Константа

public class ConstAlgorithm

{

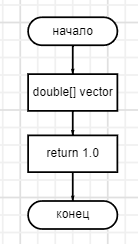
public static double ConstantFunction(double[] vector)

{

return 1.0;

}

}



Сумма

public class SumAlgorithm

{

public static double Sum(double[] vector)

{

double sum = 0;

foreach (var value in vector)

{

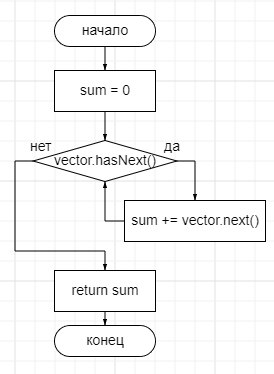
sum += value;

}

return sum;

}

}



Произведение

public class ProductAlgorithm

{

public static double Product(double[] vector)

{

double product = 1;

foreach (var value in vector)

{

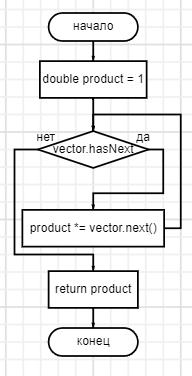
product \*= value;

}

return product;

}

}



Наивное

public class PolynomialNaiveAlgorithm

{

public static double CalculatePolynomialNaive(double[] coefficients, double x)

{

double result = 0;

for (int i = 0; i < coefficients.Length; i++)

{

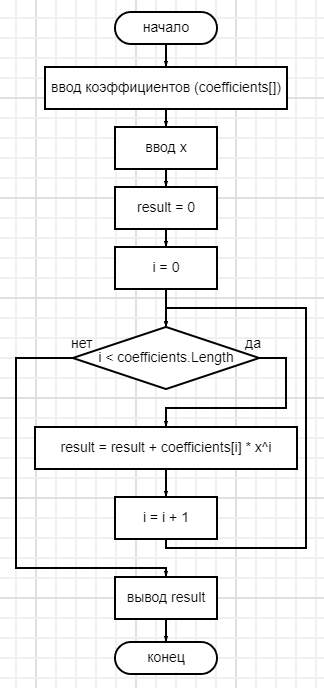
result += coefficients[i] \* Math.Pow(x, i);

}

return result;

}

}



Горнер

public class PolynomialHornerAlgorithm

{

public static double CalculatePolynomialHorner(double[] coefficients, double x)

{

double result = 0;

for (int i = coefficients.Length - 1; i >= 0; i--)

{

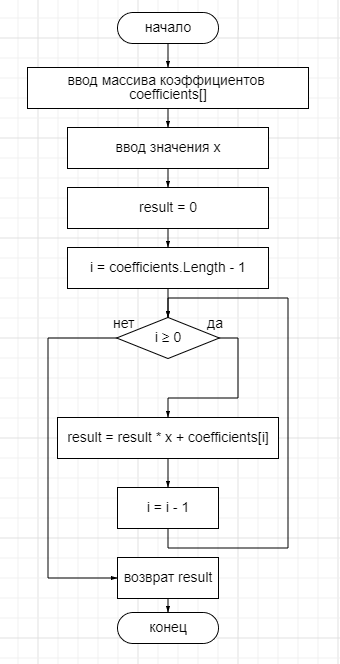
result = result \* x + coefficients[i];

}

return result;

}

}



Бабл сорт

public class BubbleSortAlgorithm

{

public static void Sort(double[] array)

{

int n = array.Length;

for (int i = 0; i < n - 1; i++)

{

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

{

if (array[j] > array[j + 1])

{

double temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

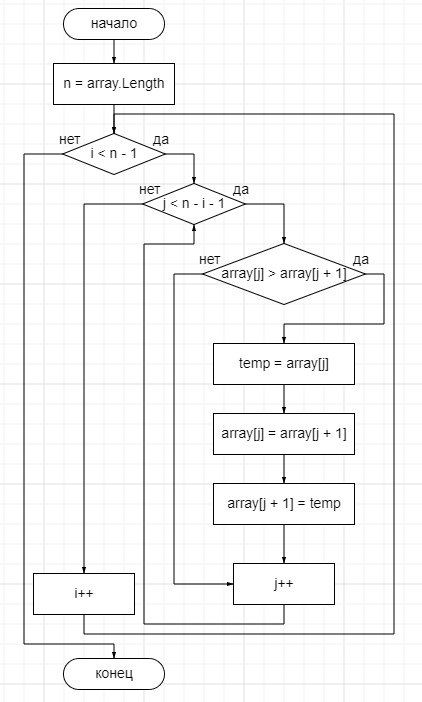
}

}

}

}

}



Квик сорт

public class QuickSortAlgorithm

{

public static void Sort(double[] array, int left, int right)

{

int i = left, j = right;

double pivot = array[(left + right) / 2];

while (i <= j)

{

while (array[i] < pivot) i++;

while (array[j] > pivot) j--;

if (i <= j)

{

double temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

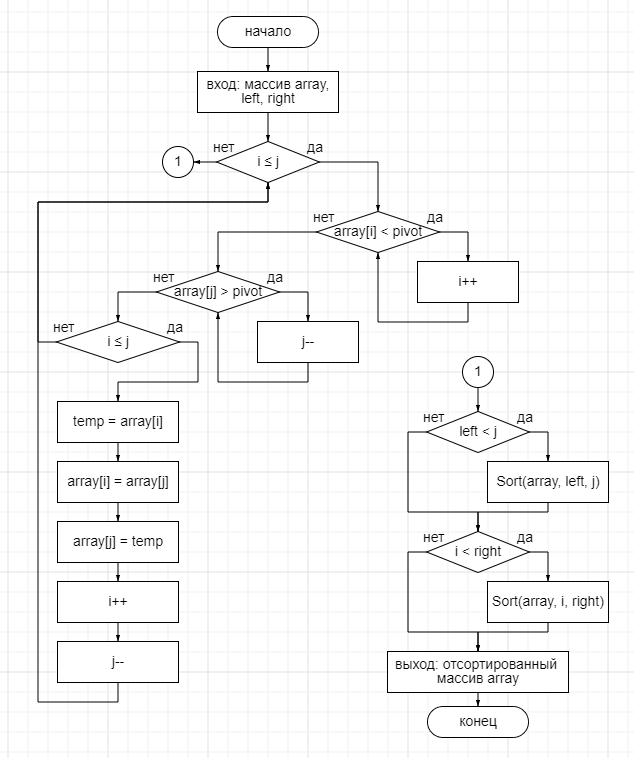
}

if (left < j) Sort(array, left, j);

if (i < right) Sort(array, i, right);

}

}



Вставка

public class InsertionSortAlgorithm

{

public static void Sort(double[] array)

{

int n = array.Length;

for (int i = 1; i < n; i++)

{

double key = array[i];

int j = i - 1;

while (j >= 0 && array[j] > key)

{

array[j + 1] = array[j];

j--;

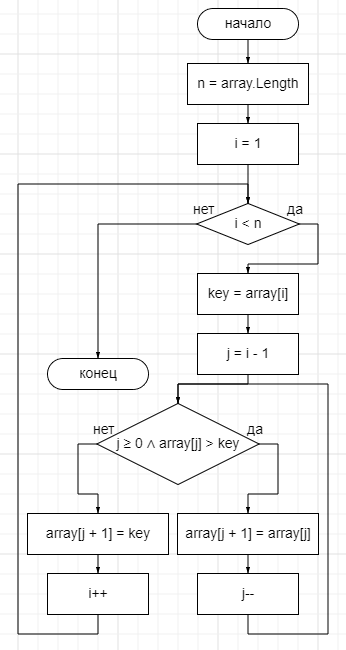
}

array[j + 1] = key;

}

}

}



Перемножение матриц

public static double[,] MatrixMultiplication(double[,] A, double[,] B)

{

int aRows = A.GetLength(0);

int aCols = A.GetLength(1);

int bCols = B.GetLength(1);

if (aCols != B.GetLength(0))

throw new ArgumentException("Размеры матриц не совпадают для умножения.");

double[,] result = new double[aRows, bCols];

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

{

for (int k = 0; k < aCols; k++)

{

double temp = A[i, k];

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

{

result[i, j] += temp \* B[k, j];

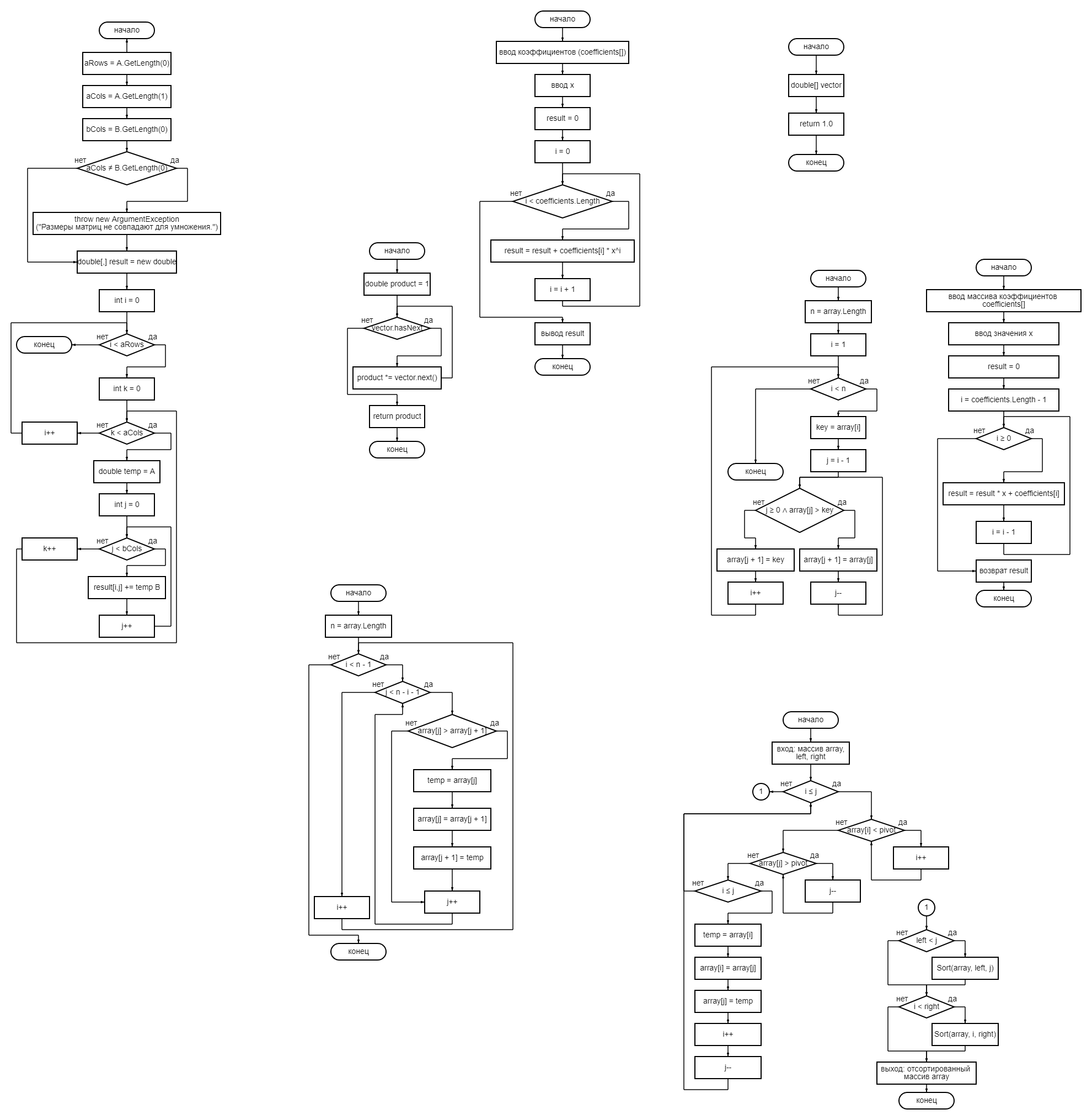
}

}

}

return result;

}



Генерация матриц

public static double[,] GenerateRandomMatrix(int rows, int cols)

{

double[,] matrix = new double[rows, cols];

Random rand = new Random();

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

{

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

{

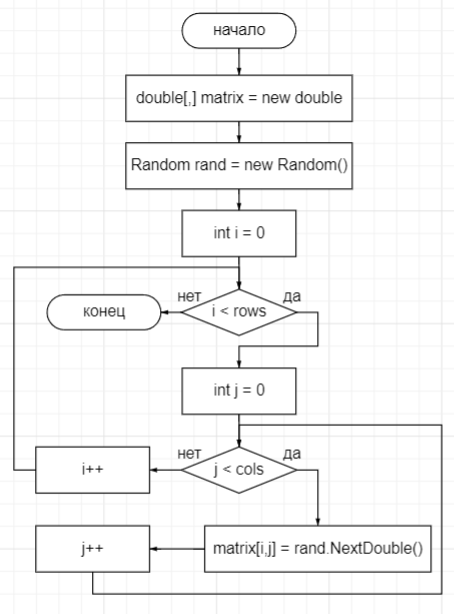
matrix[i, j] = rand.NextDouble() \* 10;

}

}

return matrix;

}



Генератор векторов

public static double[] GenerateRandomVector(int size)

{

double[] vector = new double[size];

Random rand = new Random();

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

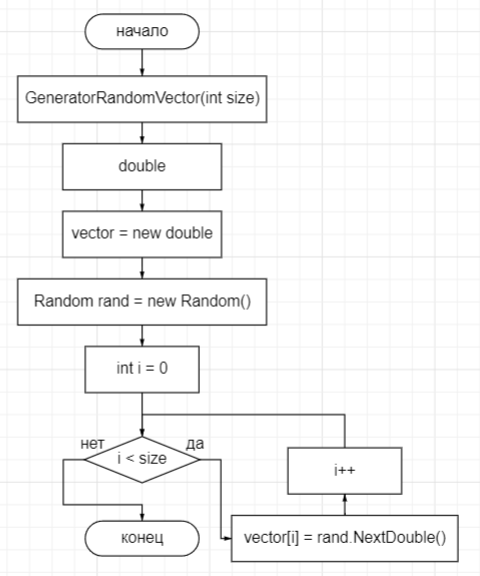
{

vector[i] = rand.NextDouble() \* 100;

}

return vector;

}



Шелл

public class ShellSort

{

public static void Sort(double[] arr)

{

int n = arr.Length;

// Начинаем с большого шага, уменьшая его постепенно

for (int gap = n / 2; gap > 0; gap /= 2)

{

// Выполняем сортировку вставками для каждого подмассива с текущим шагом gap

for (int i = gap; i < n; i++)

{

double temp = arr[i];

int j;

// Сдвигаем элементы подмассива arr[i - gap], arr[i - 2 \* gap], ..., которые больше arr[i]

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)

{

arr[j] = arr[j - gap];

}

// Вставляем arr[i] на правильную позицию

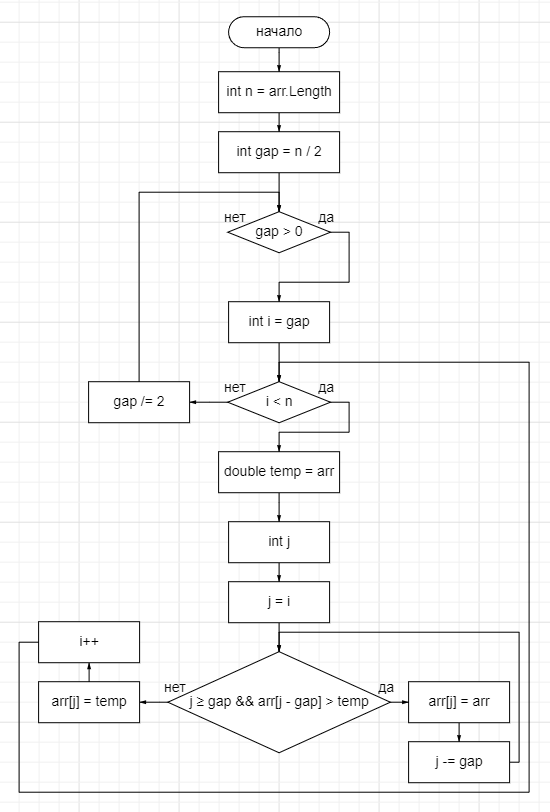
arr[j] = temp;

}

}

}

}



Коктейль

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1\_alg.Utilities

{

internal class CoctailSort

{

private static void Swap(ref double e1, ref double e2)

{

var temp = e1;

e1 = e2;

e2 = temp;

}

//сортировка перемешиванием

public static void ShakerSort(double[] array)

{

for (var i = 0; i < array.Length / 2; i++)

{

var swapFlag = false;

//проход слева направо

for (var j = i; j < array.Length - i - 1; j++)

{

if (array[j] > array[j + 1])

{

Swap(ref array[j], ref array[j + 1]);

swapFlag = true;

}

}

//проход справа налево

for (var j = array.Length - 2 - i; j > i; j--)

{

if (array[j - 1] > array[j])

{

Swap(ref array[j - 1], ref array[j]);

swapFlag = true;

}

}

//если обменов не было выходим

if (!swapFlag)

{

break;

}

}

}

}

}

