Задача 1. Распечатать время

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая распечатывает в выходной файл время, и решить с её помощью тестовую задачу.

Функция должна иметь сигнатуру:

```
void printTime(int h, int m, int s);
```

Здесь `h` — количество часов (от 0 до 23), а `m` и h — количество минут и секунд соответственно (от 0 до 59).

Формат входных данных

В первой строке содержится целое число N — количество времён в файле ($2 \leqslant N \leqslant 1\,000$). В каждой из следующих N строк описано время в формате трёх целых чисел: `h`, `m`, `s`.

Формат выходных данных

Нужно вывести N строк, по одной дате в строке, в формате `HH:MM:SS` (см. пример).

Пример

input.txt	output.txt
3 12 37 59 1 1 1 23 59 59	12:37:59 01:01:01 23:59:59

Комментарий

Вам может пригодиться формат `"%02d"`: он печатает целое число, дополняя его слева нулями до двух символов.

Задача 2. Развернуть строку

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая будет разворачивать заданную строку.

Функция должна иметь сигнатуру:

```
void reverse(char *start, int len);
```

Здесь `start` — указатель на начало строки (т.е. на первый её символ), а `len` — количество символов в ней (исключая завершающий нулевой символ).

Формат входных данных

В первой строке содержится целое число N — количество строк в файле ($2 \leqslant N \leqslant 1\,000$). В каждой из следующих N строк записана одна строка, которую нужно развернуть. Строка состоит только из символов латинского алфавита, и её длина лежит в диапазоне от 1 до 100 включительно.

Формат выходных данных

Нужно вывести N развёрнутых строк.

Пример

input.txt	output.txt
4 C is cool language	C si looc egaugnal

Задача 3. Склеить строки

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда*
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая будет конкатенировать заданные строки. Если конкретнее, она должна дописывать вторую строку в конец первой строки.

Функция должна иметь сигнатуру:

```
char* concat(char *pref, char *suff);
```

Здесь `pref` — указатель на начало первой строки, а `suff` — указатель на начало второй строки. Нужно дописать содержимое строки `suff` в конец строки `pref`. Предполагается, что вызывающий гарантирует, что в буфере `pref` хватит места на дописываемую строку.

Обе входных строки заканчиваются нулевым символом, и склеенная строка также должна заканчиваться на него. В качестве возвращаемого значения нужно вернуть указатель на конец (т.е. на нулевой символ) склеенной строки.

В качестве тестовой задачи нужно объединить все заданные строки.

Формат входных данных

В первой строке содержится целое число N — количество строк в файле ($2 \leqslant N \leqslant 10\,000$). В каждой из следующих N строк записана одна строка. Строка состоит только из символов латинского алфавита, и её длина лежит в диапазоне от 1 до 100 включительно.

Формат выходных данных

Нужно вывести одну строку, в которой объединены по порядку все N строк из входного файла.

Пример

input.txt	output.txt
4 C is cool language	Ciscoollanguage

Задача 4. Сколько буков

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, которая будет определять, сколько в строке больших букв, маленьких букв и цифр.

Функция должна иметь сигнатуру:

```
int calcLetters(char *iStr, int *oLowerCnt, int *oUpperCnt, int *oDigitsCnt);
```

Здесь `iStr` — указатель на начало строки, завершающейся нулевым символом. Параметры `oLowerCnt`, `oUpperCnt` и `oDigitsCnt` выходные: вызывающий передаёт в них указатель на какие-нибудь локальные переменные, чтобы получить в них соответствующий результат. Функция возвращает длину строки `iStr`, в переменную `*oLowerCnt` нужно записать количество маленьких букв, в `*oUpperCnt` записать количество больших букв, а в `*oDigitsCnt` записать количество цифр.

В качестве тестовой задачи нужно прочитать все строки файла и распечатать статистику для каждой из них.

Формат входных данных

Строки файла могут содержать любые печатаемые символы ASCII, включая пробелы (коды от 32 до 126 включительно). Поэтому рекомендуется использовать `gets` для чтения строк.

Длина любой строки не превышает 100, строки могут быть пустыми. Учтите, что последняя строка файла также завершается символом перевода строки.

Формат выходных данных

Для каждой строки входного файла выведите статистику ровно в том же формате, как в примере выходных данных.

Пример

input.txt

```
/*C-style comments can contain
multiple lines*/ /*or just one*/
// C++-style comment lines
int main() {
  // The below code wont be run
  //return 1;
  return 0; //this will be run
}
```

output.txt

```
Line 1 has 30 chars: 24 are letters (23 lower, 1 upper), 0 are digits.
Line 2 has 32 chars: 22 are letters (22 lower, 0 upper), 0 are digits.
Line 3 has 26 chars: 18 are letters (17 lower, 1 upper), 0 are digits.
Line 4 has 12 chars: 7 are letters (7 lower, 0 upper), 0 are digits.
Line 5 has 31 chars: 21 are letters (20 lower, 1 upper), 0 are digits.
Line 6 has 13 chars: 6 are letters (6 lower, 0 upper), 1 are digits.
Line 7 has 30 chars: 19 are letters (19 lower, 0 upper), 1 are digits.
Line 8 has 1 chars: 0 are letters (0 lower, 0 upper), 0 are digits.
```

Задача 5. Разделить на слова

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию (или функции) для выделения слов из заданной строки. Слова отделены друг от друга символами-разделителями, которые передаются в строку как параметр.

Сигнатуру функций и правила их вызова вам нужно придумать самостоятельно.

С помощью этих функций нужно решить тестовую задачу. Дана одна строка длиной до 10^6, состоящая из букв латинского алфавита и знаков препинания четырёх типов: точка, запятая, точка с запятой, двоеточие. Нужно найти слова в этой строке, состоящие из букв, и определить длину каждого слова и количество заглавных букв в нём. Для каждого слова нужно вывести эту информацию так, как показано в примере.

Пример

input.txt	output.txt

`..ko,.Privet:kreved,.;ko:;,.`	`0/2 ko`
	`1/6 Privet`
	`0/6 kreved`
	`0/2 ko`

Комментарий

Следует выводить слова ровно в том порядке, в котором они встречаются в строке.

Задача 6. Прочитать время

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется реализовать функцию, будет считывать время из заданной строки.
Функция должна иметь сигнатуру:

```
int readTime(char *iStr, int *oHours, int *oMinutes, int *oSeconds);
```

Здесь `iStr` — указатель на строку, в которой должно быть записано время. Параметры `oHours`, `oMinutes`, `oSeconds` — выходные параметры, т.е. вызывающий должен передать в них указатель на свои локальные переменные, куда будет записаны соответсвующие результаты: `*oHours` — количество часов, `oMinutes` — количество минут, `oSeconds` — количество секунд. Функция должна возвращать код возврата: `1`, если прочитать время удалось, и `0` в случае неудачи.

Параметры `oMinutes`, `oSeconds` опциональные: вызывающий может передать в них NULL, если его, например, не интересует количество секунд. При этом если указатель `oMinutes` нулевой, то и указатель `oSeconds` тоже должен быть нулевой.

Время записано в формате *H:M:S* или *H:M*. То есть строка состоит из двух или трёх частей, отделённых друг от друга одним двоеточием. Каждая часть — это целое число из одной или двух цифр, возможно с ведущими нулями. При этом если задано две части, то это часы и минуты, а если три — то это часы, минуты и секунды. Заметим, что часы должны быть в диапазоне до 0 до 23, а минуты и секунды в диапазоне от 0 до 59.

Используя функцию, нужно решить тестовую задачу. В файле записана тестовая строка длины от 3 до 15 символов без пробелов. Нужно применить функцию `readTime` к каждой строке и распечатать результат её вызова.

Нужно вызвать функцию три раза:

1. Указатели `oHours`, `oMinutes`, `oSeconds` ненулевые. После вызова нужно распечатать код возврата и числа `*oHours`, `*oMinutes`, `*oSeconds` через пробел.
2. Указатель `oSeconds` нулевой. После вызова нужно распечатать код возврата и числа `*oHours`, `*oMinutes` через пробел.
3. Указатели `oMinutes` и `oSeconds` нулевые. После вызова нужно распечатать код возврата и число `*oHours` через пробел.

Заметим, что вызывать функцию три раза подряд по сути бессмысленно: мы делаем это только для того, чтобы протестировать все случаи с нулевыми указателями.

Формат входных данных

В единственной строке задано время в формате, указанном выше (с секундами или без).

Время на входе также может быть указано неверно. Чтобы не было разногласий, в каком случае считать время корректным, а в каком нет, гарантируется, что во всех тестах будут только ошибки двух видов:

1. Формат полностью соответствует условию, но число часов, минут или секунд выходит за пределы допустимого диапазона.
2. Формат некорректный: взято корректно записанное время, и между каждой парой символов в строке вставлен символ вертикальной черты ‘|’ (ASCII 124).

Таким образом, вы можете самостоятельно решить, считать ли случаи вроде `12:001:35` или `17:0xy` корректными или нет: в тестах таких ситуаций не будет.

Формат выходных данных

Нужно вывести три строки с 4-мя, 3-мя и 2-мя целыми числами соответственно (см. описание выше в условии).

Заметьте, что если дата задана неверно, то все числа кроме кода возврата должны быть равны -1.

Пример

input.txt	output.txt
15:01:13	1 15 1 13 1 15 1 1 15
7:9	1 7 9 0 1 7 9 1 7
7:99	0 -1 -1 -1 0 -1 -1 0 -1
1|3|:|5|:|1|0	0 -1 -1 -1 0 -1 -1 0 -1

Комментарий

Вам может пригодиться функция `sscanf` и её возвращаемое значение.

Пояснение к примеру

В первом примере указаны секунды, а во втором — нет. В третьем тесте 99 минут, что выходит за пределы диапазона, поэтому возвращается неуспех. Четвертый тест неверно отформатирован: это время `13:5:10` со вставленными вертикальными чертами.

Задача 7. Гистограмма

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Во входном файле содержится некоторый текст. Вам необходимо построить гистограмму встречаемости различных символов в тексте.

Формат входных данных

Входной файл содержит просто текст. Текст состоит только из ASCII-символов с кодами от 0 до 126.

Размер текста не превосходит 100 000 байт.

Формат выходных данных

Для каждого печатаемого символа (ASCII код от 32 до 126 включительно), встретившегося в тексте хотя бы раз, выведите сам символ и через пробел выведите столько символов '#', сколько раз данный символ встретился в тексте. Символы выводить в порядке увеличения их кода.

Пример

input.txt	output.txt
This is a text. Multiline text.	␣ #### . ## M # T # a # e ### h # i #### l ## n # s ## t ##### u # x ##

Комментарий

Первый символ в примере вывода — пробел.

Для чтения данных можно использовать посимвольных ввод с помощью `getchar` или построчный с помощью `gets`.

Задача 8. Таблица

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Есть набор видео, у каждого видео есть свой идентификатор (ID) — целое число в диапазоне от 0 до 1 000. Некоторые из этих видео могут быть разбиты на фрагменты, остальные видео заданы одним фрагментом.

Во входном файле заданы все фрагменты, для каждого из них указан идентификатор видео и длительность фрагмента в секундах. Нужно вывести статистику по каждому видео в виде тройки: ID, количество фрагментов, суммарная длительность. Все тройки нужно записать в красиво отформатированную таблицу.

Формат таблицы виден в примере выходного файла. В каждой ячейке таблицы число выровнено по правому краю ячейки. Слева и справа от ячейки стоят специальные пробелы. Ширину всех ячеек в каждом столбце таблицы нужно выбрать минимально возможной с учётом этих условий.

Формат входных данных

В первой строке задано число N — суммарное количество фрагментов ($1 \leqslant N \leqslant 10^4$).

В каждой из оставшихся N строк указано по два целых числа: ID видео, в которое входит фрагмент, и длительность фрагмента. Все длительности целые, неотрицательные, не превышают 10^5.

Формат выходных данных

Выведите информацию о каждом видео в строку таблицу. Первый столбец — это ID видео, второй — количество его фрагментов, а третий — суммарная длительность. Видео должны быть перечислены в таблице в порядке увеличения ID.

Пример

input.txt	output.txt
5 37 5 1 17 313 2378 1 79 1 35	+-----+---+------+ | 1 | 3 | 131 | +-----+---+------+ | 37 | 1 | 5 | +-----+---+------+ | 313 | 1 | 2378 | +-----+---+------+

Пояснение к примеру

Здесь задано три видео, и только одно из них (ID = 1) разбито на фрагменты. У него три фрагмента, и если просуммировать их длительность, то получается 131 секунда.

Задача 9. Реальные логи

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В данной задаче во входной файл подаётся “как есть” лог проигрывания видео-файлов из настоящей игры TheDarkMod. Нужно прочитать логи и вывести некоторую простую информацию о каждом видео.

В логах записано множество сообщений, поступающих из кода, который в реальном времени декодирует видео и выдаёт наружу готовые для отрисовки кадры. Каждое сообщение начинается с указания момента времени, когда оно произошло (timestamp), который записан в формате `A.BBB.CCC`, где A, B и C — количество секунд, миллисекунд и микросекунд соответственно.

Когда какой-либо видео-файл начинает проигрываться, в логи пишется сообщение “`Decoded first frame`”, а когда заканчивает проигрываться, тогда пишется сообщение “`Video ended: no more frames`”. Время, прошедшее от начала проигрывания до конца будем считать реальным временем показа видео (его надо выводить в ответ). Известно, что в любой момент времени проигрывается не больше одного видео-файла одновременно.

В процессе проигрывания видео-файла программа декодирует кадры из него. Для декодирования каждого кадра сначала вызывается функция распаковки кадра из библиотеки FFmpeg (сообщение “`Packet decoded`”), а потом распакованный кадр переводится в цветовое пространство RGB (сообщение “`Converted to RGBA`”). У каждого из этих двух сообщений подписано, сколько времени заняла соответствующая процедура в миллисекундах, Сумму этих двух значений будем считать временем декодирования одного конкретного кадра.

Нужно собрать простую статистику по тому, как долго декодировались кадры каждого отдельного видео. Нужно посчитать:

- сколько всего кадров было декодировано,
- сколько в сумме времени затрачено на декодирование кадров,
- минимальное, максимальное и среднее время декодирования кадра.

В выходной файл нужно вывести эти результаты в формате, аналогичном показанному в примере. Если в логе проигрывалось несколько видео-файлов, то нужно указать статистику для каждого из них, перечисляя их в том порядке, в котором они проигрывались.

Формат входных данных

Внимание: входной файл для примера вы можете скачать отсюда. А вот в поле выходного файла указан собственно вывод, который должна получить ваша программа.

В данной задаче кроме примера есть ещё лишь два теста. Они очень похожи на пример (записаны таким же образом).

Комментарий

Все вещественные числа (т.е. время в миллисекундах) требуется выводить **ровно** с тремя знаками после десятичной точки, причём округлять нужно **в сторону ближайшего** числа с тремя знаками.

Пример

input.txt

output.txt

Video "video/ffmpeg_test/tearsofsteel_avi_mpeg4_mp3.avi":
  Total frames: 1007
  Total decode time: 257.397 ms
  Actual playback time: 41934.506 ms
  Min/avg/max decode time: 0.233/0.256/0.584 ms
 	
Video "video/ffmpeg_test/sykes_roq_roq.roq":
  Total frames: 354
  Total decode time: 171.307 ms
  Actual playback time: 11788.245 ms
  Min/avg/max decode time: 0.419/0.484/0.719 ms

Задача 10. Числительные

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Вася пишет программу, которая оценивает, сколько строк кода в проекте. Всё самое сложное уже написано, осталось лишь написать вывод ответа пользователю. Для красоты Вася хочет красиво распечатать результат словами, например: “две тысячи сто тридцать пять строк”. Вам нужно написать алгоритм построения этой строки.

Формат входных данных

В первой строке задано число T — количество тестовых случаев в файле ($1 \leqslant T \leqslant 10^4$). В каждой из остальных T строк указано одно целое число N — количество строк в проекте. Сегодня проекты стали довольно большими, поэтому диапазон: $1 \leqslant N < 10^9$.

Формат выходных данных

В каждую строку выходного файла нужно записать словами, сколько строк в соответствующем проекте.

Чтобы избежать вопросов кодировок, слова нужно выводить транслитом. Используется транслит BGN/PCGN, онлайн-транслитератор есть на этой странице. Мягкий знак заменяется на кавычку (ASCII 39).

(пример приведён на следующей странице)

Пример

input.txt

```
17
1
4
9
21
33
40
54
73
173
345
797
987
1000
1986
100000
5001002
32011171
```

output.txt

```
odna stroka
chetyre stroki
devyat’ strok
dvadtsat’ odna stroka
tridtsat’ tri stroki
sorok strok
pyat’desyat chetyre stroki
sem’desyat tri stroki
sto sem’desyat tri stroki
trista sorok pyat’ strok
sem’sot devyanosto sem’ strok
devyat’sot vosem’desyat sem’ strok
odna tysyacha strok
odna tysyacha devyat’sot vosem’desyat shest’ strok
sto tysyach strok
pyat’ millionov odna tysyacha dve stroki
tridtsat’ dva milliona odinnadtsat’ tysyach sto sem’desyat odna stroka
```