Задания к работе №1 по Фундаментальным алгоритмам.

Все задания реализуются на языке программирования *C* (стандарт *C99* и выше).

Реализованные в заданиях приложения не должны завершаться аварийно.

Во всех заданиях запрещено пользоваться функциями, позволяющими завершить выполнение приложения из произвольной точки выполнения, вне контекста исполнения функции *main*.

Во всех заданиях при реализации необходимо разделять контексты работы с данными (поиск, сортировка, добавление/удаление, модификация и т. п.) и отправка данных в поток вывода / выгрузка данных из потока ввода.

Во всех заданиях все параметры функций и вводимые (с консоли, файла, командной строки) пользователем данные должны подвергаться валидации в соответствии с типом валидируемых данных, если не сказано обратное; валидация должна зависеть от типа данных и логики применения этих данных для выполнения целевой подзадачи. При передаче аргументов приложению в командную строку, их количество также должно валидироваться.

Во всех заданиях необходимо контролировать ситуации с невозможностью [пере]выделения памяти; во всех заданиях необходимо корректно освобождать всю выделенную динамическую память.

Все ошибки, связанные с операциями открытия файла, должны быть обработаны; все открытые файлы должны быть закрыты.

Во всех заданиях запрещено использование глобальных переменных.

Во всех заданиях при реализации функций необходимо обеспечить возможность обработки ошибок различных типов на уровне вызывающего кода при помощи возврата целевых результатов функции через параметры функции и возврата из функции значения (либо типа *int*, либо перечислимого типа (*enum*)), репрезентирующего статус-код функции, в целях обработки последнего в вызывающем коде через оператор *switch/case* либо через управляющие конструкции языка *if / else if / else*. Возвращаемые статус-коды функций необходимо продумать самостоятельно так, чтобы были покрыты всевозможные ошибки времени выполнения функций.

Во всех заданиях сравнение (на предмет эквивалентности или отношения порядка) вещественных чисел на уровне функции должно использовать значение эпсилон, которое является параметром этой функции.

Во всех заданиях при реализации функций необходимо максимально ограничивать возможность модификации (если она не подразумевается) передаваемых в функцию параметров (используйте ключевое слово *const*).

1. Через аргументы командной строки программе подаются строковое представление числа *x* и флаг, определяющий действие с этим числом. Флаг начинается с символа ‘-’ или ‘/’. Программа распознает следующие флаги:
   - -h вывести в консоль натуральные числа в пределах 100 включительно, кратные *x*. Если таковых нету – вывести соответствующее сообщение;
   - -p определить, является ли число *x* простым; является ли *x* составным;
   - -s разделить число *x* на отдельные цифры системы счисления с основанием *16* и вывести отдельно каждую цифру числа, разделяя их пробелом, от старших разрядов к младшим, без ведущих нулей в строковом представлении;
   - -e вывести таблицу степеней (для всех показателей в диапазоне от *1* до *x*) оснований от *1* до *10*; для этого флага работает ограничение на вводимое число: *x* должен быть не больше *10*;
   - -a вычислить сумму всех натуральных чисел от *1* до *x* и вывести полученное значение в консоль;
   - -f вычислить факториал *x* и вывести полученное значение в консоль.

2. Реализовать функции, вычисляющие значения чисел e, π, $\ln 2$, $\sqrt{2}$, γ с заданной точностью. Для каждой константы реализовать три способа вычисления: как сумму ряда, как решение специального уравнения, как значение предела.

   Замечание. Вы можете использовать следующие факты:

	Предел	Ряд/Произведение	Уравнение
e	$e = \lim\limits_{n\to\infty}\left(1+\frac{1}{n}\right)^n$	$e = \sum\limits_{n=0}^{\infty}\frac{1}{n!}$	$\ln x = 1$
π	$\pi = \lim\limits_{n\to\infty}\frac{\left(2^n n!\right)^4}{n((2n)!)^2}$	$\pi = 4\sum\limits_{n=1}^{\infty}\frac{(-1)^{n-1}}{2n-1}$	$\cos x = -1$
$\ln 2$	$\ln 2 = \lim\limits_{n\to\infty} n\left(2^{\frac{1}{n}}-1\right)$	$\ln 2 = \sum\limits_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}$	$e^x = 2$
$\sqrt{2}$	$\sqrt{2} = \lim\limits_{n\to\infty} x_n$, где $x_{n+1} = x_n - \frac{x_n^2}{2} + 1,$ $x_0 = -0.5$	$\sqrt{2} = \prod\limits_{k=2}^{\infty} 2^{2^{-k}}$	$x^2 = 2$
γ	$\gamma = \lim\limits_{m\to\infty}\left(\sum\limits_{k=1}^{m} C_m^k \frac{(-1)^k}{k}\ln(k!)\right)$	$\gamma = -\frac{\pi^2}{6} + \sum\limits_{k=2}^{\infty}\left(\frac{1}{\lfloor\sqrt{k}\rfloor^2}-\frac{1}{k}\right)$	$e^{-x} = \lim\limits_{t\to\infty}\left(\ln t \prod\limits_{p\le t,\ p\in P}\frac{p-1}{p}\right)$

   Точность вычислений (значение эпсилон) подаётся программе в качестве аргумента командной строки. Продемонстрируйте выполнение реализованных функций.

3. Через аргументы командной строки программе подается флаг, который определяет действие, и набор чисел. Флаг начинается с символа ‘-’ или ‘/’. Необходимо проверять соответствие количества параметров введённому флагу. Программа распознает следующие флаги:
   - -q первый параметр (вещественное число) задаёт точность сравнения вещественных чисел (эпсилон), оставшиеся три (вещественные числа) являются коэффициентами квадратного уравнения; необходимо вывести в консоль решения этого уравнения при всевозможных уникальных перестановках значений коэффициентов при степенях переменной;
   - -m необходимо задать два ненулевых целых числа, после чего определить, кратно ли первое число второму;
   - -t первый параметр (вещественное число) задаёт точность сравнения вещественных чисел (эпсилон); необходимо проверить, могут ли оставшиеся три (вещественные числа) параметра являться длинами сторон прямоугольного треугольника.

4. На вход программе, через аргументы командной строки, подается флаг и путь к файлу. Флаг определяет действие с входным файлом. Флаг начинается с символа ‘-’ или ‘/’. Если флаг содержит в качестве второго символа опциональный символ ‘n’ (то есть “-nd”, “/nd”, “-ni”, “/ni”, “-ns”, “/ns”, “-na”, “/na”), то путь к выходному файлу является третьим аргументом командной строки; иначе имя выходного файла генерируется приписыванием к имени входного файла префикса “out_”. Вывод программы должен транслироваться в выходной файл. Программа распознает следующие флаги:
   - -d необходимо исключить символы арабских цифр из входного файла;
   - -i для каждой строки входного файла в выходной файл необходимо записать сколько раз в этой строке встречаются символы букв латинского алфавита;
   - -s для каждой строки входного файла в выходной файл необходимо записать сколько раз в этой строке встречаются символы, отличные от символов букв латинского алфавита, символов арабских цифр и символа пробела;
   - -a необходимо заменить символы, отличные от символов цифр, ASCII кодом, записанным в системе счисления с основанием 16.

5. Вычислить значения сумм с точностью ε, где ε (вещественное число) подаётся программе в виде аргумента командной строки:
   a. $\sum_{n=0}^{\infty} \frac{x^n}{n!}$;
   b. $\sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$;
   c. $\sum_{n=0}^{\infty} \frac{3^{3n}(n!)^3 x^{2n}}{(3n)!}$;
   d. $\sum_{n=1}^{\infty} \frac{(-1)^n (2n-1)!! x^{2n}}{(2n)!!}$.

6. Вычислить значения интегралов с точностью ε, где ε (вещественное число) подаётся программе в виде аргумента командной строки:
   a. $\int_0^1 \frac{ln(1+x)}{x} dx$
   b. $\int_0^1 e^{-\frac{x^2}{2}} dx$
   c. $\int_0^1 ln\,\frac{1}{1-x}\, dx$
   d. $\int_0^1 x^x dx$

7. На вход программе подаётся флаг и пути к файлам. Флаг начинается с символа ‘-’ или “/”. Необходимо проверять соответствие количества параметров введённому флагу. Программа распознает следующие флаги:
   - -r записать в файл, путь к которому подаётся последним аргументом командной строки, лексемы из файлов *file1* и *file2*, пути к которым подаются как третий и четвёртый аргументы командной строки соответственно, где на нечётных позициях находятся лексемы из *file1*, а на чётных – из *file2*. Лексемы во входных файлах разделяются произвольным количеством символов пробела, табуляций и переносов строк. Если количество лексем в файлах различается, после достижения конца одного из входных файлов, необходимо переписать в выходной файл оставшиеся лексемы из другого из входных файлов. Лексемы в выходном файле должны быть разделены одним символом пробела.
   - -a записать в файл, путь к которому подаётся последним аргументом командной строки, файл, путь к которому подаётся третьим аргументом командной строки, таким образом, чтобы:
     - в каждой десятой лексеме сначала все символы букв латинского алфавита были преобразованы в эквивалентные символы строчных букв латинского алфавита, а затем все символы были преобразованы в эквивалентные им ASCII-коды, записанные в системе счисления с основанием 4;
     - в каждой второй (и одновременно не десятой) лексеме все символы букв латинского алфавита были преобразованы в эквивалентные символы строчных букв латинского алфавита;
     - в каждой пятой (и одновременно не десятой) лексеме все символы были преобразованы в эквивалентные им ASCII-коды, записанные в системе счисления с основанием 8.

   Лексемы во входном файле разделяются произвольным количеством символов пробела, табуляций и переносов строк. Лексемы в выходном файле должны быть разделены одним символом пробела.

8. В текстовом файле, путь к которому подаётся как второй аргумент командной строки, находятся числа, записанные в разных системах счисления (в диапазоне [2..36]), при этом информация о конкретной системе счисления для каждого числа утеряна. В файле числа разделены произвольным количеством разделителей (символов пробела, табуляций и переносов строки). Для каждого числа из входного файла программа должна определить минимальное основание системы счисления (в диапазоне [2..36]), в которой представление этого числа корректно, и в выходной файл, путь к которому подаётся третьим аргументом командной строки, построчно выводит: входное число без ведущих нулей, определенное для него минимальное основание системы счисления и представление этого числа в системе счисления с основанием 10. Прописные и строчные символы букв латинского алфавита отождествляются.

9. 1. Заполнить массив фиксированного размера псевдослучайными числами в диапазоне [a..b], где a, b задаются в качестве аргументов командной строки. Реализовать функцию, выполняющую поиск максимального и минимального значений элементов массива и меняющую местами максимальный и минимальный элементы в исходном массиве за один проход по нему.
   2. Заполнить динамические массивы A и B псевдослучайного размера в диапазоне [10..10000] псевдослучайными числами в диапазоне [-1000..1000]. Сформировать из них динамический массив C, где i-й элемент массива C есть i-й элемент массива A (A[i]), к которому добавлено значение ближайшего к A[i] по значению элемента из массива B (если ближайших по значению элементов несколько, допустимо добавление любого из них).

10. Пользователь вводит в консоль основание системы счисления (в диапазоне [2..36]) и затем строковые представления целых чисел в системе счисления с введённым основанием (цифры со значением больше 9 должны вводиться как прописные буквы латинского алфавита). Окончанием ввода является ввод строки “Stop”. Найти среди введённых чисел максимальное по модулю. Напечатать его без ведущих нулей, а также его строковые представления в системах счисления с основаниями 9, 18, 27 и 36.