Федеральное государственное автономное

Образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1**

Алгоритмы поиска значений

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Р. Ю. Царев

подпись, дата инициалы, фамилия

Студент КИ16-16Б, №031620303 \_\_\_\_\_\_\_\_\_ В.В. Кокташев

Номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2018

# Цель работы

# Изучение алгоритмов поиска значения и сравнения их эффективности по времени их выполнения.

# Задание

1. реализовать в программе алгоритм поиска моды;
2. реализовать в программе алгоритм поиска медианы;
3. сравнить эффективность реализованных алгоритмов по времени их выполнения;

# Ход работы

1. Было разработано консольное приложение, реализующее алгоритмы поиска моды и медианы (рисунки 1-3). Также обеспечена устойчивость программы при любых воздействиях, задаваемых пользователем.

Алгоритм поиска медианы:

1. Массив со значениями сортируется;
2. Если количество элементов в массиве нечётное, то выводится средний элемент, иначе возвращается среднее арифметическое значение двух чисел в середине списка.

Алгоритм поиска моды:

1. Массив со значениями сортируется;
2. Значения массива перебираются в цикле. Наиболее повторяющиеся из них записываются в массив. В конец этого массива добавляется количество повторений моды.

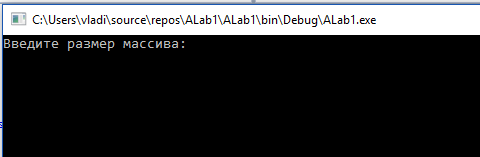


Рисунок 1 - Ввод размера массива

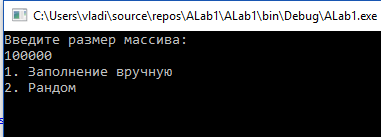


Рисунок 2- Выбор способа заполнения массива

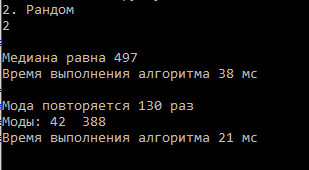


Рисунок 3 – Результат работы программы

1. Проведено сравнение эффективности данных алгоритмов. Для каждого алгоритма в качестве примера использован массив, состоящий из 1000000 элементов. Каждый элемент массива имеет случайное значение (от 1 до 1000). Время выполнения отсчитывается при помощи класса Stopwatch. На рисунке 4 можно проследить разницу во времени выполнения алгоритмов.

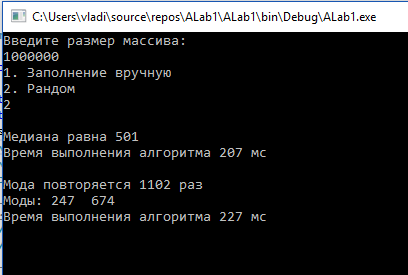


Рисунок 6 - Время выполнения алгоритмов поиска моды и медианы

1. В результате среднее время выполнения алгоритма поиска моды больше времени алгоритма поиска медианы. Это обуславливается более высокой сложностью алгоритма поиска моды.

# Вывод

В ходе практической работы были разработаны и отлажены алгоритмы поиска моды и медианы. По времени выполнения эффективнее оказался алгоритм поиска медианы.

# Исходный код программы

Листинг 1 - файл Program.cs

|  |
| --- |
| **using** System;  **using** System.Diagnostics;    **namespace** ALab1  {  **class** Program  {  **static** **void** Main(**string**[] args)  {  **float**[] mas = new **float**[1];  **float**[] mas2 = new **float**[1];  **int** size; **float**[] moda; **int** menu;  Stopwatch timer = new Stopwatch();  Stopwatch timer2 = new Stopwatch();  Random random = new Random();    **do**  {  Console.WriteLine("Введите размер массива:");  **if** (**int**.TryParse(Console.ReadLine(), **out** size))  **break**;  } **while** (**true**);  Array.Resize(**ref** mas, size);  Array.Resize(**ref** mas2, size);    **do**  {  Console.WriteLine("1. Заполнение вручную**\n**2. Рандом");  **if** (**int**.TryParse(Console.ReadLine(), **out** menu) && menu > 0 && menu < 3)  **break**;  } **while** (**true**);    **if** (menu == 2)  **for** (**int** i = 0; i < size; i++)  mas[i] = random.Next(1, 1000);  **else**  **for** (**int** i = 0; i < size; i++)  **do**  {  Console.Write( "mas[{0}]: ", i);  **if** (**float**.TryParse(Console.ReadLine(), **out** mas[i]))  **break**;  } **while** (**true**);  Array.Copy(mas, mas2, mas.Length);    timer.Start();  Console.WriteLine("**\n**Медиана равна {0}", Task1(mas));  timer.Stop();  Console.WriteLine("Время выполнения алгоритма {0} мс", (timer.ElapsedMilliseconds).ToString());    timer2.Start();  moda = Task2(mas2);  **if** (moda[moda.Length - 1] != 1)  {  Console.Write("**\n**Мода повторяется {0} раз**\n**Моды:", moda[moda.Length - 1]);  **for** (**int** i = 0; i < moda.Length - 1; i++)  Console.Write(" {0} ", moda[i]);  }  **else**  {  Console.Write("**\n**Мода повторяется 1 раз**\n**Моды:");  **for** (**int** i = 0; i < mas2.Length; i++)  Console.Write(" {0} ", mas2[i]);  }  timer2.Stop();  Console.WriteLine("**\n**Время выполнения алгоритма {0} мс", (timer2.ElapsedMilliseconds).ToString());    Console.ReadKey();  }    **public** **static** **float** Task1(**float**[] mas)  {  Array.Sort(mas);  **if** (mas.Length % 2 == 1)  **return** mas[mas.Length / 2];  **else**  **return** ((mas[mas.Length / 2 - 1] + mas[mas.Length / 2])/2);  }    **public** **static** **float**[] Task2(**float**[] mas)  {  Array.Sort(mas);  **float**[] results = new **float**[1];  **int** max = 1; **int** now = 1;    **for**(**int** i = 1; i < mas.Length; i++)  {  **if** (mas[i] == mas[i - 1])  {  now++;  **if** (now == max)  {  Array.Resize(**ref** results, results.Length + 1);  results[results.Length - 1] = mas[i];  }  **if** (now > max)  {  max = now;  Array.Resize(**ref** results, 1);  results[0] = mas[i];  }  }  **else**  now = 1;  }  Array.Resize(**ref** results, results.Length + 1);  results[results.Length - 1] = max;  **return** results;    }  } |