Задача 1. Сортировка по возрастанию

Написать только одну функцию, которая сортирует массив по возрастанию. Необходимо реализовать только одну функцию, всю программу составлять не надо. Строго согласно прототипу. Имя функции и все аргументы должны быть: void sort\_array(int size, int a[]) Всю программу загружать не надо, только одну эту функцию. Можно просто закомментировать текст всей программы, кроме данной функции.

Данные на входе: Функция принимает на вход, первый аргумент - размер массива, второй аргумент - адрес нулевого элемента.

Данные на выходе: Функция ничего не возвращает. Производит сортировку переданного ей массива по возрастанию.

void sort\_array(int size, int a[])

{

int tmp;

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size - 1; j++)

{

tmp = a[j];

if (tmp > a[j + 1])

{

a[j] = a[j + 1];

a[j + 1] = tmp;

}

}

}

}

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Задача 2. Четные в начало

Написать только одну функцию, которая ставит в начало массива все четные элементы, а в конец – все нечетные. Не нарушайте порядок следования чисел между собой. Строго согласно прототипу: void sort\_even\_odd(int n, int a[])

Данные на входе: Функция принимает на вход целые числа

Данные на выходе: Отсортированный исходный массив

void sort\_even\_odd(int size, int a[])

{

int tmp;

for (int i = 0; i < size / 2; i++)

{

for (int j = 0; j < size - 1; j++)

{

tmp = a[j];

if (tmp % 2 != 0 && a[j + 1] % 2 == 0)

{

a[j] = a[j + 1];

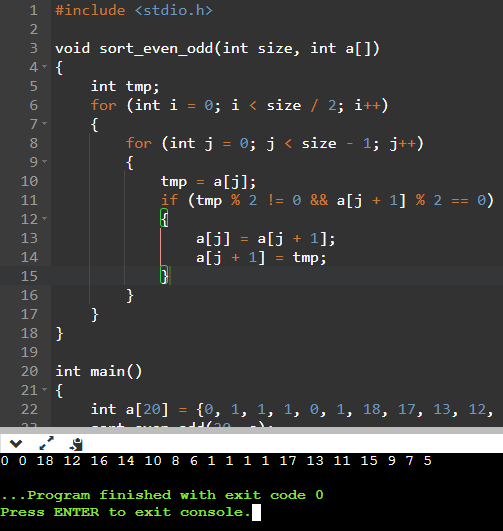
a[j + 1] = tmp;

}

}

}

}



Задача 3. Максимум в массиве

Написать только одну функцию, которая находит максимальный элемент в массиве. Всю программу загружать не надо. Прототип функции: int find\_max\_array(int size, int a[])

Данные на входе: Массив состоящий из целых чисел. Первый аргумент, размер массива, второй аргумент адрес нулевого элемента.

Данные на выходе: Одно целое число

int find\_max\_array(int size, int a[])

{

int max = 0;

for (int i = 0; i < size; i++, size--)

{

if (a[i] < a[size])

{

if (max < a[size])

max = a[size];

}

else

{

if (max < a[i])

max = a[i];

}

}

return max;

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Задача 4. Два одинаковых

Написать только одну логическую функцию, которая определяет, верно ли, что среди элементов массива есть два одинаковых. Если ответ «да», функция возвращает 1; если ответ «нет», то 0. Строго согласно прототипу: int is\_two\_same(int size, int a[]);

Данные на входе: Массив из целых чисел

Данные на выходе: Функция возвращает 1 или 0

int is\_two\_same(int size, int a[])

{

int flag = 0;

for (int i = 0; i < size; i++)

{

for (int j = i; j < size - 1; j++)

{

if (a[i] == a[j + 1])

{

flag = 1;

break;

}

}

}

return flag;

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Задача 5. Поменять местами

Составить функцию, которая меняет в массиве минимальный и максимальный элемент местами. Прототип функции void change\_max\_min(int size, int a[])

Данные на входе: Функция принимает на вход размер массива и массив чисел типа int

Данные на выходе: Функция не возвращает значения, измененный массив сохраняется на месте исходного.

void change\_max\_min(int size, int a[])

{

int flag\_min, flag\_max, tmp\_min, tmp\_max;

tmp\_max = a[0];

tmp\_min = a[0];

for (int i = 0; i < size; i++)

{

if (tmp\_min > a[i])

{

flag\_min = i;

tmp\_min = a[i];

}

if (tmp\_max < a[i])

{

flag\_max = i;

tmp\_max = a[i];

}

}

tmp\_min = a[flag\_min];

a[flag\_min] = a[flag\_max];

a[flag\_max] = tmp\_min;

}

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Задача 6. Больше среднего

Определить количество положительных элементов квадратной матрицы, превышающих по величине среднее арифметическое всех элементов главной диагонали. Реализовать функцию среднее арифметическое главной диагонали.

Данные на входе: 5 строк по 5 целых чисел через пробел

Данные на выходе: Одно целое число

#include <stdio.h>

int sum\_diagonal(int SIZE, int a[SIZE][SIZE])

{

int sum = 0;

for (int i = 0; i < SIZE; i++)

sum = sum + a[i][i];

return sum;

}

void set\_mass(int SIZE, int a[SIZE][SIZE])

{

for (int i = 0; i < SIZE; i++)

{

for (int j = 0; j < SIZE; j++)

scanf("%d", &a[i][j]);

}

}

int main()

{

int SIZE = 5;

int a[SIZE][SIZE];

set\_mass(SIZE, a);

int average = (sum\_diagonal(SIZE, a)) / SIZE;

int count\_max = 0;

for (int i = 0; i < SIZE; i++)

{

for (int j = 0; j < SIZE; j++)

{

if (a[i][j] > average)

count\_max++;

}

}

printf("%d", count\_max);

return 0;

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Задача 7\*. Клетки

Известно, что шахматная доска имеет размерность 8х8 и состоит из клеток 2х цветов, например, черного и белого (см. рисунок). Каждая клетка имеет координату, состоящую из буквы и цифры. Горизонтальное расположение клетки определяется буквой от A до H, а вертикальное – цифрой от 1 до 8. Заметим, что клетка с координатой А1 имеет черный цвет. Требуется по заданной координате определить цвет клетки.

Данные на входе: В единственной строке входного файла записана координата клетки на шахматной доске: всего два символа – буква и цифра (без пробелов).

Данные на выходе: В выходной файл нужно вывести «WHITE», если указанная клетка имеет белый цвет и «BLACK», если она черная.