения – при стандартном способе возведения в степень требуется $n - 1$ операций умножения.

Для программной реализации полученной зависимости можем использовать так называемую «рекурсию». Напомним, что рекурсивной (использующей рекурсию) называется функция, обращающаяся как к вспомогательной к самой себе, но с другими аргументами).

Соответствующая рекурсивная функция `ст`:

```
алг вещ ст(арг вещ a, цел n)
нач
  если n = 1
    то
      знач := a    | Значение функции
    иначе | Рекурсивные вызовы функции ст
          | Вызываем эту же функцию, но с другими аргументами
      если mod(n, 2) = 0
        то
          знач := ст(a, div(n, 2)) * ст(a, div(n, 2))
        иначе
          знач := ст(a, n - 1) * a
      все
  все
кон
```

где `mod` – функция школьного алгоритмического языка, возвращающая остаток от деления своего первого аргумента на второй, `div` – функция, возвращающая целую часть частного от такого деления (в других языках программирования для расчёта указанных значений используются не функции, а специальные операции).

Один из самых эффективных методов возведения в степень – так называемый «бинарный алгоритм». Суть этого метода заключается в том, что степень n, в которую необходимо возвести число, представляется в двоичном (отсюда и название метода) виде:

$n = (m_k m_{k-1} \ldots m_1 m_0)_2$

где m_i – двоичные цифры.

Представим значение n в так называемой «развернутой записи»:

$n = m_{k-1} \times 2^{k-1} + m_{k-2} \times 2^{k-2} + \ldots + m_1 \times 2^1 + m_0$

Вспомнив свойство степеней ($a^{m+n} = a^m \times a^n$), получим значение a^n:

$$a^n = a^{m_{k-1} \times 2^{k-1}} \times a^{m_{k-2} \times 2^{k-2}} \times \ldots \times a^{m_1 \times 2} \times a^{m_0} \qquad (*)$$

Обратим внимание на то, что в полученной зависимости искомое значение степени определяется, кроме числа a, цифрами двоичного представления показателя степени n и степенями двойки (весомостью двоичных разрядов).

Существуют два варианта бинарного алгоритма: по схеме «слева направо» и по схеме «справа налево». Во всех имеющихся источниках такие названия связывают с порядком обработки битов двоичного представления показателя n, хотя, строго говоря, это не совсем

так, поскольку, кроме значения битов, в расчётах участвуют и степени двойки. По сути соответствующие варианты алгоритмов отличаются порядком расчёта множителей в зависимости (*). Опишем второй вариант как более простой.

В зависимости (*) имеются операции возведения в степень. Так как количество цифр в двоичном числе может быть очень большим, то желательно отказаться от возведения в большую степень – ведь именно эту задачу мы пытаемся оптимизировать. Можно рассуждать следующим образом.

Прежде всего напомним, что цифры m_i могут иметь значения 0 или 1. Это значит, что в зависимости (*) значения показателей степеней во множителях могут быть равны 1 или степени двойки, а сами множители, соответственно, – a или a в степени двойки.

Можно при обработке не вычислять каждый раз значение этой степени, а учитывать ранее рассчитанное значение, умножив его само на себя (например, $a^8 = a^4 \times a^4$ и т.п.). Меняющееся значение этой степени обозначим *степень_а*. Значение этой переменной будем рассчитывать для всех множителей зависимости (*), а учитывать его – только для цифр m_i, равных 1. При этом двоичные цифры отдельно выделять и записывать в массив не будем, а будем определять их (с сразу обрабатывать) как остатки от деления на 2 десятичного значения n и всех целочисленных частных от такого деления (то есть используем так называемый «перевод в двоичную систему методом последовательного деления на основание 2»).

Если переменную, последовательно накапливающую в себе произведение множителей в выражении (*), обозначить x, то соответствующий фрагмент программы определения значения a^n будет таким:

```
| Начальное значение переменной x
| зависит от последней цифры двоичной записи числа n
если mod(n, 2) = 1
  то
    x := a     | степень_а = a
  иначе
    x := 1     | степень_а = 1
все
| Определяем целую часть частного от деления на 2
n := div(n, 2)
| Рассматриваем остальные двоичные цифры (справа налево)
нц пока  n > 0
  | Определяем соответствующее значение степень_а
  степень_а := степень_а * степень_а
  если цифры[i] = 1          | Только в этом случае
    то
      x := x * степень_а     | учитываем его в произведении
    | иначе степень_а равна 1 (можно не умножать)
  все
  | Определяем целую часть частного от деления на 2
  n := div(n, 2)
кц
| Выводим результат
вывод x
```

Приведённый фрагмент можно сократить, не проверяя отдельно последнюю цифру двоичного представления числа n:

```
| Начальные значения
x := 1
степень_а := a  | Для последнего разряда
нц пока n > 0
  если mod(n, 2) = 1
    то
      x := x * степень_а
```

```
  все
  | Определяем значение степень_а для следующего (левого) разряда
  степень_а := степень_а * степень_а
  n := div(n, 2)
кц
…
```

Видно, что количество операций возведения в квадрат равно количеству цифр в двоичной записи числа *n*, а операций умножения – количеству единиц в ней.

Для варианта алгоритма по схеме «слева направо» все оценки – те же. Его программная реализация осложняется тем, что затруднено определение весомости начальных двоичных разрядов числа *n*. В решении проблемы помогает так называемая «схема Горнера» (предлагаем читателям разработать соответствующий вариант программы самостоятельно).

Итак, мы сравнили три метода возведения в степень с точки зрений количества выполняемых при расчётах умножений. Но так как при вычислениях на время получения результата влияют и другие факторы (выделение двоичных цифр показателя степени *n* и их проверка, время на выделение памяти для функций, вызываемых рекурсивно, и др.), то желательно сравнить перечисленные методы с точки зрения времени получения результата. Такое сравнение будет являться **целью** обсуждаемого проекта.

Указанное сравнение следует провести для различных значений *a* и *n*.

Чтобы при расчётах многократно не вводить в программы значения *a* и *n*, создадим и используем функции[3] для определения результата возведения в степень с двумя указанными аргументами. При использовании рекурсии такая функция уже была создана. Для двух других методов она будет такой:

- при использовании многократного умножения:

```
алг вещ ст (арг вещ a, цел n)
нач цел i, вещ произв
  произв := a
  нц для i от 1 до n - 1
    произв:= произв * a
  кц
  знач := произв
кон
```

- при использовании бинарного алгоритма по схеме «справа налево»:

```
алг вещ ст(арг вещ a, арг рез цел n)
нач вещ x, степень_а, i
  x := 1
  степень_а := a
  нц пока n > 0
    если mod(n, 2)  = 1
      то
        x := x * степень_а
    все
    степень_а := степень_а * степень_а
    n := div(n, 2)
  кц
  знач := x
кон
```

Обратим внимание на то, что аргумент *n* функции описан как параметр-переменная (**арг рез**), поскольку в ходе выполнения функции значение *n* меняется. Эта особенность

[3] С особенностями оформления функций в языке программирования, которым вы владеете, ознакомьтесь самостоятельно.

должна быть учтена и в программе на языке Паскаль (в программе на языке Python аргументы функции изменять можно).

После проверки правильности работы указанных функций можно перейти к расчётам времени.

Так как во время расчётов степени процессор компьютера будет выполнять и другие задачи, то для одних и тех же значений *a* и *n* в каждом из трёх случаев следует провести ряд замеров времени выполнения, которые потом – усреднить для каждого случая.

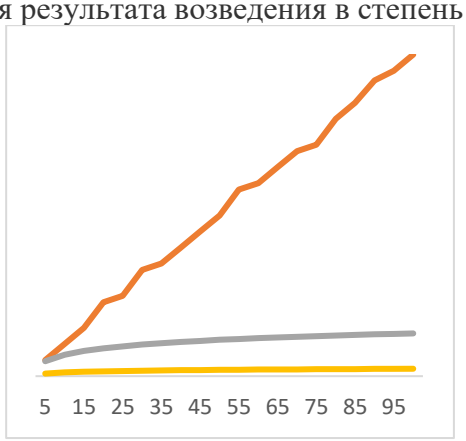
Время выполнения можно определить, зафиксировав текущее время, отслеживаемое в компьютере, до и после расчётов, а затем провести вычитание. В школьном алгоритмическом языке для этого можно использовать стандартную функцию `время`, которая возвращает текущее время в виде количества миллисекунд, прошедших с начала суток по местному времени. В языке Python соответствующая функция – `time` (модуль `time`). В языке Паскаль имеется процедура `GetTime`, которая возвращает четыре значения: количество полных часов, прошедших с начала суток, количество полных минут, прошедших с начала неполного часа, количество полных секунд, прошедших с начала неполной минуты, и количество миллисекунд, прошедших с начала неполной секунды (`GetTime(Hour, Min, Sec, MSec)`). Как, имея четыре перечисленные переменные, определить количество миллисекунд, прошедших с начала суток, установите самостоятельно.

С учётом сказанного, например, программа, определяющая среднее время определения значения степени для некоторых значений *a* и *n* при использовании многократного умножения, имеет вид:

```
алг
нач цел в1, в2, i, вещ результат
  в1 := время  | Фиксируем время начала расчётов
  нц для i от 1 до 100 | 100 повторений
    результат := ст(…, …)
  кц
  в2 := время  | Фиксируем время окончания расчётов
  | Выводим среднюю продолжительность расчётов
  вывод (в2 - в1)/100
кон
```

Обратите внимание, что значения результатов расчёта с помощью функции `ст` на экран не выводятся.

Проведя расчёты тремя рассмотренными методами с различными значениями *a* и *n*, с помощью программы Microsoft Excel или аналогичной программы получите графическую зависимость времени определения результата возведения в степень от значения *n*.



Примечание. Вид графиков – условный

Определите также:

- какое максимальное значение степени можно получить на вашем компьютере и сколько времени при этом проводились расчеты;

- до каких примерных значений *n* метод многократного умножения соизмерим[4] по времени расчёта с другими методами;

- до каких примерных значений *n* метод расчёта степени с помощью рекурсивной функции соизмерим по времени расчёта на основе бинарного алгоритма;

- влияет ли на время возведения в степень значение переменной *a*; если влияет, то как;

- зависит ли время возведения в степень от типа переменной *a* – целого и вещественного.

Самостоятельно изучите понятие «сложность алгоритма» и установите сложность алгоритмов, использованных в трёх рассмотренных методах.

Результаты работы присылайте в редакцию. Авторы лучших работ будут награждены дипломами. Срок представления – не ограничен.

Японский уголок

Два судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

	9		6			4		7
4		1	9		8	5		
	8				5	9		
	7			4				
9	6			5			1	2
				6			3	
		6	3				7	
		2	7		6	1		8
7		9			2		4	

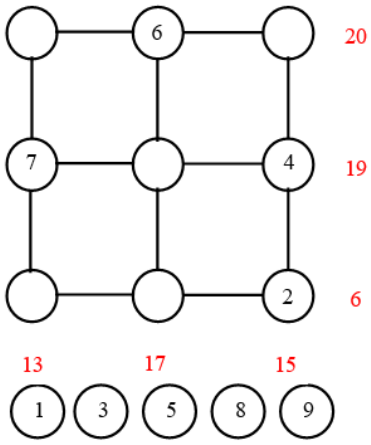
2) сложную:

		2		4					
4		6	8	1		9			
		5					4		
				9		3	5		
6						4			
				2			3		
		8	6	5				3	7
				3					6
7		3						9	4

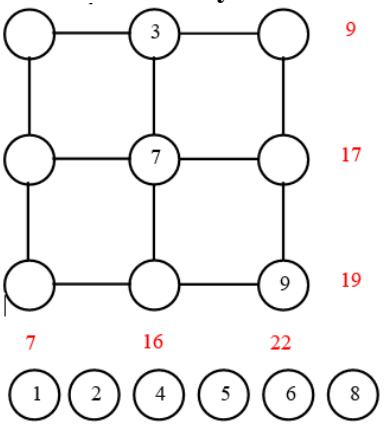
Сан-го-ку

Предлагаем читателям решить три японских головоломки другого типа — “сан-го-ку”. Их правила просты: необходимо расставить цифры в свободных кружочках на пустые места так, чтобы сумма цифр каждого ряда равнялась числу справа, а сумма цифр каждого столбца — числу снизу.

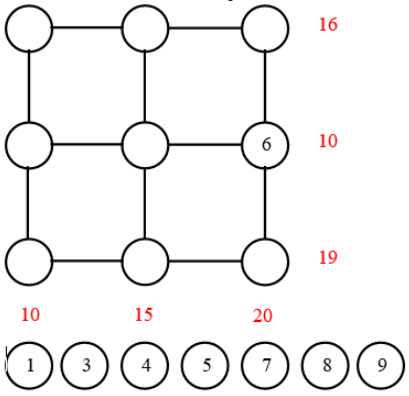
Сан-го-ку № 1



Сан-го-ку № 2



Сан-го-ку № 3



Решения (можно не всех головоломок) присылайте в редакцию.

---

[4] То есть когда время выполнения отличается не более чем на 8-10%.

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

Джордж Пойя

Лысый, Кудрявый и Усатый

В преступлении обвинялись три члена преступной группировки: Лысый, Кудрявый и Усатый. На следствии каждый из них сделал два заявления:

Лысый: “Я не делал этого. Это сделал Усатый”.

Кудрявый: “Усатый не виновен. Это сделал Лысый”.

Усатый: “Я этого не делал. Кудрявый этого тоже не делал”.

Суд установил, что один из них дважды солгал, другой дважды сказал правду, а третий — один раз солгал, а другой раз сказал правду. Кто совершил преступление

Лесная школа танцев

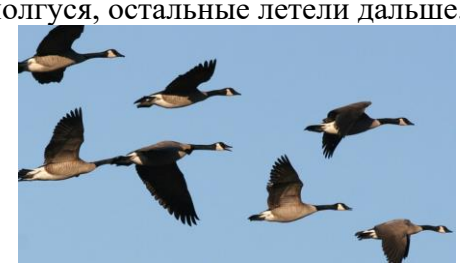
В лесную школу танцев пришли слон, волк и лев. Партнершами у них были выбраны мышка, белочка и лисичка. Учитель-жираф должен расставить их в пары. Сколько вариантов составления есть у него, если белочка боится, что ее съест волк, а слон — что он раздавит мышку?

Литература

1. Богомолова О.Б. Логические задачи. М. БИНОМ. Лаборатория знаний, 2005.

Сказочные гуси

В одной сказке над озерами летели гуси. На каждом озере садилась половина гусей и еще полгуся, остальные летели дальше. Все гуси сели на 7 озерах. Сколько было гусей?



Три студента

Три студента — Андреев, Борисов и Воронов — учатся на различных факультетах педагогического института (историческом, физико-математическом и иностранных языков). Все они приехали из различных городов: Калязина, Твери и Вышнего Волочка, причем один из них увлекается футболом, другой — баскетболом, третий — волейболом. Известно, что:

1) Андреев не из Вышнего Волочка, а Борисов не из Твери;

2) студент, приехавший из Вышнего Волочка, учится не историческом факультете;

3) тверянин учится на факультете иностранных языков и увлекается футболом;

4) Воронов учится на историческом факультете;

5) студент физико-математического факультета не любит волейбол.

Из какого города приехал каждый студент, на каком факультете он учится и каким видом спорта увлекается?

Это полезно знать

Язык программирования Python ушёл в отрыв

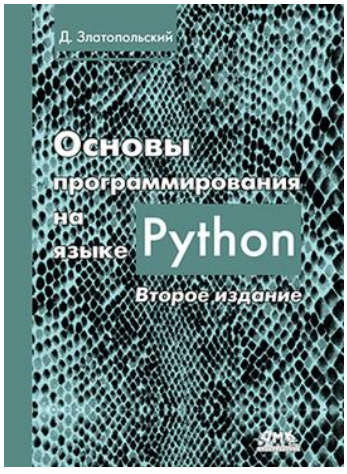
Индекс сообщества программистов TIOBE является индикатором популярности языков программирования. Индекс обновляется раз в месяц. По состоянию на сентябрь 2024 года доля языков программирования следующая (показаны 10 самых популярных языков, в последнем столбце – изменение за месяц).

Sep 2024	Sep 2023	Change	Programming Language	Ratings	Change
1	1		Python	20.17%	+6.01%
2	3	↑	C++	10.75%	+0.09%
3	4	↑	Java	9.45%	-0.04%
4	2	↓	C	8.89%	-2.38%
5	5		C#	6.08%	-1.22%
6	6		JavaScript	3.92%	+0.62%
7	7		Visual Basic	2.70%	+0.48%
8	12	⇈	Go	2.35%	+1.16%
9	10	↑	SQL	1.94%	+0.50%
10	11	↑	Fortran	1.78%	+0.49%

Книжная полка

Основы программирования на языке Python. Второе издание

Издательство ДМК Пресс предлагает книгу – учебник по языку программирования Python (https://dmkpress.com/catalog/computer/programming/python/978-5-97060-641-4/).

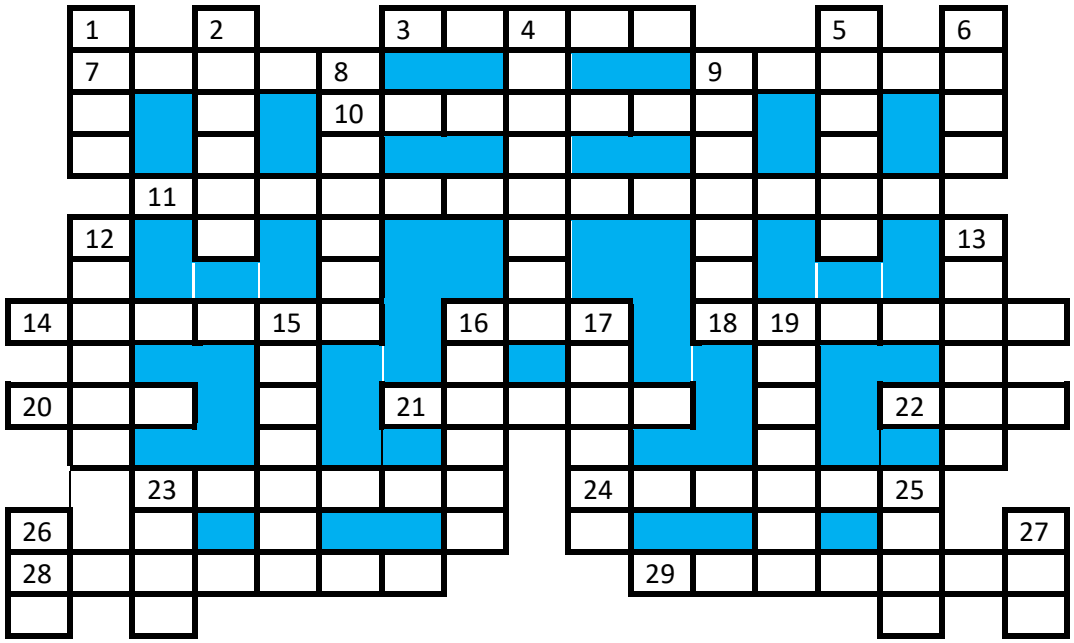


Если вы хотите научиться программировать на языке Python, который в последнее время становится популярным у нас в стране и за рубежом, то эта книга — для вас. В ней рассматриваются особенности разработки компьютерных программ и соответствующие инструкции языка Python, основные структуры данных этого языка (строки, списки, словари, файлы), типовые задачи программирования и методы их решения, а также вопросы совершенствования программы на основе использования функций. Впервые системно и популярно изложена методика разработки программ на языке Python с графическим пользовательским интерфейсом.

Книга предназначена для всех, кто изучает или преподает программирование.
Через сайт издательства книгу можно приобрести гораздо дешевле, чем в магазинах.

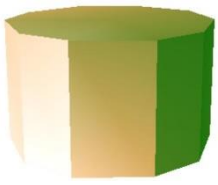
Кроссворд

Решите, пожалуйста, кроссворд.



По горизонтали

3. Системная (материнская) … .
7. Программа, обладающая способностью к самовоспроизведению.
9. Разработчик программы.
10. Электронная схема для запоминания одного бита информации.
11. Построение “заменителей” реально существующих объектов (предметов, явлений, процессов) с целью их исследования.
14. Наука о законах и формах мышления.
16. Средства массовой информации СМИ.
18. Так называют информацию, которая передается по Интернету.
20. Структура данных — двусторонняя очередь.
21. Один из контактов транзистора типа МОП (Металл-Оксид-Полупроводник).
22. Единица измерения скорости передачи данных.
23.



24. Шрифт наклонного начертания.
28. Разновидность, модификация, версия.
29. Часть представления числа с плавающей запятой, а также правильное, налаженное состояние, расположение чего-нибудь.

По вертикали

1. Так называют клавишу **Enter**.
2. Язык логического программирования.
4. Совокупность четко определенных правил для решения задачи за определенное число шагов.
5. Операция, производимая с файлом при работе компьютера.

6. Конечное число точек на плоскости, соединенных отрезками линий.

8. →.

9.

12. Название клавиши.

13. Перечень предметов, фамилий или т.п., а в программировании — совокупность элементов, каждый из которых содержит указатель на следующий элемент.

15. Элемент стандартного устройства для ввода информации в компьютер.

16. Структура, содержание.

17. Деталь печатающего элемента матричного принтера.

19. Место хранения информации в процессоре.

23. Точка подключения внешних устройств к внутренней шине микропроцессора.

25. H_2O.

26. Основание системы счисления, используемой в компьютере.

27. Буква греческого алфавита, которой, как правило, обозначают неизвестную величину.

Робот и книга

Представьте, что имеется робот, способный шагать, поднимать и опускать руки, сжимать и разжимать пальцы и т.п. по соответствующим командам.

Ему была дана для выполнения программа для переноса книги на стол, состоящая из следующих команд:

1. Вытянуть правую руку.
2. Разжать пальцы правой руки.
3. Сжать пальцы правой руки (предполагается, что между командами 2 и 3 мы положили книгу в руку робота).
4. Опустить правую руку.
5. Повернуться на 180°.
6. Поставить правую ногу впереди левой.
7. Поставить левую ногу впереди правой.
8. Поставить правую логу впереди левой.
9. Поставить левую ноту впереди правой.
10. Поставить правую ногу впереди левой.
11. Поставить левую ногу впереди правой.
12. Вытянуть правую руку.
13. Разжать пальцы правой руки.

Что произойдет в результате выполнения роботом такой программы?

Какова система команд робота?

Как можно уменьшить размер программы?

Три кучки спичек

Положите на стол 3 кучки спичек. В одну кучку положите 11 спичек, во вторую — 7, в третью — 6. Перекладывая спички из одной кучки в другую, нужно сделать так, чтобы в каждой кучке было бы по 8 спичек (это возможно, так как общее число спичек — 24 — делится на 3 без остатка). При этом требуется соблюдать следующее правило: к любой кучке

разрешается дополнять ровно столько спичек, сколько в ней есть. Например, если в кучке 6 спичек, то и добавлять к ней можно только 6, если в кучке 4 спички, то и добавлять к ней можно только 4.

Решение, пожалуйста, оформите в виде:

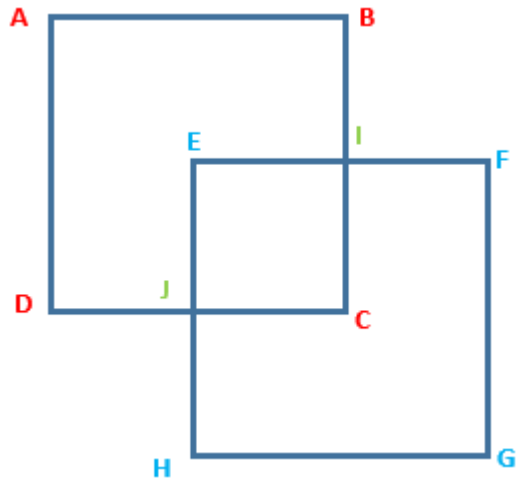
Кучка	Начальное состояние	Первый ход	Второй ход	…
Первая	11			
Вторая	7			
Третья	6			

Два квадрата

Можно ли нарисовать фигуру, изображенную ниже, не отрывая карандаша от бумаги и не проводя дважды по одному и тому же отрезку? Если можно, то как?

Алгоритм решения задачи, пожалуйста, оформите в виде:

1. AB.
2. BI.
3. …



Головоломка “Цифры в квадратиках”

Впишите в приведённые ниже квадратики цифры от 1 до 9 так, чтобы все эти девять цифр были использованы и выполнялись указанные равенства.

□ □ : □ = □ – □ = □ + □ = □ × □

Восстановите набор чисел

Ниже представлены три набора чисел. Используя первые два, восстановите третий набор. Какое число скрывается под знаком вопроса?

4: 6, 11, 7;

3: 8, 4, 12;

?: 10, 55, 15.

Конкурс № 67 «Отмеривание воды». Тур 2

В качестве заданий этого конкурса, который будет проходить *в несколько туров*, а его итоги будут подводиться *с учётом всех туров в целом*, будут предложены задания следующего типа.

Представьте себе, что имеется исполнитель, которого назовем Водомером, так как он занимается отмериванием того или иного количества воды. У него есть источник воды (река, озеро или т. п.), количество воды в котором неограниченно, и две или три ёмкости (банки, ведра или т. п.): A и B (или A, B и C). При двух ёмкостях система команд, которые может выполнять Водомер, такая:

К1. Наполнить A
К2. Наполнить B
К3. Перелить из A в
К4. Перелить из B A
К5. Вылить из A
К6. Вылить из B

при трёх:

К1. Наполнить A
К2. Наполнить B
К3. Наполнить C
К4. Перелить из A в B
К5. Перелить из A в C
К6. Перелить из B в A
К7. Перелить из B в C
К8. Перелить из C в A
К9. Перелить из C в B
К10. Вылить из A
К11. Вылить из B
К12. Вылить из B

Составьте алгоритмы решения следующих задач, которые должен решить Водомер. Если иное не оговорено особо, задача должна решаться за минимально возможное количество действий[5].

Алгоритм решения задач, пожалуйста, оформите в виде таблицы (приведены условные значения):

№ пп	Обозначение команды	В чём она заключается	Станет объём воды в ёмкости	
			A	B
Сначала			0	0
1	К1	Наполнить A	5	0
2				
3				
4				

[5] Обращаем внимание на то, что среди предложенных задач имеются задачи с одинаковыми исходными данными и требуемым результатом, но отличающиеся условиями получения результата (минимальное число операций или минимально возможное количество используемой воды — см. комментарий к задаче 9).

Тур 2

11. Объем емкости A равен 3 л, емкости B — 10 л, емкости C — 11 л, надо получить 9 л.
12. Объем емкости A равен 3 л, емкости B — 7 л, емкости C — 12 л, надо получить 11 л.
13. Объем емкости A равен 4 л, емкости B — 5 л, емкости C — 8 л, надо получить 6 л.
14. Объем емкости A равен 4 л, емкости B — 5 л, емкости C — 8 л, надо получить 2 л, использовав при этом минимально возможное количество воды.
15. Объем емкости A равен 4 л, емкости B — 7 л, емкости C — 11 л, надо получить 10 л.
16. Объем емкости A равен 4 л, емкости B — 7 л, емкости C — 11 л, надо получить 6 л.
17. Объем емкости A равен 4 л, емкости B — 8 л, емкости C — 9 л, надо получить 3 л.
18. Объем емкости A равен 4 л, емкости B — 8 л, емкости C — 9 л, надо получить 7 л.
19. Объем емкости A равен 3 л, емкости B — 5 л, емкости C — 11 л, надо отмерить 9 л.
20. Объем емкости A равен 3 л, емкости B — 6 л, емкости C — 10 л, надо получить 2 л в емкости A.

Ответы присылайте по адресу: mirinfo@infojournal.ru. Укажите в письме свои фамилию и имя, населенный пункт, номер школы и фамилию, имя и отчество учителя информатики. Срок представления ответов — 30 сентября 2024 г.

Поиск информации

Про птиц и собаку

Ответы на приведённые ниже вопросы найдите в Интернете или по другим источникам информации.

1. Некий Нильс Олаф вот уже около 30 лет служит в гвардии одной из скандинавских стран и с завидной регулярностью получает очередное воинское звание. В этом нет ничего странного, если бы ни одно “но”: этот Нильс Олаф — птица. Какая именно?

2. Когда-то прежде у англичан эту птицу считали ой, кто “взбивает ветер”. Оно и понятно, она находится в прямом родстве с соколами и стремительность в полете для нее норма. Что это за птица?

3. В одной из азиатских столиц запрещено иметь в доме более одной собаки. Кроме того, закон строго ограничивает рост лучшего друга семьи. Что это за город? Какое ограничение по росту устанавливает закон?

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

Уважаемые читатели!

Сообщаем, что очередные номера нашего интернет-журнала публикуются в период с 15 по 20 число каждого месяца, кроме июля.