МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

Отчет к практической работе № 6

Тема: **«Алгоритмы сортировки»**

Дисциплина: **«Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
|  |  | **Выполнил:** |
|  |  | Студент группы ПИЖ-б-о-23-1, направление подготовки: 09.03.04 «Программная инженерия»  Панчешный Александр Алексеевич  **Проверил:**  Доцент ДЦРСИЭ  Николаев Евгений Иванович |

Ставрополь 2025

**Лабораторная работа 6.** **Алгоритмы сортировки**

**Цель работы:** исследовать основные методы сортировки.

**Ход работы**

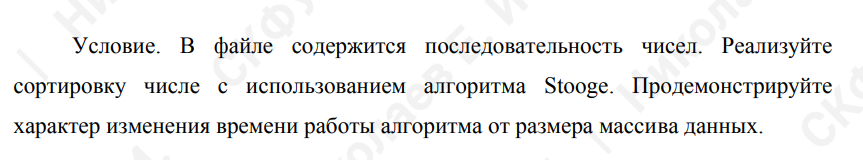
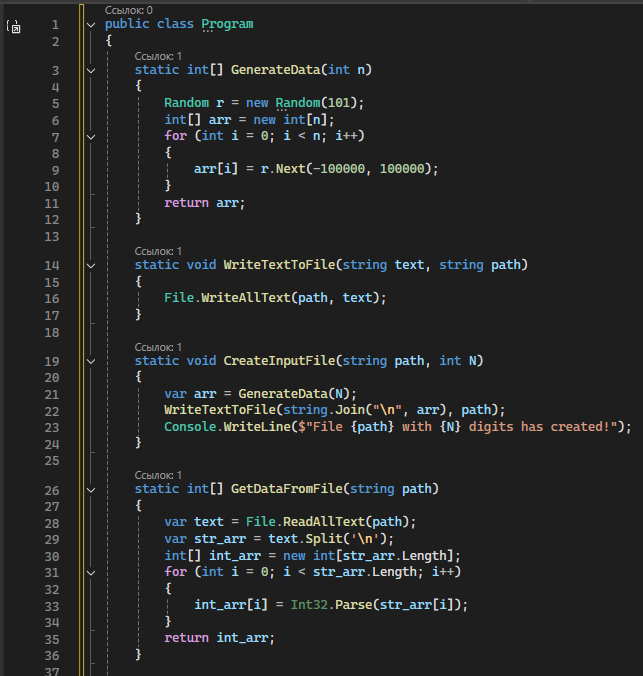


Рисунок 1. Учебная заадча



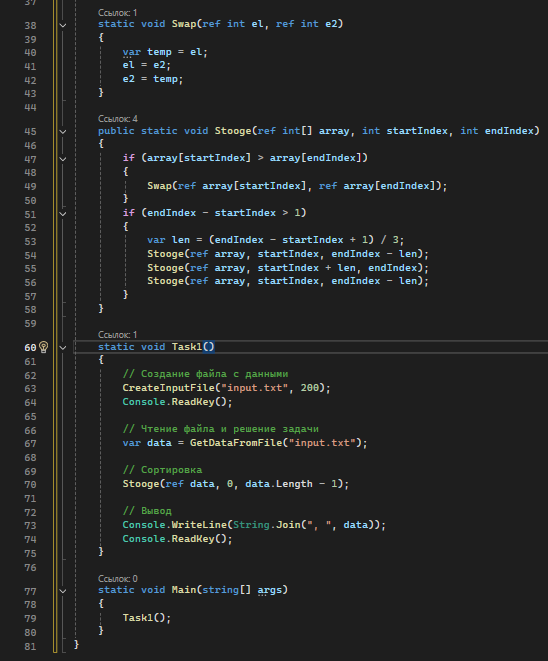


Рисунок 2. Решение учебной задачи

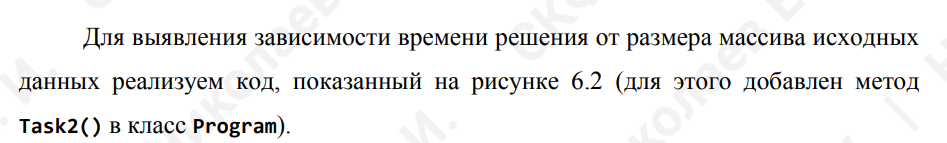


Рисунок 3. Добавление функции Task2()

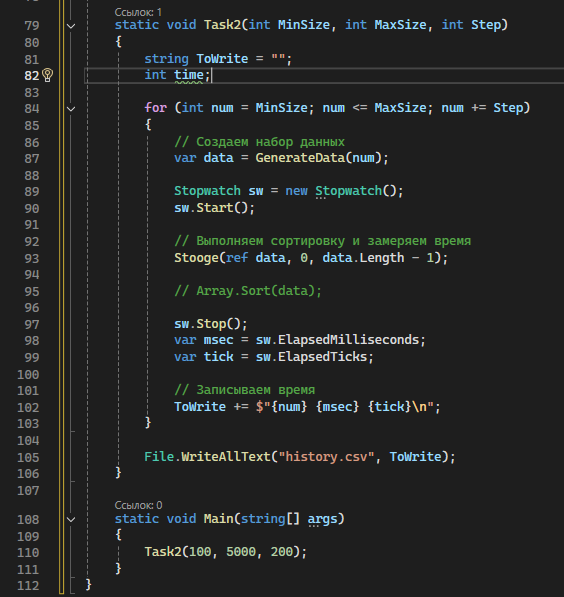
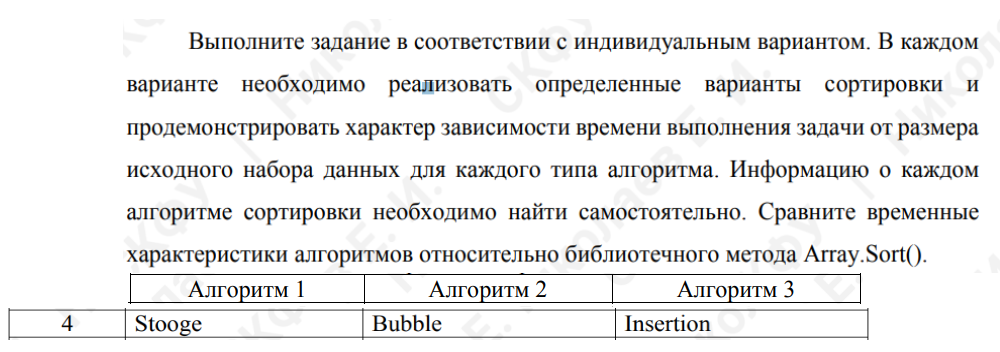


Рисунок 4. Исходный код функции Task2()

**I. Задание и исходный код индивидуального задания**



using System;

using System.Diagnostics;

using System.IO;

public class Program

{

// Алгоритм Stooge Sort

public static void StoogeSort(int[] arr, int i, int j)

{

if (arr[i] > arr[j])

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

if (j - i > 1)

{

int t = (j - i + 1) / 3;

StoogeSort(arr, i, j - t);

StoogeSort(arr, i + t, j);

StoogeSort(arr, i, j - t);

}

}

// Алгоритм Bubble Sort

public static void BubbleSort(int[] arr)

{

int n = arr.Length;

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

{

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

{

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

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

// Алгоритм Insertion Sort

public static void InsertionSort(int[] arr)

{

int n = arr.Length;

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

{

int key = arr[i];

int j = i - 1;

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

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

// Генерация случайного массива

public static int[] GenerateRandomArray(int size)

{

Random random = new Random();

int[] arr = new int[size];

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

{

arr[i] = random.Next(0, 1000);

}

return arr;

}

public static void Main(string[] args)

{

int[] sizes = { 10, 100, 500, 1000, 2000, 5000 };

string filePath = "sorting\_results.csv";

using (StreamWriter writer = new StreamWriter(filePath))

{

writer.WriteLine("Size,Stooge,Bubble,Insertion,ArraySort");

foreach (int size in sizes)

{

int[] arr1 = GenerateRandomArray(size);

int[] arr2 = (int[])arr1.Clone();

int[] arr3 = (int[])arr1.Clone();

int[] arr4 = (int[])arr1.Clone();

// Stooge Sort

Stopwatch sw1 = Stopwatch.StartNew();

StoogeSort(arr1, 0, arr1.Length - 1);

sw1.Stop();

// Bubble Sort

Stopwatch sw2 = Stopwatch.StartNew();

BubbleSort(arr2);

sw2.Stop();

// Insertion Sort

Stopwatch sw3 = Stopwatch.StartNew();

InsertionSort(arr3);

sw3.Stop();

// Array.Sort()

Stopwatch sw4 = Stopwatch.StartNew();

Array.Sort(arr4);

sw4.Stop();

writer.WriteLine($"{size},{sw1.ElapsedMilliseconds},{sw2.ElapsedMilliseconds},{sw3.ElapsedMilliseconds},{sw4.ElapsedMilliseconds}");

Console.WriteLine($"Size: {size} - Stooge: {sw1.ElapsedMilliseconds}ms, Bubble: {sw2.ElapsedMilliseconds}ms, Insertion: {sw3.ElapsedMilliseconds}ms, Array.Sort: {sw4.ElapsedMilliseconds}ms");

}

}

Console.WriteLine($"Результаты записаны в файл {filePath}");

}

}

**II. Контрольные вопросы**

**1. Приведите пример объявления массива. Приведите пример инициализации элементов массива.**

**Пример объявления массива:**

C#

int[] numbers; // Объявление одномерного массива целых чисел

string[] names; // Объявление одномерного массива строк

double[,] matrix; // Объявление двумерного массива чисел с плавающей точкой

**Пример инициализации элементов массива:**

// Инициализация одномерного массива целых чисел

numbers = new int[5]; // Создание массива из 5 элементов

numbers[0] = 10;

numbers[1] = 20;

numbers[2] = 30;

numbers[3] = 40;

numbers[4] = 50;

// Инициализация одномерного массива строк

names = new string[] { "Иван", "Мария", "Петр" };

// Инициализация двумерного массива чисел с плавающей точкой

matrix = new double[,] { { 1.1, 2.2, 3.3 }, { 4.4, 5.5, 6.6 } };

**2. Что будет выведено в консоль в результате работы приложения?**

static void Main(string[] args)

{

int[,] dots = { { 0, 0, 0 }, { 1, 1, 1 }, { 2, 2, 2 } };

int zz = 10;

for (int i = 0; i < 3; i++)

{

zz \*= dots[0, 1];

}

Console.Write(zz);

Console.ReadKey();

}

**Ответ:**

1. Создается двумерный массив dots с элементами { { 0, 0, 0 }, { 1, 1, 1 }, { 2, 2, 2 } }.
2. Переменной zz присваивается начальное значение 10.
3. Цикл for выполняется 3 раза (от i = 0 до i = 2).
4. В каждой итерации цикла zz умножается на dots[0, 1], который равен 0.
5. После первой итерации zz становится 10 \* 0 = 0.
6. После второй и третьей итераций zz остается равным 0.
7. В консоль выводится значение zz.

**3. Что будет выведено в консоль в результате работы приложения?**

static void Main(string[] args)

{

int[,] dots = {

{ 10, 20, 30 },

{ 21, 17, 13 },

{ 23, 82, 82 }

};

Console.Write(dots.Length);

Console.ReadKey();

}

**Ответ:**

1. Создается двумерный массив dots размером 3x3.
2. dots.Length возвращает общее количество элементов в массиве.
3. В массиве dots 3 строки и 3 столбца, поэтому общее количество элементов равно 3 \* 3 = 9.
4. В консоль выводится значение dots.Length.

**4. Что будет выведено в консоль в результате работы приложения?**

static void Main(string[] args)

{

int[,] dots = { { 10, 20, 30 }, { 40, 50, 60 }, { 70, 80, 90 } };

int[,] res = new int[3, 3];

int i = 0;

while (i < res.Length)

{

res[i / 3, i % 3] = dots[i % 3, i / 3];

i++;

}

Console.Write(res[1, 2]);

Console.ReadKey();

}

**Ответ:**

1. Создаются два двумерных массива dots и res размером 3x3.
2. Цикл while выполняется res.Length раз, то есть 9 раз.
3. В каждой итерации цикла элемент res[i / 3, i % 3] заполняется элементом dots[i % 3, i / 3].
4. В результате массив res будет заполнен следующим образом:
5. {{10, 40, 70},
6. {20, 50, 80},
7. {30, 60, 90}}
8. В консоль выводится значение res[1, 2], которое равно 80.

**5. Приведите фрагмент кода, демонстрирующий использование генератора случайных чисел.**

**Ответ:**

Random random = new Random(); // Создание объекта Random

int randomNumber = random.Next(1, 101); // Генерация случайного числа от 1 до 100

Console.WriteLine(randomNumber);

**6. В чем преимущества и недостатки непосредственного задания значений элементов массива в {} при его объявлении? В чем недостатки такого подхода?**

**Преимущества:**

* **Простота и наглядность:** Удобно задавать небольшие массивы с известными значениями.
* **Инициализация при объявлении:** Массив сразу создается и заполняется данными.

**Недостатки:**

* **Ограниченность:** Не подходит для больших массивов или массивов, значения которых неизвестны заранее.
* **Негибкость:** Изменение значений требует редактирования кода.
* **Не подходит для динамических данных:** Нельзя использовать для данных, получаемых во время выполнения программы (например, из файлов или от пользователя).

**7. Как изменить размер объявленного и инициализированного массива (например, добавить дополнительные элементы)?**

В C# массивы имеют фиксированный размер. Для изменения размера массива необходимо создать новый массив большего размера и скопировать в него элементы из старого массива.

int[] myArray = { 1, 2, 3 };

int[] newArray = new int[myArray.Length + 2]; // Создание массива большего размера

Array.Copy(myArray, newArray, myArray.Length); // Копирование элементов

newArray[3] = 4; // Добавление новых элементов

newArray[4] = 5;

myArray = newArray; // Замена старого массива новым

**8. Что будет выведено в консоль в результате работы приложения?**

int[,] dots = { { 10, 20, 30 }, { 40, 50, 60 }, { 70, 80, 90 } };

int[,] res = dots;

int A = 0;

for (int i = 0; i < 3; i++)

A += res[i, 2];

Console.WriteLine(A);

Console.ReadKey();

**Ответ:**

1. Создается двумерный массив dots размером 3x3.
2. Создается двумерный массив res, который ссылается на тот же участок памяти, что и dots.
3. Цикл for выполняется 3 раза.
4. В каждой итерации цикла к A добавляется значение res[i, 2], которое равно 30, 60, 90 соответственно.
5. После выполнения цикла A будет равно 30 + 60 + 90 = 180.
6. В консоль выводится значение A.

**9. Опишите принцип работы алгоритма сортировки, который был реализован в ходе выполнения индивидуального задания.**

**1. Алгоритм Stooge Sort:**

* **Принцип работы:** **Stooge Sort** — это рекурсивный алгоритм сортировки, известный своей низкой эффективностью. Он основан на принципе "разделяй и властвуй", но с тремя рекурсивными вызовами.
* **Шаги:**
  1. Если первый элемент больше последнего, они меняются местами.
  2. Если в массиве больше двух элементов:
     + Сортируются первые 2/3 массива.
     + Сортируются последние 2/3 массива.
     + Снова сортируются первые 2/3 массива.
* **Особенности:**
  1. Очень неэффективен, имеет сложность O(n^(log 3 / log 1.5)) ≈ O(n^2.7).
  2. Используется в основном в образовательных целях для демонстрации рекурсии.

**2. Алгоритм Bubble Sort (Пузырьковая сортировка):**

* **Принцип работы:** **Bubble Sort** — простой алгоритм сортировки, который многократно проходит по массиву, сравнивая соседние элементы и меняя их местами, если они находятся в неправильном порядке.
* **Шаги:**
  1. Проходим по массиву от первого элемента до предпоследнего.
  2. Сравниваем каждый элемент с последующим.
  3. Если текущий элемент больше следующего, меняем их местами.
  4. Повторяем шаги 1-3, пока не пройдем по всему массиву без изменений.
* **Особенности:**
  1. Простой в реализации, но неэффективный для больших массивов.
  2. Имеет сложность O(n^2) в худшем и среднем случаях.
  3. Хорошо подходит для почти отсортированных массивов.

**3. Алгоритм Insertion Sort (Сортировка вставками):**

* **Принцип работы:** **Insertion Sort** — алгоритм сортировки, который строит отсортированный массив, вставляя каждый элемент в правильную позицию в уже отсортированной части массива.
* **Шаги:**
  1. Начинаем со второго элемента массива.
  2. Сравниваем текущий элемент с элементами в отсортированной части массива (слева от него).
  3. Перемещаем элементы, большие текущего, вправо, чтобы освободить место для вставки текущего элемента.
  4. Вставляем текущий элемент в правильную позицию.
  5. Повторяем шаги 1-4 для всех оставшихся элементов.
* **Особенности:**
  1. Простой в реализации и эффективен для небольших массивов или почти отсортированных массивов.
  2. Имеет сложность O(n^2) в худшем и среднем случаях.
  3. Имеет сложность O(n) в лучшем случае (когда массив уже отсортирован).