МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**ЛАБОРАТОРНАЯ РАБОТА № 5**

по дисциплине

**«Объектно-ориентированное программирование»**

на тему:

**«Классы, объекты, наследование в C#»**

*Вариант № 15*

Выполнил:

Студент группы

КТбо2-8

Лахматов Д. И.

Проверил:

Тарасов С. А.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Таганрог 2020

# **1 Цель работы**

Ознакомление с основами объектно-ориентированного программирования на С#.

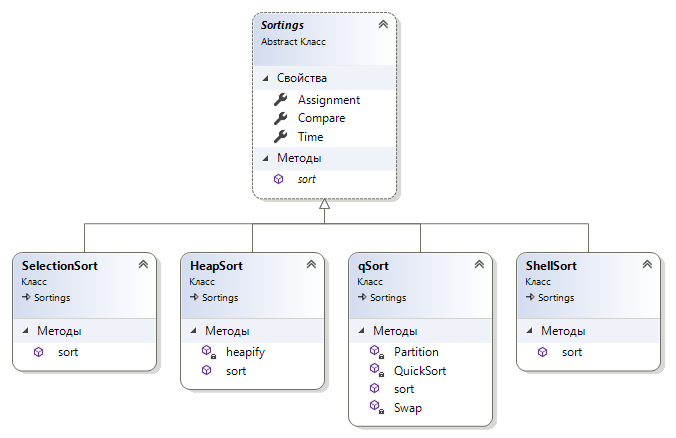
# **2 Задание, вариант № 8**

# Создать класс Sorting (сортировка), и на его основе классы Choice (сортировка выбором) и Quick (быстрая сортировка). Размер сортируемых массивов задается при их создании, а элементы считываются из файла.

# **3 Ход работы**

**3.1 Спецификации классов**

Диаграмма используемых классов приведена на Рисунке 1.



*Рисунок 1 — Диаграмма классов*

**3 Листинг**

//Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace oop\_lab5\_6\_part\_II

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

//Sorting.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Sort

{

abstract class Sortings

{

public double Time { get; protected set; } = 0;

public int Compare { get; protected set; } = 0; // сравнивание

public int Assignment { get; protected set; } = 0; // присваивание

abstract public int[] sort(int[] arr);

}

class qSort : Sortings

{

void Swap(ref int x, ref int y)

{

Assignment -= -2;

var t = x;

x = y;

y = t;

}

int Partition(int[] array, int minIndex, int maxIndex)

{

var pivot = minIndex - 1;

for (var i = minIndex; i < maxIndex; i++)

{

Compare++;

if (array[i] < array[maxIndex])

{

pivot++;

Swap(ref array[pivot], ref array[i]);

}

}

pivot++;

Swap(ref array[pivot], ref array[maxIndex]);

return pivot;

}

int[] QuickSort(int[] array, int minIndex, int maxIndex)

{

if (minIndex >= maxIndex)

{

return array;

}

var pivotIndex = Partition(array, minIndex, maxIndex);

QuickSort(array, minIndex, pivotIndex - 1);

QuickSort(array, pivotIndex + 1, maxIndex);

return array;

}

override public int[] sort(int[] arr)

{

arr = QuickSort(arr, 0, arr.Length - 1);

return arr;

}

}

class SelectionSort : Sortings

{

override public int[] sort(int[] arr)

{

for (int i = 0; i < arr.Length - 1; i++)

{

int min = i;

for (int j = i + 1; j < arr.Length; j++)

{

Compare++;

if (arr[j] < arr[min])

{

min = j;

}

}

int temp = arr[min];

arr[min] = arr[i];

arr[i] = temp;

Assignment -= -2;

}

return arr;

}

}

class HeapSort : Sortings

{

void heapify(int[] arr, int n, int i)

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

Compare++;

if (l < n && arr[l] > arr[largest])

{

largest = l;

}

Compare++;

if (r < n && arr[r] > arr[largest])

{

largest = r;

}

if (largest != i)

{

int swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

Assignment += 2;

heapify(arr, n, largest);

}

}

override public int[] sort(int[] arr)

{

int n = arr.Length;

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--)

{

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

Assignment -= -2;

heapify(arr, i, 0);

}

return arr;

}

}

class ShellSort : Sortings

{

override public int[] sort(int[] arr)

{

int j;

int step = arr.Length / 2;

while (step > 0)

{

for (int i = 0; i < (arr.Length - step); i++)

{

j = i;

Compare++;

while ((j >= 0) && (arr[j] > arr[j + step]))

{

int tmp = arr[j];

arr[j] = arr[j + step];

arr[j + step] = tmp;

j -= step;

Assignment -= -2;

}

}

step = step / 2;

}

return arr;

}

}

}