КОНТРОЛЬНАЯ РАБОТА №3.

Студенту необходимо предоставить отчёт о выполнении контрольной работы в распечатанном виде и в электронном виде, на любом носителе информации. Отчёт должен включать титульный лист и для каждой задачи: формулировку задания, текст программы, тестовый пример. В конце отчета общий вывод по контрольной работе. На титульном листе отчёта, о выполнении контрольных работ, необходимо указать фамилию, имя и отчество студента, номер учебной группы, номер варианта.

Номер варианта студенту выдается преподавателем.

ЗАДАНИЕ №1. ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

Вариант №1

Разработать программу перемножения двух матриц *A, B* размерности *n*x*n*. Все матрицы размещаются в оперативной памяти динамически, а значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходные и результирующую матрицы.

Вариант №2

Разработать программу нормирования матрицы размерности *m*x*n*, которое заключается в том, что каждый элемент в этой матрице вычисляется на основании исходной матрицы, как отношение суммы всех других элементов в его строке к сумме всех других элементов в его столбце. Матрица должна размещаться в оперативной памяти динамически, значения *m* и *n* вводятся с клавиатуры по запросу. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №3

Разработать программу, которая в матрице размерности *n*x*n* меняет местами строку, содержащую элемент с минимальным значением со столбцом, содержащим элемент с максимальным значением. Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №4

Дана действительная квадратная матрица порядка *n*, все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на

пересечении этих диагоналей. Матрица размещается в памяти, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №5

Построить квадратную матрицу порядка $2n$:

$$\begin{matrix} n\{ \\ n\{ \\ \\ \end{matrix} \begin{bmatrix} 11\ldots1 & 22\ldots2 \\ 11\ldots1 & 22\ldots2 \\ \ldots & \ldots \\ 11\ldots1 & 22\ldots2 \\ 33\ldots3 & 44\ldots4 \\ 33\ldots3 & \ldots4 \\ \ldots & \ldots \\ \underbrace{33\ldots3}_{n} & \underbrace{44\ldots}_{n} \end{bmatrix}$$

Матрица размещается в памяти динамически, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №6

Дано действительное число x. Получить квадратную матрицу порядка $n<10$:

$$\begin{bmatrix} 1 & x & \ldots & x^{n-2} & x^{n-1} \\ x & 0 & \ldots & 0 & x^{n-2} \\ \vdots & & & & \\ x^{n-2} & 0 & \ldots & 0 & x \\ x^{n-1} & x^{n-2} & \ldots & x & 1 \end{bmatrix}$$

Матрица размещается в памяти динамически, значения x и n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №7

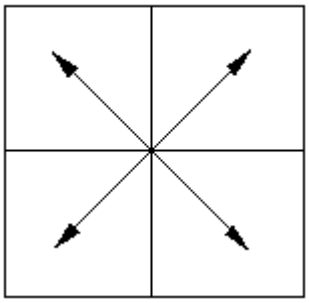
Даны действительные числа a_1, …, a_n. Получить квадратную матрицу порядка n:

$$\begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_{n-2} & a_{n-1} & a_n \\ a_2 & a_3 & a_4 & \dots & a_{n-1} & a_n & a_1 \\ a_3 & a_4 & a_5 & \dots & a_n & a_1 & a_2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_n & a_1 & a_2 & \dots & a_{n-3} & a_{n-2} & a_{n-1} \end{bmatrix}$$

Матрица размещается в памяти динамически, значение n и числа a_1, …, a_n вводятся по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №8

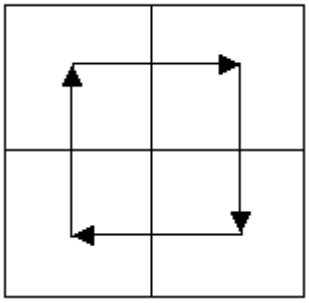
Дана действительная квадратная матрица порядка 2*n*. Получить новую матрицу, переставляя ее блоки размера *n*x*n*:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №9

Дана действительная квадратная матрица порядка 2n. Получить новую матрицу, переставляя ее блоки размера *n*x*n*:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №10

Получить квадратную матрицу порядка *n*:

$$\begin{bmatrix} n & & & \\ n-1 & n & & 0 \\ n-2 & n-1 & n & \\ \cdots & \cdots & \cdots & \\ 1 & 2 & 3 \ldots & n \end{bmatrix}$$

Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №11

Получить квадратную матрицу порядка *n*:

$$\begin{bmatrix} 1 & 2 & 3 & \ldots & n-1 & n \\ 2 & 1 & 2 & \ldots & n-2 & n-1 \\ 3 & 2 & 1 & & n-3 & n-2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ n-1 & n-2 & n-3 & \ldots & 1 & 2 \\ n & n-1 & n-2 & \ldots & 2 & 1 \end{bmatrix}$$

Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №12

Дана действительная квадратная матрица порядка *n*. Преобразовать матрицу по правилу: строку с номером *n* сделать столбцом с номером *n*, а столбец с номером *n* сделать строкой с номером *n*. Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №13

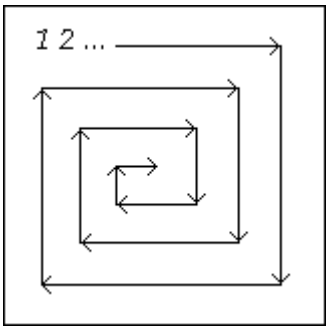
Даны две действительные квадратные матрицы порядка *n*. Получить новую матрицу умножением элементов каждой строки первой матрицы на наибольшее из значений элементов соответствующей строки второй матрицы. Матрицы размещаются в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходные и результирующую матрицы.

Вариант №14

Даны две действительные квадратные матрицы порядка n. Получить новую матрицу прибавлением к элементам каждого столбца первой матрицы произведения элементов соответствующих строк второй матрицы. Матрицы размещаются в памяти динамически, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходные и результирующую матрицы.

Вариант №15

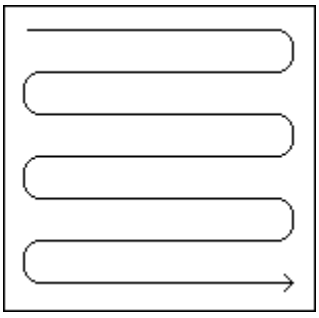
Получить целочисленную квадратную матрицу порядка $n<8$, элементами которой являются числа 1, 2, ..., n^2, расположенные в ней по спирали:



Матрица размещается в памяти динамически, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №16

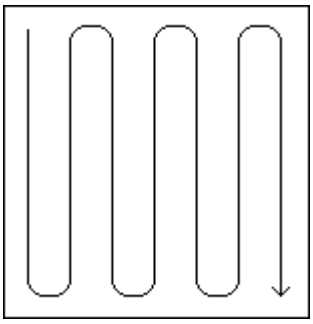
Даны действительные числа $a_1, a_2, ..., a_{n^2}$. Получить действительную квадратную матрицу порядка $n<8$, элементами которой являются числа $a_1, a_2, ..., a_{n^2}$, расположенные в ней по схеме:



Матрица размещается в памяти динамически, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №17

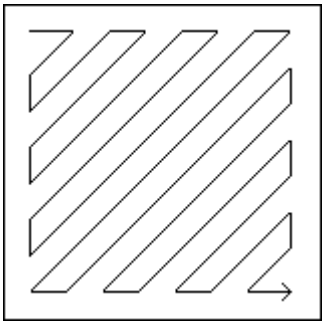
Даны действительные числа $a_1, a_2, ..., a_{n^2}$. Получить действительную квадратную матрицу порядка n<8, элементами которой являются числа $a_1, a_2, ..., a_{n^2}$, расположенные в ней по схеме:



Матрица размещается в памяти динамически, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №18

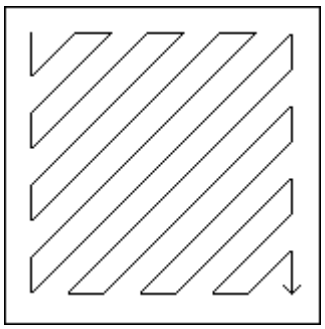
Даны действительные числа $a_1, a_2, ..., a_{n^2}$. Получить действительную квадратную матрицу порядка n<8, элементами которой являются числа $a_1, a_2, ..., a_{n^2}$, расположенные в ней по схеме:



Матрица размещается в памяти динамически, значение n вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №19

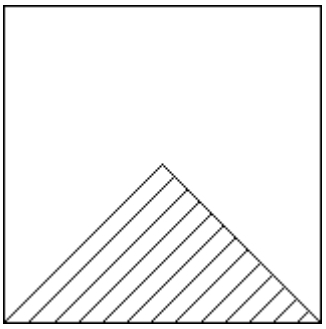
Даны действительные числа $a_1, a_2, ..., a_{n^2}$. Получить действительную квадратную матрицу порядка n<8, элементами которой являются числа $a_1, a_2, ..., a_{n^2}$, расположенные в ней по схеме:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести полученную матрицу.

Вариант №20

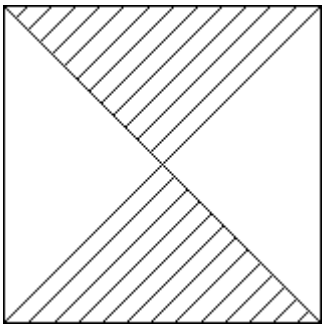
Дана действительная квадратная матрица порядка *n*. Найти и вывести наибольшее из значений элементов, расположенных в заштрихованной части матрицы:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную матрицу.

Вариант №21

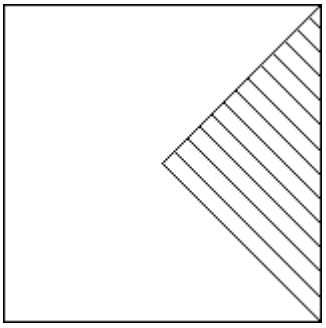
Дана действительная квадратная матрица порядка *n*. Найти и вывести наибольшее из значений элементов, расположенных в заштрихованной части матрицы:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную матрицу.

Вариант №22

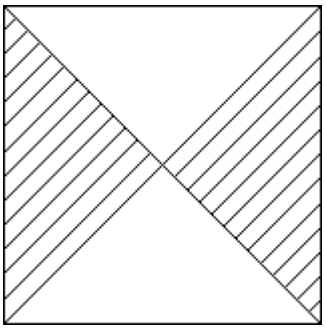
Дана действительная квадратная матрица порядка *n*. Найти и вывести наибольшее из значений элементов, расположенных в заштрихованной части матрицы:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную матрицу.

Вариант №23

Дана действительная квадратная матрица порядка *n*. Найти и вывести наибольшее из значений элементов, расположенных в заштрихованной части матрицы:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную матрицу.

Вариант №24

Дана целочисленная матрица размера *m*x*n*. Найти матрицу, получающуюся из данной перестановкой столбцов – первого с последним, второго с предпоследним и т.д. Матрица размещается в памяти динамически, значения *m* и *n* вводятся по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

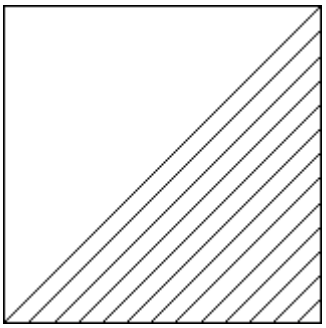
Вариант №25

Дана целочисленная матрица размера размера *m*x*n*. Найти матрицу, получающуюся из данной перестановкой строк – первой с последней, второй

с предпоследней и т.д. Матрица размещается в памяти динамически, значения *m* и *n* вводятся по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

Вариант №26

Дана действительная квадратная матрица порядка *n*. Найти и вывести наибольшее из значений элементов, расположенных в заштрихованной части матрицы:



Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную матрицу.

Вариант №27

Дана действительная квадратная матрица порядка *n*, все элементы которой различны. В этой матрице в каждой строке элементы, стоящие на нечетных местах, заменить суммой, на четных – произведением соответствующей пары. Матрица размещается в памяти динамически, значение *n* вводится по запросу с клавиатуры. В конце работы программы освободить выделенную память. Вывести исходную и результирующую матрицы.

ЗАДАНИЕ №2. РАБОТА С БИТАМИ.

Вариант №1

Ввести 8 символов. В символе с наибольшим кодом заменить 3-й бит нулем, а в символе с наименьшим кодом 4-й бит – единицей. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №2

Ввести последовательность из 8 символов. В каждом из символов в их двоичном представлении заменить: для нечетных (по порядку) символов 3-й бит единицей; для четных символов 4-й бит нулем. Вывести исходную

последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №3

Ввести последовательность из 8 символов. В их двоичном представлении заменить: если младший бит 1, заменить его на 0; если младший бит 0, заменить его и 2-й бит единицами. Вывести исходную последовательность и ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №4

Ввести последовательность из 8 символов. Если символ – буква, то заменить в нем 3-й бит нулем, иначе – заменить 2-й бит единицей. Вывести исходную последовательность и ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №5

Ввести последовательность из 8 символов. Если символ – цифра, то заменить в нем 4-й бит единицей, иначе – 2-й бит нулем. Вывести исходную последовательность и ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №6

Ввести последовательность из 8 целых чисел. Если число четное, то заменить его младший байт нулями, если нечетное, то заменить в его младшем байте 3-й и 4-й бит единицами. Вывести исходную последовательность в десятичной и восьмеричной форме; преобразованную последовательность в десятичной и восьмеричной форме.

Вариант №7

Ввести последовательность из 8 целых чисел. В каждом втором числе заменить (i–1)-й бит единицей, где i-номер члена последовательности. Вывести исходную последовательность в десятичной и восьмеричной формах; преобразованную последовательность в десятичной и восьмеричной формах.

Вариант №8

Ввести последовательность из 8 символов. Если символ – русская гласная буква, то заменить в нем младший бит единицей, иначе – заменить 2-

й и 3-й биты нулями. Вывести исходную и преобразованную последовательности в символьной форме и в восьмеричных кодах.

Вариант №9

Ввести последовательность из 8 символов. Если символ – восьмеричная цифра, то заменить в нем бит, номер которого совпадает с этой цифрой, нулем; иначе – заменить младший бит единицей. Вывести исходную и преобразованную последовательности в символьной и восьмеричной формах.

Вариант №10

Ввести последовательность из 8 символов. Если код символа четный, то заменить в нем младший бит единицей, иначе – заменить два младших бита нулями. Вывести исходную и преобразованную последовательности в символьной и восьмеричной формах.

Вариант №11

Ввести последовательность из 8 символов. Если символ есть + – * / % , то заменить в нем четыре младших бита единицами, иначе – заменить 5-й бит нулем. Вывести исходную и преобразованную последовательности в символьной и восьмеричной формах.

Вариант №12

Ввести последовательность из 8 символов. Если символ – большая латинская буква, то заменить в нем 3-й бит нулем, иначе – заменить младший бит единицей. Вывести исходную и преобразованную последовательности в символьной и восьмеричной формах.

Вариант №13

Ввести последовательность из 8 целых чисел. В каждом нечетном числе заменить (i–1)-й бит нулем (i-номер члена последовательности). Вывести исходную последовательность в десятичной и восьмеричной формах; преобразованную последовательность в десятичной и восьмеричной формах.

Вариант №14

Ввести последовательность из 8 символов. Если символ – латинская согласная буква, то заменить в нем младший бит единицей, иначе – заменить 3-й и 5-й биты нулями. Вывести исходную и преобразованную последовательности в символьной форме и в восьмеричных кодах.

Вариант №15

Реализовать алгоритм инвертирования *n*-разрядов целого числа без знака, начинающихся с *p*-ой позиции. Оставшиеся разряды остаются без изменения. Значения переменной, подлежащей преобразованию, а также значения *n* и *p* вводятся с клавиатуры. Результат вывести на экран в восьмеричном виде.

Вариант №16

Реализовать алгоритм, выполняющий зеркальное отображение битов значения целого числа без знака. Значение переменной, подлежащей преобразованию, вводится с клавиатуры. Результат вывести на экран в восьмеричном виде.

Вариант №17

Реализовать алгоритм поиска первой пары несовпадающих разрядов в двух переменных типа unsigned. Значения сравниваемых переменных вводятся с клавиатуры, результат выводится на экран.

Вариант №18

Ввести последовательность из 8 символов. Сравнить 5-й и 6-й биты каждого символа. Если они не равны, то сделать их равными младшему биту, иначе – старшему. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №19

Ввести 8 символов. В символе с наибольшим кодом заменить 5-й бит единицей, а в символе с наименьшим кодом 6-й бит – нулем. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №20

Ввести последовательность из 8 символов. В их двоичном представлении заменить: если старший бит 1, заменить его на 0; если старший бит 0, заменить его и младший бит единицами. Вывести исходную последовательность и ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №21

Реализовать алгоритм зеркального отображения тетрад битов значения целого числа без знака. Значение переменной, подлежащей преобразованию, вводится с клавиатуры. Результат вывести на экран в восьмеричном виде.

Вариант №22

Ввести последовательность из 8 целых чисел. Если код символа нечетный, то заменить в нем старший бит нулем, иначе – заменить два младших бита единицами. Вывести исходную последовательность в десятичной и восьмеричной формах; преобразованную последовательность в десятичной и восьмеричной формах.

Вариант №23

Ввести последовательность из 8 символов. В каждом из символов в их двоичном представлении заменить все четные биты единицами. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №24

Ввести последовательность из 8 символов. Если сумма единиц в представлении символа нечетная, то заменить 2 старших бита нулями, иначе – единицами. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №25

Ввести последовательность из 8 символов. Если сумма трех старших бит в символе равна единице, то заменить их единицами, иначе – нулями. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №26

Ввести последовательность из 8 символов. Сравнить их младший и старший биты. Если они равны, то заменить старший нулем, младший – единицей, иначе заменить старший бит единицей, младший – нулем. Вывести исходную последовательность, ее восьмеричные коды; преобразованную последовательность и ее восьмеричные коды.

Вариант №27

Ввести последовательность из 8 символов. Если символ – цифра, то заменить в нем 3 младших бита единицами, иначе – первый и последний нулями. Вывести исходную и преобразованную последовательности в символьной форме и в восьмеричных кодах.

Вариант №28

Ввести последовательность из 8 символов. Если символ – латинская гласная буква, то заменить в нем 2 младших бита нулем, иначе – 2-й и 4-й единицами. Вывести исходную и преобразованную последовательности в символьной форме и в восьмеричных кодах.

ЗАДАНИЕ №3. УКАЗАТЕЛИ НА ФУНКЦИИ.

Вариант №1

Разработать функцию filter, которая отфильтровывает все цифры из массива символов. Функция filter получает в качестве аргументов указатель на исходный массив символов, его размер, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №2

Разработать функцию fold, которая позволяет суммировать все числа в массиве целочисленных чисел с использованием функции sum. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию sum от двух аргументов. Функция fold возвращает сумму всех элементов массива. Функция sum производит сложение двух аргументов и возвращает их сумму.

Протестировать разработанную функцию fold.

Вариант №3

Разработать функцию map, которая позволяет все отрицательные значения заменить на ноль в массиве целых чисел. Функция map получает в качестве аргументов массив, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива внутри функции map. Функция cut

получает в качестве аргумента указатель на целое число, и, если это число оказывается отрицательным оно заменяется на ноль.

Протестировать разработанную функцию map.

Вариант №4

Разработать функцию filter, которая отфильтровывает все положительные числа из массива целочисленных чисел. Функция filter получает в качестве аргументов указатель на исходный массив, размер массива, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №5

Разработать функцию fold, которая позволяет найти минимальный символ в массиве символов с использованием функции min. Функция fold получает в качестве аргументов указатель на исходный массив символов, его размер, указатель на функцию min от двух аргументов. Функция fold возвращает минимальный символ в массиве. Функция min производит сравнение двух аргументов и возвращает минимальный из них.

Протестировать разработанную функцию fold.

Вариант №6

Разработать функцию map, которая позволяет все положительные значения заменить на ноль в массиве целых чисел. Функция map получает в качестве аргументов массив, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива внутри функции map. Функция cut получает в качестве аргумента указатель на целое число, и, если это число оказывается положительным оно заменяется на ноль.

Протестировать разработанную функцию map.

Вариант №7

Разработать функцию filter, которая отфильтровывает все четные числа из массива целочисленных чисел. Функция filter получает в качестве аргументов указатель на исходный массив, размер массива, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №8

Разработать функцию fold, которая позволяет найти произведение всех значений в массиве целочисленных чисел с использованием функции mul. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию mul от двух аргументов. Функция fold возвращает значение произведения всех элемент массива. Функция mul производит произведение двух аргументов и возвращает результат произведения.

Протестировать разработанную функцию fold.

Вариант №9

Разработать функцию map, которая позволяет все символы верхнего регистра заменить на символы нижнего регистра в массиве символов. Функция map получает в качестве аргументов массив символов, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива символов внутри функции map. Функция cut получает в качестве аргумента указатель на символ, и, если этот символ оказывается буквой верхнего регистра, то он заменяется на ту же букву нижнего регистра.

Протестировать разработанную функцию map.

Вариант №10

Разработать функцию filter, которая отфильтровывает все нечетные числа из массива целочисленных чисел. Функция filter получает в качестве аргументов указатель на исходный массив, размер массива, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №11

Разработать функцию fold, которая позволяет найти минимальное значение в массиве целочисленных чисел с использованием функции min. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию min от двух аргументов. Функция fold возвращает минимальный элемент массива. Функция min производит сравнение двух аргументов и возвращает минимальный из них.

Протестировать разработанную функцию fold.

Вариант №12

Разработать функцию map, которая позволяет все нечетные значения заменить на ноль в массиве целых чисел. Функция map получает в качестве аргументов массив, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива внутри функции map. Функция cut получает в качестве аргумента указатель на целое число, и, если это число оказывается нечетным оно заменяется на ноль.

Протестировать разработанную функцию map.

Вариант №13

Разработать функцию filter, которая отфильтровывает все отрицательные числа из массива целочисленных чисел. Функция filter получает в качестве аргументов указатель на исходный массив, размер массива, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №14

Разработать функцию fold, которая позволяет найти максимальное значение в массиве целочисленных чисел с использованием функции max. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию max от двух аргументов. Функция fold возвращает максимальный элемент массива. Функция max производит сравнение двух аргументов и возвращает максимальный из них.

Протестировать разработанную функцию fold.

Вариант №15

Разработать функцию map, которая позволяет все четные значения заменить на ноль в массиве целых чисел. Функция map получает в качестве аргументов массив, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива внутри функции map. Функция cut получает в качестве аргумента указатель на целое число, и, если это число оказывается четным оно заменяется на ноль.

Протестировать разработанную функцию map.

Вариант №16

Разработать функцию filter, которая отфильтровывает все буквы верхнего регистра из массива символов. Функция filter получает в качестве аргументов указатель на исходный массив символов, его размер, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №17

Разработать функцию fold, которая позволяет найти сумму всех разностей по модулю соседних элементов в массиве целочисленных чисел с использованием функции sum. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию sum от двух аргументов. Функция fold возвращает сумму всех разностей по модулю соседних элементов в массиве целочисленных чисел. Функция sum производит расчет разности двух аргументов по модулю и возвращает результат расчёта.

Протестировать разработанную функцию fold.

Вариант №18

Разработать функцию map, которая позволяет все символы нижнего регистра заменить на символы верхнего регистра в массиве символов. Функция map получает в качестве аргументов массив символов, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива символов внутри функции map. Функция cut получает в качестве аргумента указатель на символ, и, если этот символ оказывается буквой нижнего регистра, то он заменяется на ту же букву верхнего регистра.

Протестировать разработанную функцию map.

Вариант №19

Разработать функцию filter, которая отфильтровывает все буквы нижнего регистра из массива символов. Функция filter получает в качестве аргументов указатель на исходный массив символов, его размер, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №20

Разработать функцию fold, которая позволяет найти сумму всех произведений соседних элементов в массиве целочисленных чисел с использованием функции sum. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию sum от двух аргументов. Функция fold возвращает сумму всех произведений соседних элементов в массиве целочисленных чисел. Функция sum производит расчет произведения двух аргументов и возвращает результат расчёта.

Протестировать разработанную функцию fold.

Вариант №21

Разработать функцию map, которая позволяет все отрицательные значения заменить на те же положительные значения в массиве целых чисел. Функция map получает в качестве аргументов массив, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива внутри функции map. Функция cut получает в качестве аргумента указатель на целое число, и, если это число оказывается отрицательным оно заменяется на тоже положительное число.

Протестировать разработанную функцию map.

Вариант №22

Разработать функцию filter, которая отфильтровывает все нулевые значения из массива целочисленных чисел. Функция filter получает в качестве аргументов указатель на исходный массив, размер массива, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №23

Разработать функцию fold, которая позволяет найти сумму всех сложений соседних элементов в массиве целочисленных чисел с использованием функции sum. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию sum от двух аргументов. Функция fold возвращает сумму всех сложений соседних элементов в массиве целочисленных чисел. Функция sum производит расчет суммы двух аргументов и возвращает результат расчёта.

Протестировать разработанную функцию fold.

Вариант №24

Разработать функцию map, которая позволяет все положительные значения заменить на те же отрицательные значения в массиве целых чисел. Функция map получает в качестве аргументов массив, его размер, указатель на функцию cut. Функция cut применяется ко всем элементам массива внутри функции map. Функция cut получает в качестве аргумента указатель на целое число, и, если это число оказывается положительным оно заменяется на тоже отрицательное число.

Протестировать разработанную функцию map.

Вариант №25

Разработать функцию filter, которая отфильтровывает все цифры из массива символов. Функция filter получает в качестве аргументов указатель на исходный массив символов, его размер, предикат, указатель на отфильтрованный массив и возвращает размер нового (отфильтрованного) массива, оставляя в нём только те элементы, для которых переданный предикат возвращает логическую истину (предикат – функция, которая возвращает истину или ложь).

Протестировать разработанную функцию filter.

Вариант №26

Разработать функцию fold, которая позволяет суммировать все числа в массиве целочисленных чисел с использованием функции sum. Функция fold получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию sum от двух аргументов. Функция fold возвращает сумму всех элементов массива. Функция sum производит сложение двух аргументов и возвращает их сумму.

Протестировать разработанную функцию fold.