1 Последовательности

1. Написать функцию, получающую в качестве аргумента имя файла, содержащего последовательность вещественных чисел неизвестной длины, и возвращающую целое число, равное количеству максимальных элементов этой последовательности. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
2. Написать функцию, получающую в качестве аргумента имя файла, содержащего последовательность вещественных чисел неизвестной длины, и возвращающую целое число, равное количеству локальных максимумов этой последовательности. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
3. Написать функцию, получающую в качестве аргумента имя файла, содержащего последовательность вещественных чисел неизвестной длины, и возвращающую целое число, равное 1, если эта последовательность возрастает, 2, если убывает, 3, если постоянна, 4, если недостаточно данных для принятия решения, и 0, если не монотонна. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
4. Написать функцию, получающую в качестве аргумента имя файла, содержащего последовательность целых чисел неизвестной длины, и возвращающую целое число, равное 1, если эта последовательность является последовательностью чисел Фибоначчи, и 0 в противном случае (последовательность является последовательностью Фибоначчи, если первые два ее элемента равны 1, а каждый последующий равен сумме двух предыдущих). Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
5. Написать функцию, получающую в качестве аргумента имя файла, содержащего последовательность целых чисел неизвестной длины, и возвращающую целое число, равное 1, если эта последовательность является последовательностью чисел Фибоначчи, записанной в обратном порядке, и 0 в противном случае. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
6. Написать функцию, получающую в качестве аргументов имена двух файлов, содержащих последовательности $a_1, a_2, \ldots$ и $b_1, b_2, \ldots$ вещественных чисел неизвестной длины. Функция должна возвращать целое число, равное 1, если каждый элемент второй последовательности (кроме первого и последнего) равен полусумме элементов первой последовательности с соседними ему номерами: $b_i = (a_{i+1} + a_{i-1})/2$, и 0 в противном случае. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
7. Написать функцию, получающую в качестве аргументов имя файла, содержащего последовательность вещественных чисел неизвестной длины, и адрес вещественного числа d, и возвращающую в d вещественное число, равное среднему квадратическому отклонению чисел из этого файла от их среднего арифметического. Возвращаемое значение функции равно длине последовательности в случае успешного завершения и равно -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
8. Написать функцию, получающую в качестве аргумента имя файла, содержащего последовательность вещественных чисел неизвестной длины, и возвращающую целое число, равное 1, если эта последовательность является арифметической прогрессией, 2, если она является геометрической прогрессией, 3, если постоянна, 4, если в ней недостаточно элементов для принятия решения, и 0 в противном случае. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д.. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.

9. Написать функцию, получающую в качестве аргументов адреса вещественной переменной x и целых переменных i, j, и имя файла, содержащего последовательность $x_1, x_2, \ldots$ вещественных чисел неизвестной длины. В результате работы функции вещественная переменная x получает значение, равное максимальному элементу этой последовательности. Целая переменная i получает значение, равное номеру первого максимального элемента в последовательности, переменная j получает значение, равное номеру последнего максимального элемента в последовательности. Функция должна возвращать 0 в случае успеха и -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д., при этом переменные x, i, j не изменяются. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.
10. Написать функцию, получающую в качестве аргументов вещественное число x, адреса целых переменных i, j и имя файла, содержащего последовательность $x_1, x_2, \ldots$ вещественных чисел неизвестной длины. Функция возвращает целое число, равное 1, если x присутствует в этой последовательности, и 0 в противном случае. Если x есть в последовательности, то целая переменная i получает значение, равное номеру первого элемента последовательности, совпадающего с x, переменная j получает значение, равное номеру последнего элемента последовательности, совпадающего с x. Если числа x в последовательности нет, то переменные i и j не изменяются. Функция должна возвращать -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д., при этом переменные i, j не изменяются. Основная программа должна вызывать эту функцию и выводить на экран результат ее работы.

2 Массивы

1. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел и целое число, являющееся длиной этого массива, и заменяющую каждый элемент массива (кроме первого и последнего) на полусумму соседей. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.
2. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел, целое число n, являющееся длиной этого массива, и вещественное число x, и переставляющую элементы этого массива так, чтобы вначале шли все элементы массива, меньшие x, а затем – все элементы, большие x, произведя при этом не более $3n/2 + O(1)$ перемещений элементов. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.
3. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел и целое число n, являющееся длиной этого массива, и циклически сдвигающую элементы массива на одну позицию вправо, произведя не более $n + O(1)$ перемещений элементов. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.
4. Написать подпрограмму, получающую в качестве аргументов массив вещественных чисел, целое число n, являющееся длиной этого массива и целое число k, и циклически сдвигающую элементы массива на k позиций вправо произведя не более $n + O(1)$ перемещений элементов. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.
5. Написать функцию, получающую в качестве аргументов имя файла, массив $a[n]$ вещественных чисел и целое число n, и возвращающую целое число, равное количеству вхождений подпоследовательности $a[n]$ в последовательность, содержащуюся в заданном файле. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту функцию и выводить на экран результат ее работы.

6. Написать функцию, получающую в качестве аргументов массив $a[n]$ вещественных чисел, целое число n, и вещественное число x, и выбрасывающую из массива все элементы, меньшие x (при выбрасывании элемента $a[i]$ все элементы с номером, большим i, сдвигаются на одну позицию к началу массива и длина массива уменьшается на 1), произведя при этом не более $n+O(1)$ перемещений элементов. Функция возвращает длину получившегося в результате массива. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту функцию и выводить на экран результат ее работы.

3 Сортировки

1. Написать функцию, получающую в качестве аргументов неубывающий массив $a[n]$ вещественных чисел, целое число n, являющееся длиной этого массива и вещественное число x, и возвращающую место, где в этот массив можно вставить число x (т.е. целое число i такое, что $a[i] \le x \le a[i+1]$). Место определяется двоичным поиском по следующему алгоритму (*алгоритм деления пополам*).

   > Взять первоначально 0 и n в качестве границ поиска элемента; далее, до тех пор, пока границы не совпадут, шаг за шагом сдвигать эти границы следующим образом: сравнить x с $a[s]$, где s – целая часть среднего арифметического границ; если $a[s] < x$, то заменить прежнюю нижнюю границу на $s+1$, а верхнюю оставить без изменений, иначе оставить без изменения нижнюю границу, а верхнюю заменить на s; когда границы совпадут, став равными некоторому числу t, выполнение закончится с результатом t.

   Общее количество сравнений и перестановок элементов в наихудшем случае не должно превышать $\log_2 n + O(1)$. Основная программа должна заполнять данными массив (из файла), выводить его на экран, вводить с клавиатуры число x, вызывать эту функцию и выводить на экран результат ее работы.
2. Написать подпрограмму, получающую в качестве аргументов три массива $a[n]$, $b[m]$, $c[n+m]$ вещественных чисел, где $a[n]$, $b[m]$ не убывают, и целые числа n и m, и строящую по неубывающим массивам $a[n]$, $b[m]$ неубывающий массив $c[n+m]$ слиянием первых двух за $n+m+O(1)$ сравнений и $n+m+O(1)$ пересылок элементов по следующему алгоритму

   > Просматриваем очередные элементы $a[i]$, $i = 0,\ldots,n-1$ и $b[j]$, $j = 0,\ldots,m-1$ массивов a и b. Если $a[i] < b[j]$, то $c[k] = a[i]$ и увеличиваем i на 1, иначе $c[k] = b[j]$ и увеличиваем j на 1. Здесь k, $k = 0,\ldots,n+m-1$ – очередной элемент массива c (здесь всегда равен $i+j$).

   Основная программа должна заполнять данными массивы $a[n]$ и $b[m]$ (из файла), выводить их на экран, вызывать эту подпрограмму и выводить на экран результат ее работы – массив $c[n+m]$.
3. Написать функцию, получающую в качестве аргументов три массива $a[n]$, $b[m]$, $c[n+m]$ вещественных чисел, где $a[n]$, $b[m]$ возрастают, и целые числа n и m, и строящую по возрастающим массивам $a[n]$, $b[m]$ возрастающий массив $c[n+m]$ слиянием первых двух (с выбрасыванием повторяющихся чисел) за $n+m+O(1)$ сравнений и $n+m+O(1)$ пересылок элементов по следующему алгоритму

   > Просматриваем очередные элементы $a[i]$, $i = 0,\ldots,n-1$ и $b[j]$, $j = 0,\ldots,m-1$ массивов a и b. Если $a[i] < b[j]$, то $c[k] = a[i]$ и увеличиваем i на 1, если $a[i] > b[j]$, то $c[k] = b[j]$ и увеличиваем j на 1, иначе увеличиваем на 1 i или j. Здесь k, $0 \le k \le n+m-1$ – очередной элемент массива c.

   Функция должна возвращать длину получившегося в результате массива c. Основная программа должна заполнять данными массивы $a[n]$ и $b[m]$ (из файла), выводить их на экран, вызывать эту подпрограмму и выводить на экран результат ее работы - массив c.

4. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ вещественных чисел и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*"пузырьковая"сортировка*):

   Последовательным просмотром чисел $a[0], a[1], \ldots, a[n-1]$ найти наименьшее i такое, что $a[i] > a[i+1]$. Поменять $a[i]$ и $a[i+1]$ местами, возобновить просмотр с элемента $a[i+1]$ и т.д. Тем самым наибольшее число передвинется на последнее место. Следующие просмотры начинать опять сначала, уменьшая на 1 количество просматриваемых элементов. Массив будет упорядочен после просмотра, в котором участвовали только первый и второй элементы.

   Общее количество сравнений в наихудшем случае не должно превышать $n^2/2 + O(n)$, а пересылок элементов – $3n^2/2 + O(n)$. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

5. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ вещественных чисел и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*сортировка нахождением минимума*):

   Найти элемент массива, имеющий наименьшее значение, переставить его с первым элементом, затем проделать то же самое, начав со второго элемента и т.д.

   Общее количество сравнений в наихудшем случае не должно превышать $n^2/2 + O(n)$, а пересылок элементов – $3n + O(1)$. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

6. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ вещественных чисел и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*сортировка линейной вставкой*):

   Просматривать последовательно $a[1], \ldots, a[n-1]$ и каждый новый элемент $a[i]$ вставлять на подходящее место в уже упорядоченную совокупность $a[0], \ldots, a[i-1]$. Это место определяется последовательным сравнением $a[i]$ с упорядоченными элементами $a[0], \ldots, a[i-1]$.

   Общее количество сравнений и пересылок элементов в наихудшем случае не должно превышать $n^2/2 + O(n)$ (каждое). Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

7. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ вещественных чисел и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*сортировка двоичной вставкой*):

   Просматривать последовательно $a[1], \ldots, a[n-1]$ и каждый новый элемент $a[i]$ вставлять на подходящее место в уже упорядоченную совокупность $a[0], \ldots, a[i-1]$. Это место определяется алгоритмом деления пополам (см. задачу 1).

   Общее количество сравнений в наихудшем случае не должно превышать $n \log_2 n + O(n)$, а пересылок элементов – $n^2/2 + O(n)$. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

8. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ и вспомогательный массив $b[n]$ вещественных чисел, и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*сортировка Неймана*):

   Вначале весь массив рассматривается как совокупность упорядоченных групп по одному элементу в каждом. Слиянием соседних групп (см. задачу 2) получаются упорядоченные группы, каждая из которых содержит два элемента (кроме, может быть, последней группы, которой не нашлось парной). Далее упорядоченные группы укрупняются тем же способом и т.д. Используется вспомогательный массив b[n].

   Общее количество сравнений и пересылок элементов в наихудшем случае не должно превышать $n\log_2 n + O(n)$ (каждое). Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

9. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ вещественных чисел и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*"быстрая"сортировка*):

   Полагаем $x = a[n/2]$. Просматривая массив с начала (линейным поиском), находим i такое, что $a[i] \ge x$. Просматривая массив с конца (линейным поиском), находим j такое, что $a[j] \le x$. Если $i \le j$, то меняем $a[i]$ и $a[j]$ местами, i увеличиваем на 1, j уменьшаем на 1. Повторяем эту процедуру пока $i \le j$. После этого все элементы $a[0],\ldots,a[j]$ будут меньше всех элементов $a[i],\ldots,a[n-1]$. Затем сортируем каждую из частей (т.е. $a[0],\ldots,a[j]$ и $a[i],\ldots,a[n-1]$) этим же алгоритмом. Для уменьшения глубины рекурсии вначале сортируем более короткий массив. Затем вместо повторного вызова процедуры переходим к ее началу с новыми значениями a и n.

   Общее количество сравнений в наихудшем случае не должно превышать $n^2/2 + O(n)$, а пересылок элементов – $3n^2/2 + O(n)$. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

10. Написать подпрограмму, получающую в качестве аргументов массив $a[n]$ вещественных чисел и целое число n, и переставляющую элементы массива $a[n]$ в неубывающем порядке: $a[0] \le a[1] \le \cdots \le a[n-1]$ по следующему алгоритму (*"турнирная"сортировка или алгоритм heapsort*):

    Массив $a[n]$ рассматриваем как бинарное дерево с корнем $a[0]$. Для элемента с номером $a[k]$ потомками являются элементы $a[2*k+1]$ и $a[2*k+2]$, а родителем – элемент $a[(k-1)/2]$. На первом этапе алгоритма превращаем наше бинарное дерево в т.н. упорядоченную пирамиду, т.е. в дерево, в котором всякая цепочка от корня до любого конечного элемента выстроена по убыванию. Для этого для всех $k = 1,\ldots,n-1$ идем от элемента $a[k]$ по цепочке его родителей, продвигая его на свое место (подобно тому, как это делалось в алгоритме сортировки линейной вставкой). После этого этапа алгоритма в корне дерева (т.е. в $a[0]$) будет находиться максимальный элемент массива. На втором этапе алгоритма для всех $k = n-1,\ldots,1$: меняем корень (т.е. $a[0]$) и элемент $a[k]$ местами и рассматриваем массив a как имеющий длину k. В этом массиве восстанавливаем структуру упорядоченной пирамиды, нарушенную помещением элемента $a[k]$ в $a[0]$. Для этого идем от элемента $a[0]$ по цепочкам его потомков, продвигая его на свое место (подобно тому, как это делалось в алгоритме сортировки линейной вставкой, с одним отличием: элемент пирамиды должен быть больше обоих своих потомков).

    Общее количество сравнений в наихудшем случае не должно превышать $4n\log_2 n + O(n)$, а пересылок элементов – $2n\log_2 n + O(n)$. Основная программа должна заполнять данными массив (из файла или случайными числами), выводить его на экран, вызывать эту подпрограмму и выводить на экран результат ее работы.

4 Строки

1. Написать функцию

   ```
   char *strcpy_ (char *string1, const char *string2);
   ```

   которая копирует строку, адрес которой определяется значением аргумента `string2`, включая завершающий нулевой символ, по адресу, определяемому аргументом `string1`, и возвращает указатель на измененную строку (т.е. `string1`). Основная программа должна выделять память под строки, вводить с клавиатуры строку `string2`, вызывать эту подпрограмму и выводить на экран результат ее работы.
2. Написать функцию

   ```
   char *strcat_ (char *string1, const char *string2);
   ```

   которая добавляет строку, адрес которой определяется значением аргумента `string2`, включая завершающий нулевой символ, в конец строки, адрес которой определяется значением аргумента `string1`, и возвращает указатель на сцепленную строку (значение аргумента `string1`). Основная программа должна выделять память под строки, вводить с клавиатуры строки `string1` и `string2`, вызывать эту подпрограмму и выводить на экран результат ее работы.
3. Написать функцию

   ```
   int strcmp_ (const char *string1, const char *string2);
   ```

   которая сравнивает строки, адреса которых задаются значениями аргументов `string1` и `string2`, лексикографически и возвращает значение, определяющее их соотношение:

   - <0, если `string1` лексикографически меньше, чем `string2`,
   - =0, если `string1` лексикографически равна `string2`,
   - >0, если `string1` лексикографически больше, чем `string2`.

   Основная программа должна выделять память под строки, вводить с клавиатуры строки `string1` и `string2`, вызывать эту подпрограмму и выводить на экран результат ее работы.
4. Написать функцию

   ```
   char *strchr_ (const char *string, int ch);
   ```

   которая возвращает указатель на первое вхождение символа, имеющего код `ch`, в строку, адрес которой определяется значением аргумента `string` (возвращается 0, если символ не найден). Завершающий нулевой символ включается в поиск. Основная программа должна выделять память под строку, вводить ее и символ `ch` с клавиатуры, вызывать эту подпрограмму и выводить на экран результат ее работы.
5. Написать функцию

   ```
   char *strrchr_ (const char *string, int ch);
   ```

   которая возвращает указатель на последнее вхождение символа, имеющего код `ch`, в строку, адрес которой определяется значением аргумента `string` (возвращается 0, если символ не найден). Завершающий нулевой символ включается в поиск. Основная программа должна выделять память под строку, вводить ее и символ `ch` с клавиатуры, вызывать эту подпрограмму и выводить на экран результат ее работы.
6. Написать функцию

   ```
   int strcspn_ (const char *string1, const char *string2);
   ```

   которая возвращает индекс первого символа в строке, адрес которой определяется значением аргумента `string1`, который принадлежит набору символов, содержащихся в строке, адрес которой определяется значением аргумента `string2` (например, если строка `string1` не содержит ни одного символа из `string2`, то возвращается значение, равное длине `string1`, а если строка `string1` начинается с символа из `string2`, то возвращается 0). Основная программа должна выделять память под строки, вводить с клавиатуры строки `string1` и `string2`, вызывать эту подпрограмму и выводить на экран результат ее работы.

7. Написать функцию

```
int strspn_ (const char *string1, const char *string2);
```

которая возвращает индекс первого символа в строке, адрес которой определяется значением аргумента `string1`, который не принадлежит набору символов, содержащихся в строке, адрес которой определяется значением аргумента `string2` (например, если строка `string1` содержит только символы из набора символов в строке `string2`, то возвращается значение, равное длине `string1`, а если строка `string1` начинается с символа не из набора символов в строке `string2`, то возвращается 0). Основная программа должна выделять память под строки, вводить с клавиатуры строки `string1` и `string2`, вызывать эту подпрограмму и выводить на экран результат ее работы.

8. Написать функцию

```
char *strstr_ (const char *string1, const char *string2);
```

которая возвращает указатель на первое вхождение строки, адрес которой определяется значением аргумента `string2`, в строке, адрес которой определяется значением аргумента `string1` (возвращается 0, если вхождение не найдено). Основная программа должна выделять память под строки, вводить с клавиатуры строки `string1` и `string2`, вызывать эту подпрограмму и выводить на экран результат ее работы.

5 Текстовые файлы

1. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьные строки s и r, и выводящую в файл b строки файла a, заменяя каждое вхождение строки s на строку r. Функция возвращает количество измененных строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..
2. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьные строки s и t, и выводящую в файл b те строки файла a, которые не совпадают со строкой s; при этом пробельные символы не учитываются (пробельным называется символ, содержащийся в строке t). Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..
3. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьные строки s и t, и выводящую в файл b те строки файла a, которые состоят только из символов строки s; при этом пробельные символы не учитываются (пробельным называется символ, содержащийся в строке t). Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..
4. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьные строки s и t, и выводящую в файл b те строки файла a, в которые входит строка s; при этом пробельные символы не учитываются (пробельным называется символ, содержащийся в строке t). Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..
5. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьные строки s и t, и выводящую в файл b те строки файла a, которые имеют общее слово со строкой s (словом называется последовательность символов, не содержащая пробельных символов, пробельным называется символ, содержащийся в строке t). Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..

6. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьную строку s, и выводящую в файл b те строки файла a, которые совпадают со строкой s; при этом символ '.' в строке s соответствует любому символу строки из файла a, символам '.' и '\' соответствуют последовательности символов '\.' и '\\' в строке s. Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..
7. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьную строку s, и выводящую в файл b те строки файла a, которые совпадают со строкой s; при этом символ '$*$' в строке s означает, что предыдущий символ строки s может учитываться 0 или более раз, символам '$*$' и '\' соответствуют последовательности символов '\$*$' и '\\' в строке s. Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..
8. Написать функцию, получающую в качестве аргументов имя a текстового файла неизвестной длины, имя файла b для вывода информации и символьную строку s, и выводящую в файл b те строки файла a, которые совпадают со строкой s; при этом последовательность '[n–m]' (n, m – символы) в строке s соответствует любому символу строки из файла a, имеющему код в диапазоне $n \ldots m$, символам '[', ']' и '\' соответствуют последовательности символов '\[', '\]' и '\\' в строке s. Функция возвращает количество таких строк или -1, -2 и т.д., если она не смогла открыть файл, прочитать элемент и т.д..

6 Интерполяция

1. Написать функцию, получающую в качестве аргументов целое число n, вещественное число x_0 и массивы вещественных чисел $x[n]$, $y[n]$, и возвращающую значение интерполяционного многочлена Лагранжа, построенного по точкам $x[n]$, $y[n]$, в точке x_0. Многочлен Лагранжа строится по его определению. Основная программа должна заполнять массивы $x[n]$, $y[n]$ из файла, вводить с клавиатуры число x_0, вызывать эту функцию и выводить на экран результат ее работы.
2. Написать функцию, получающую в качестве аргументов целое число n, вещественное число x_0 и массивы вещественных чисел $x[n]$, $y[n]$, и возвращающую значение интерполяционного многочлена Лагранжа, построенного по точкам $x[n]$, $y[n]$, в точке x_0. Многочлен Лагранжа строится по интерполяционной формуле Ньютона с разделенными разностями. Основная программа должна заполнять массивы $x[n]$, $y[n]$ из файла, вводить с клавиатуры число x_0, вызывать эту функцию и выводить на экран результат ее работы.
3. Написать функцию, получающую в качестве аргументов целое число n, вещественное число x_0 и массивы вещественных чисел $x[n]$, $y[n]$, и возвращающую значение интерполяционного многочлена Лагранжа, построенного по точкам $x[n]$, $y[n]$, в точке x_0. Многочлен Лагранжа строится по схеме Эйткена. Основная программа должна заполнять массивы $x[n]$, $y[n]$ из файла, вводить с клавиатуры число x_0, вызывать эту функцию и выводить на экран результат ее работы.
4. Написать функцию, получающую в качестве аргументов целое число n, вещественное число x_0 и массивы вещественных чисел $x[n]$, $y[n]$, $d[n]$ и возвращающую значение в точке x_0 интерполяционного многочлена Лагранжа, построенного по точкам $x[n]$, $y[n]$ и имеющего производные $d[n]$. Многочлен Лагранжа строится по интерполяционной формуле Ньютона с кратными узлами. Основная программа должна заполнять массивы $x[n]$, $y[n]$, $d[n]$ из файла, вводить с клавиатуры число x_0, вызывать эту функцию и выводить на экран результат ее работы.
5. Написать функцию, получающую в качестве аргументов вещественные числа x и ε, и возвращающую значение функции sin в точке x, вычисленное с заданной точностью ε суммированием ряда Тейлора. Основная программа должна в цикле вводить с клавиатуры вещественные числа x и ε, вызывать эту функцию и выводить на экран результат ее работы и модуль разности между полученным приближенным значением $\sin(x)$ и истинным значением $\sin(x)$, заканчивая работать при вводе $\varepsilon < 0$.

6. Написать функцию, получающую в качестве аргументов вещественные числа x и ε, и возвращающую значение функции cos в точке x, вычисленное с заданной точностью ε суммированием ряда Тейлора. Основная программа должна в цикле вводить с клавиатуры вещественные числа x и ε, вызывать эту функцию и выводить на экран результат ее работы и модуль разности между полученным приближенным значением $\cos(x)$ и истинным значением $\cos(x)$, заканчивая работать при вводе $\varepsilon < 0$.
7. Написать функцию, получающую в качестве аргументов вещественные числа x и ε, и возвращающую значение функции exp в точке x, вычисленное с заданной точностью ε суммированием ряда Тейлора. Основная программа должна в цикле вводить с клавиатуры вещественные числа x и ε, вызывать эту функцию и выводить на экран результат ее работы и модуль разности между полученным приближенным значением $\exp(x)$ и истинным значением $\exp(x)$, заканчивая работать при вводе $\varepsilon < 0$.
8. Написать функцию, получающую в качестве аргументов вещественные числа x и ε, и возвращающую значение функции log в точке x, вычисленное с заданной точностью ε суммированием ряда Тейлора. Основная программа должна в цикле вводить с клавиатуры вещественные числа x и ε, вызывать эту функцию и выводить на экран результат ее работы и модуль разности между полученным приближенным значением $\log(x)$ и истинным значением $\log(x)$, заканчивая работать при вводе $\varepsilon < 0$.

7 Уравнения

1. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x) = 0$ на отрезке $[a,b]$, вычисленный с точностью ε. Корень находится методом деления пополам. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.
2. Написать функцию, получающую в качестве аргументов функцию f, ее производную d, вещественные числа x_0, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x) = 0$, находящийся в окрестности точки x_0 и вычисленный с точностью ε. Корень находится методом Ньютона. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа x_0 и ε, вызывать эту функцию и выводить на экран результат ее работы.
3. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x) = 0$ на отрезке $[a,b]$, вычисленный с точностью ε. Корень находится методом хорд. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.
4. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x) = 0$, находящийся на отрезке $[a,b]$ или в его окрестности и вычисленный с точностью ε. Корень находится методом секущих. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.
5. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x) = 0$, находящийся на отрезке $[a,b]$ или в его окрестности и вычисленный с точностью ε. Корень находится методом обратной квадратичной интерполяции. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.

6. Написать функцию, получающую в качестве аргументов функцию f, целое число m, массив d вещественных чисел длины $3(m+1)$, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x)=0$, находящийся на отрезке $[a,b]$ или в его окрестности и вычисленный с точностью ε. Корень находится методом обратной интерполяции порядка m. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры целое число m, вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.
7. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа x_0, ε и адрес вещественного числа x, и возвращающую в переменной x корень уравнения $f(x)=x$, находящийся в окрестности точки x_0 и вычисленный с точностью ε. Корень находится методом последовательных приближений. Функция возвращает отрицательное значение, если она не смогла обнаружить корень, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа x_0 и ε, вызывать эту функцию и выводить на экран результат ее работы.
8. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x максимальное значение функции f на отрезке $[a,b]$, вычисленное с точностью ε. Максимальное значение находится линейным поиском с изменением шага. Функция возвращает отрицательное значение, если она не смогла найти максимум, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.
9. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x максимальное значение функции f на отрезке $[a,b]$, вычисленное с точностью ε. Максимальное значение находится методом золотого сечения. Функция возвращает отрицательное значение, если она не смогла найти максимум, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.
10. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа x, и возвращающую в переменной x максимальное значение функции f, находящееся на отрезке $[a,b]$ или в его окрестности и вычисленное с точностью ε. Максимальное значение находится методом квадратичной интерполяции. Функция возвращает отрицательное значение, если она не смогла найти максимум, и число итераций иначе. Основная программа должна вводить с клавиатуры вещественные числа a, b и ε, вызывать эту функцию и выводить на экран результат ее работы.

8 Интегралы

1. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b и целое число n, и возвращающую приближенное значение $\int_a^b f(x)\,dx$, которое находится по составной формуле трапеций:

$$\int_a^b f(x)\,dx = h(f(a)/2 + f(a+h) + f(a+2h) + \cdots + f(a+(n-1)h) + f(b)/2),$$

где $h=(b-a)/n$. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b и целое число n, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $n<0$.

2. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b и целое число n, и возвращающую приближенное значение $\int_a^b f(x)\,dx$, которое находится по составной формуле Симпсона:

$$\int_a^b f(x)\,dx = (2/3)h(f(a)/2 + 2f(a+h) + f(a+2h) + 2f(a+3h) + \\ + f(a+4h) + \cdots + f(a+(2n-2)h) + 2f(a+(2n-1)h) + f(b)/2),$$

где $h = (b-a)/(2n)$. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b и целое число n, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $n < 0$.

3. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b и целое число n, и возвращающую приближенное значение $\int_a^b \frac{f(x)}{\sqrt{|x|}}, dx$, которое находится по составной формуле трапеций с весом $1/\sqrt{|x|}$. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b и целое число n, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $n < 0$.
4. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b и целое число n, и возвращающую приближенное значение $\int_a^b \frac{f(x)}{\sqrt{|x|}}, dx$, которое находится по составной формуле Симпсона с весом $1/\sqrt{|x|}$. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b и целое число n, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $n < 0$.
5. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа r, и возвращающую в переменной r значение $\int_a^b f(x)\,dx$, вычисленное с заданной точностью ε по составной формуле трапеций с автоматическим выбором шага. Функция возвращает отрицательное значение, если она не смогла вычислить интеграл, и конечное значение n иначе. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b, ε, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $\varepsilon < 0$.
6. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, b, ε и адрес вещественного числа r, и возвращающую в переменной r значение $\int_a^b f(x)\,dx$, вычисленное с заданной точностью ε по составной формуле Симпсона с автоматическим выбором шага. Функция возвращает отрицательное значение, если она не смогла вычислить интеграл, и конечное значение n иначе. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b, ε, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $\varepsilon < 0$.
7. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, ε и адрес вещественного числа r, и возвращающую в переменной r значение $\int_a^{+\infty} f(x)\,dx$, вычисленное с заданной точностью ε по составной формуле трапеций с автоматическим выбором шага. Функция возвращает отрицательное значение, если она не смогла вычислить интеграл, и конечное значение b иначе. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, ε, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $\varepsilon < 0$.
8. Написать функцию, получающую в качестве аргументов функцию f, вещественные числа a, ε и адрес вещественного числа r, и возвращающую в переменной r значение $\int_a^{+\infty} f(x)\,dx$, вычисленное с заданной точностью ε по составной формуле Симпсона с автоматическим выбором шага. Функция возвращает отрицательное значение, если она не смогла вычислить интеграл, и конечное значение b иначе. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, ε, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $\varepsilon < 0$.

9. Написать функцию, получающую в качестве аргументов функции x и y, вещественные числа a, b, ε и адрес вещественного числа r, и возвращающую в переменной r значение длины кривой $(x(t), y(t))$ в пределах изменения t от a до b, вычисленное с заданной точностью ε. Длина кривой находится как предел сумм длин ломаных с вершинами

$$(x(a), y(a)), (x(a+h), y(a+h)), (x(a+2h), y(a+2h)), \ldots (x(a+(n-1)h), y(a+(n-1)h)), (x(b), y(b))$$

при $n \to \infty$, где $h = (b-a)/n$. Функция возвращает отрицательное значение, если она не смогла вычислить длину кривой, и конечное значение n иначе. Основная программа должна в цикле вводить с клавиатуры вещественные числа a, b, ε, вызывать эту функцию и выводить на экран результат ее работы, заканчивая работать при вводе $\varepsilon < 0$.

9 Матрицы

1. Написать подпрограмму, получающую в качестве аргументов $n \times n$ массив a вещественных чисел и целое число n, и заменяющую матрицу a на ее транспонированную. Основная программа должна вводить число n и массив a (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.
2. Написать подпрограмму, получающую в качестве аргументов $n \times n$ массив a вещественных чисел и целое число n, и заменяющую матрицу a на матрицу $(a + a^t)/2$, где a^t - транспонированная матрица a. Основная программа должна вводить число n и массив a (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.
3. Написать подпрограмму, получающую в качестве аргументов $n \times n$ массив a вещественных чисел и целое число n, и заменяющую матрицу a на матрицу $(a - a^t)/2$, где a^t - транспонированная матрица a. Основная программа должна вводить число n и массив a (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.
4. Написать подпрограмму, получающую в качестве аргументов $m \times n$ массив a вещественных чисел и целые числа m, n, i и j, и переставляющую местами в матрице а i-ю и j-ю строки. Основная программа должна вводить число n и массив a (из файла или по заданной формуле), вводить с клавиатуры числа i и j, вызывать эту подпрограмму и выводить на экран результат ее работы.
5. Написать подпрограмму, получающую в качестве аргументов $m \times n$ массив a вещественных чисел и целые числа m, n, i и j, и переставляющую местами в матрице а i-й и j-й столбцы. Основная программа должна вводить число n и массив a (из файла или по заданной формуле), вводить с клавиатуры числа i и j, вызывать эту подпрограмму и выводить на экран результат ее работы.
6. Написать подпрограмму, получающую в качестве аргументов $m \times n$ массив a вещественных чисел, целые числа m, n, i, j и вещественное число b, и прибавляющую к j-й строке матрицы а i-ю строку, умноженную на число b. Основная программа должна вводить число n и массив a (из файла или по заданной формуле), вводить с клавиатуры числа i, j и b, вызывать эту подпрограмму и выводить на экран результат ее работы.
7. Написать подпрограмму, получающую в качестве аргументов $m \times n$ массив a вещественных чисел, вектор b длины n, вектор c длины m и целые числа m, n, и заменяющую вектор c на вектор, равный произведению матрицы a на вектор b ($c = ab$). Основная программа должна вводить числа n, m, массивы a и вектор b (из файла или по заданным формулам), вызывать эту подпрограмму и выводить на экран результат ее работы.
8. Написать подпрограмму, получающую в качестве аргументов $m \times n$ массив a вещественных чисел, $n \times k$ массив b вещественных чисел, $m \times k$ массив c вещественных чисел, и целые числа m, n, k, и заменяющую массив c на массив, равный произведению матрицы a на матрицу b ($c = ab$). Основная программа должна вводить числа n, m, k и массивы a, b (из файла или по заданным формулам), вызывать эту подпрограмму и выводить на экран результат ее работы.
9. Написать функцию, получающую в качестве аргументов $n \times n$ матрицу A, вектор x_0, целые числа n и m, и возвращающую m-й член последовательности $\{\lambda_k\}$, где $\lambda_k = (Ax_k, x_k)/(x_k, x_k)$, $x_k = Ax_{k-1}$, $(\cdot,\cdot)$ — евклидово скалярное произведение.

10. Написать подпрограмму, получающую в качестве аргументов $n \times n$ матрицу A, вектора x_0, b, x, целые числа n, m и вещественное число τ, и возвращающую в векторе x m-й член последовательности $\{x_k\}$, где $(x_k - x_{k-1})/\tau + Ax_{k-1} = b$.
11. Написать подпрограмму, получающую в качестве аргументов $n \times n$ матрицу A, вектора x_0, b, x, целые числа n и m, и возвращающую в векторе x m-й член последовательности $\{x_k\}$, где $(x_k - x_{k-1})/\tau_{k-1} + Ax_{k-1} = b$, $\tau_k = (r_k, r_k)/(Ar_k, Ar_k)$, $r_k = Ax_k - b$, $(\cdot,\cdot)$ — евклидово скалярное произведение.
12. Написать подпрограмму, получающую в качестве аргументов $n \times n$ матрицу A, вектора x_0, b, x, целые числа n и m, и возвращающую в векторе x m-й член последовательности $\{x_k\}$, где $(x_k - x_{k-1})/\tau_{k-1} + Ax_{k-1} = b$, $\tau_k = (Ar_k, r_k)/(Ar_k, Ar_k)$, $r_k = Ax_k - b$, $(\cdot,\cdot)$ — евклидово скалярное произведение.
13. Написать подпрограмму, получающую в качестве аргументов $n \times n$ матрицу A, вектора x_0, b, x, целые числа n, m и вещественное число τ, и возвращающую в векторе x m-й член последовательности $\{x_k\}$, где $D(x_k - x_{k-1})/\tau + Ax_{k-1} = b$, D — диагональ матрицы A.
14. Написать подпрограмму, получающую в качестве аргументов $n \times n$ матрицу A, вектора x_0, b, x, целые числа n, m и вещественное число τ, и возвращающую в векторе x m-й член последовательности $\{x_k\}$, где $L(x_k - x_{k-1})/\tau + Ax_{k-1} = b$, L — нижняя треугольная часть матрицы матрицы A (включая диагональ).
15. Написать подпрограмму, получающую в качестве аргументов $n \times n$ матрицу A, вектора x_0, b, x, целые числа n, m и вещественное число τ, и возвращающую в векторе x m-й член последовательности $\{x_k\}$, где $U(x_k - x_{k-1})/\tau + Ax_{k-1} = b$, U — верхняя треугольная часть матрицы матрицы A (включая диагональ).
16. Написать функцию, получающую в качестве аргументов $m \times n$ массив a вещественных чисел и целые числа m, n, и возвращающую $\max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}|$. Основная программа должна вводить числа m, n и массив a (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.
17. Написать функцию, получающую в качестве аргументов $m \times n$ массив a вещественных чисел и целые числа m, n, и возвращающую $\max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|$. Основная программа должна вводить числа m, n и массив a (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.
18. Написать функцию, получающую в качестве аргументов $m \times n$ массив A вещественных чисел, вектор x длины n, вектор b длины m и целые числа m, n, и возвращающую $\sum_{k=1}^{m} |r_k|$, где $r = Ax - b$. Основная программа должна вводить числа m, n и массивы a, b, x (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.
19. Написать функцию, получающую в качестве аргументов $m \times n$ массив A вещественных чисел, вектор x длины n, вектор b длины m и целые числа m, n, и возвращающую $\max_{k=1,\ldots,m} |r_k|$, где $r = Ax - b$. Основная программа должна вводить числа m, n и массивы a, b, x (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.
20. Написать функцию, получающую в качестве аргументов $m \times n$ массив A вещественных чисел, $n \times m$ массив B вещественных чисел, и целые числа m, n, и возвращающую $\max_{i=1,\ldots,m} \sum_{j=1}^{m} |r_{ij}|$, где $R = AB - I$, I – единичная матрица. Основная программа должна вводить числа m, n и массивы a, b (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.
21. Написать функцию, получающую в качестве аргументов $m \times n$ массив A вещественных чисел, $n \times m$ массив B вещественных чисел, и целые числа m, n, и возвращающую $\max_{j=1,\ldots,m} \sum_{i=1}^{m} |r_{ij}|$, где $R = AB - I$, I – единичная матрица. Основная программа должна вводить числа m, n и массивы a, b (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.