МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО "ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |
|  |  |

**ОТЧЕТ ПО ПРАКТИКЕ ПО ПОЛУЧЕНИЮ ПРОФЕССИОНАЛЬНЫХ УМЕНИЙ И НАВЫКОВ, В ТОМ ЧИСЛЕ ПЕРВИЧНЫХ УМЕНИЙ И НАВЫКОВ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ**

(2022/2023 учебный год)

  Уткин Михаил Максимович

Направление подготовки (специальность) 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки (специализация) «Вычислительные системы машины, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М. А.      

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Научный руководитель к.т.н., доцент Карамышева

*(должность, ученая степень, ученое звание)*

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО "ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ ПРАКТИКИ ПО ПОЛУЧЕНИЮ ПРОФЕССИОНАЛЬНЫХ УМЕНИЙ И НАВЫКОВ, В ТОМ ЧИСЛЕ ПЕРВИЧНЫХ УМЕНИЙ И НАВЫКОВ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ**

(2022/2023 учебный год)

  Уткин Михаил Максимович

Направление подготовки (специальность) 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки (специализация) «Вычислительные системы машины, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М. А.      *(должность, ученая степень, ученое звание, Ф.И.О.)*

Научный руководитель к.т.н., доцент Карамышева Н.С.*(должность, ученая степень, ученое звание*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Планируемая форма работы во время практики | Количество часов | Календарные сроки проведения работы | Подпись  руководителя  практики от вуза |
| 1 | Выбор темы и разработка индивидуального плана проведения работ | 2 | 29.06.2023-  29.06.2023 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 30.07.2023-  02.07.2023 |  |
| 3 | Разработка алгоритма | 43 | 02.07.2023-  06.07.2023 |  |
| 4 | Описание алгоритма и программы | 18 | 06.07.2023-  08.07.2023 |  |
| 5 | Тестирование | 5 | 08.07.2023-  08.07.2023 |  |
| 6 | Получение и анализ результатов | 10 | 08.07.2023-  10.07.2023 |  |
| 7 | Оформление отчёта | 15 | 10.07.2023-  12.07.2023 |  |
|  | **Общий объём часов** | 108 |  |  |

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО "ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ВЫПОЛНЕНИИ ПРАКТИКИ ПО ПОЛУЧЕНИЮ ПРОФЕССИОНАЛЬНЫХ УМЕНИЙ И НАВЫКОВ, В ТОМ ЧИСЛЕ ПЕРВИЧНЫХ УМЕНИЙ И НАВЫКОВ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ**

(2022/2023 учебный год)

  Уткин Михаил Максимович

Направление подготовки (специальность) 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки (специализация) «Вычислительные системы машины, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 29.06.2023 по 12.07.2023

Кафедра «Вычислительная техника»

В процессе выполнения научно-исследовательской работы Уткин М. М. решал следующие задачи: поиск и анализ данных, анализ и добавление в программу работу с файлами, а также производил тестирование программы.

За период выполнения научно-исследовательской работы были освоены основные понятия и технологии языка программирования C++, изучены основные инструменты выполнения сортировки данных. Получены следующие результаты: была создана программа, производящая сортировку данных методом двоичной сортировки. Во время выполнения работы Уткин М. М. показал себя ответственным, добросовестным студентом, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и методам сортировки данных.

За выполнение работы Уткин М. М. заслуживает оценки «Отлично».

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ « »\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**Оглавление**

1 Введение………..………………………………………………….……………3

2 Разработка и описание алгоритма………………………………….………….4

3 Описание программы………………………………………..………………….6

4 Тестирование……..……………………………………..……………………….9

5 Работа с GitHub……..…………..……………………..……………………….10

6 Заключение……….…………………………………………………………….11

Список литературы………………………………………………………………12

Приложение А………………………………………………………………13

Приложение Б………………………………………………………………17

Приложение В………………………………………………………………21

**1 Введение**

При работе с массивами данных нередко возникает задача их сортировки по возрастанию или убыванию, т.е. упорядочивания. Это значит, что элементы того же массива нужно расположить строго по порядку. Например, в случае сортировки по возрастанию предшествующий элемент должен быть меньше последующего (или равен ему).

Существует множество методов сортировки. Одни из них являются более эффективными, другие – проще для понимания. Достаточно непростым для понимания является метод двоичной сортировки.

Целью прохождения практики является осуществление алгоритма двоичной сортировки с использованием консольного интерфейса.

**2 Разработка и описание алгоритма**

Сортировка с помощью двоичного дерева (сортировка двоичным деревом, двоичная сортировка, сортировка с помощью бинарного дерева, англ. tree sort) — универсальный алгоритм сортировки, заключающийся в построении двоичного дерева поиска по ключам массива (списка), с последующей сборкой результирующего массива путём обхода узлов построенного дерева в необходимом порядке следования ключей. Данная сортировка является оптимальной при получении данных путём непосредственного чтения из потока (например, файла, сокета или консоли). Решение задачи сортировки состоит из двух этапов:

1. Построение двоичного дерева;

2. Сборка результирующего массива путём обхода узлов в необходимом порядке следования ключей.

Само бинарное дерево удобно формировать по мере поступления новых данных, например при считывании данных из файла или с консоли. При добавлении нового элемента в дерево происходит его просейка вниз, начиная корня по следующим правилу: Если новый элемент меньше текущего узла, то перенаправляем его в левое поддерево, иначе в правое. Повторяем этот шаг до тех пор, пока не окажемся в пустом листе дерева. Это и будет место нового элемента.

Общая блок-схема программы представлена на рисунке 1.

Блок-схемы функций newNode, storeSorted, insert, treeSort представлены в Приложении А.

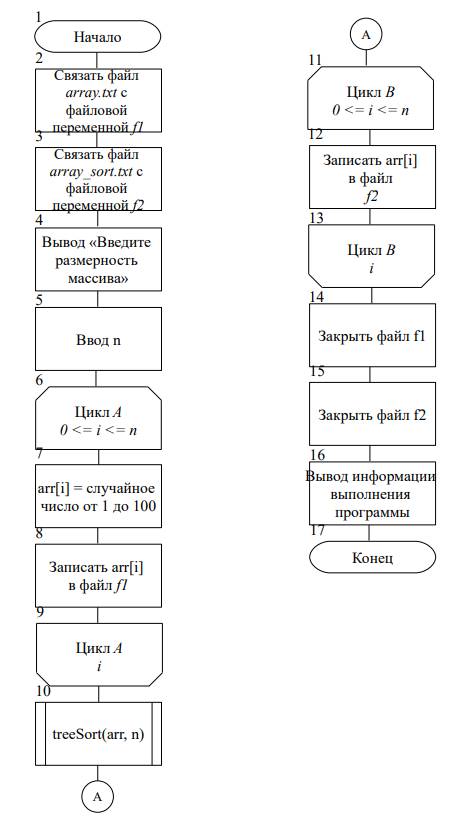


Рисунок 1 - Блок-схема главной программы

**3 Описание программы**

Для написания данной программы использован язык программирования C++. В качестве среды программирования был выбран программный продукт Microsoft Visual Studio 2022. Проект был создан в виде приложения с консольным интерфейсом ConsoleApplication.

Преимущества данной программы – высокая скорость работы, удобное представление для пользователя. Программа работает с целыми положительными числами. Максимальная размерность массива задается в программе и не подлежит изменению в процессе работы приложения. Пользователю необходимо предварительно ввести размер массива в консоль в процессе работы программы,.

Ниже представлено описание, описывающая действие каждой функции.

В программе для двоичной сортировки подключены следующие заголовочные файлы:

iostream – заголовочный файл с классами, функциями и переменными для организации ввода-вывода;

Windows.h – заголовочный файл, необходимый для использования в программе функционала, предоставляемого операционной системой Windows;

fstream – заголовочный файл, предоставляющий функционал для считывания данных из файла и для записи в файл;

ctime – заголовочный файл стандартной библиотеки C++, содержащий типы и функции для работы с датой и временем.

Using namespace std – использование стандартного пространства имен.

Далее идет текст функции создания нового узла ***newNode*** с формальными параметрами node, key. Создается новый узел Node. Далее элементу temp->key присваивается числовое значение item. Затем элементам temp->left и temp->right присваивается значение 0. После этого в главную программу возвращается значение temp.

После этого идёт текст функции сортировки дерева ***storeSorted*** с формальными параметрами root, arr, i. Проверяется условие (root != 0) и, если оно истинно, осуществляется рекурсивный вызов функции storeSorted с фактическими параметрами root->left, arr, i. Затем элементу массива arr[i++] присваивается значение root->key. После этого осуществляется рекурсивный вызов функции storeSorted с фактическими параметрами root->right, arr, i.

Затем идет текст функции вставки ***insert*** с формальными параметрами node, key. Проверяется условие (node == 0) и, если оно истинно, происходит вызов подпрограммы newNode с фактическим параметром key. Проверяется условие (key < node->key) . Если условие истинно, элементу node->left присваивается значение insert(node->left, key). Иначе проверяется условие (key > node->key) и, если оно истинно, элементу node->right присваивается значение insert(node->right, key). После этого в главную программу возвращается указатель node.

Затем идет текст функции сортировки массива ***treeSort*** с формальными параметрами arr, n. Элементу Node\*root присваивается значение 0. Переменной root присваивается значение insert(root, arr[0]). Далее следует цикл А с счетчиком i ( for (int i = 1; i < n; i++) ). В теле цикла А переменной root присваивается значение insert(root, arr[i]). После окончания цикла с счетчиком переменной i присваивается значение 0. После этого осуществляется вызов подпрограммы storeSorted с фактическими параметрами root, arr, i.

Затем идет текст главной программы. Сначала выполняется русификация. Затем открывается файл array.txt и связывается с файловой переменной f1. Затем открывается файл array\_sort.txt и связывается с файловой переменной f2. Далее происходит вывод сообщения на экран «Введите размерность массива». После этого осуществляется ввод переменной n. Затем следует цикл А с счетчиком i ( for (int i = 0; i < n; i++) ). В теле цикла А элементу массива arr[i] присваивается случайное число от 1 до 100. Потом элемент массива arr[i] записывается в файл f1. После этого осуществляется вызов подпрограммы treeSort с фактическим параметрами arr, n. Далее следует цикл В с счетчиком i ( for (int i = 0; i < n; i++) ). В теле цикла В происходит запись в файл f2 элемента массива arr[i]. После окончания цикла В с счетчиком происходит закрытие файлов f1 и f2. После этого происходит вывод на экран информации о работе программы.

**4 Тестирование**

|  |  |  |
| --- | --- | --- |
| № теста | Размер массива size | Время выполнения сортировки в секундах |
| 1 | 1000 | 0.0014 |
| 2 | 2000 | 0.0016 |
| 3 | 3000 | 0.0017 |
| 4 | 4000 | 0.0022 |
| 5 | 5000 | 0.0028 |
| 6 | 6000 | 0.0033 |
| 7 | 7000 | 0.0038 |
| 8 | 8000 | 0.0045 |
| 9 | 9000 | 0.0049 |
| 10 | 10000 | 0.0055 |

|  |  |  |
| --- | --- | --- |
| № теста | Размер массива size | Время выполнения сортировки в секундах |
| 1 | 10000 | 0.00059 |
| 2 | 20000 | 0.00118 |
| 3 | 30000 | 0.00177 |
| 4 | 40000 | 0.00219 |
| 5 | 50000 | 0.00272 |
| 6 | 60000 | 0.00328 |
| 7 | 70000 | 0.00380 |
| 8 | 80000 | 0.00438 |
| 9 | 90000 | 0.00498 |
| 10 | 100000 | 0.00554 |

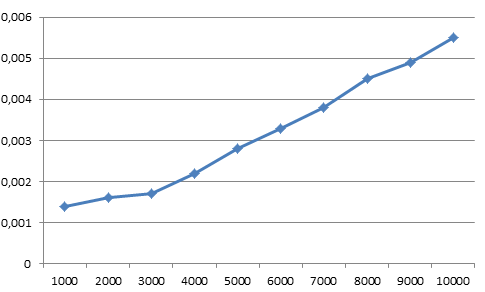


Диаграмма 1 “Тестирование программы на диапазоне от 1 до 10000”

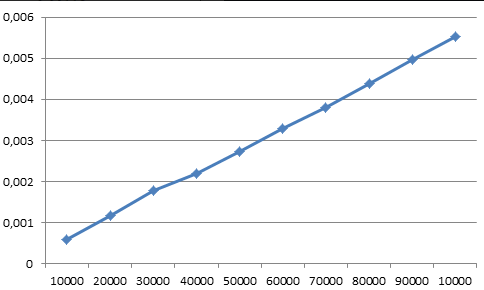


Диаграмма 2 “Тестирование программы на диапазоне от 1 до 100000”

**5 Работа с GitHub**

В репозитории было создано три ветки, помимо главной, где участники команды могли создавать и редактировать проект.

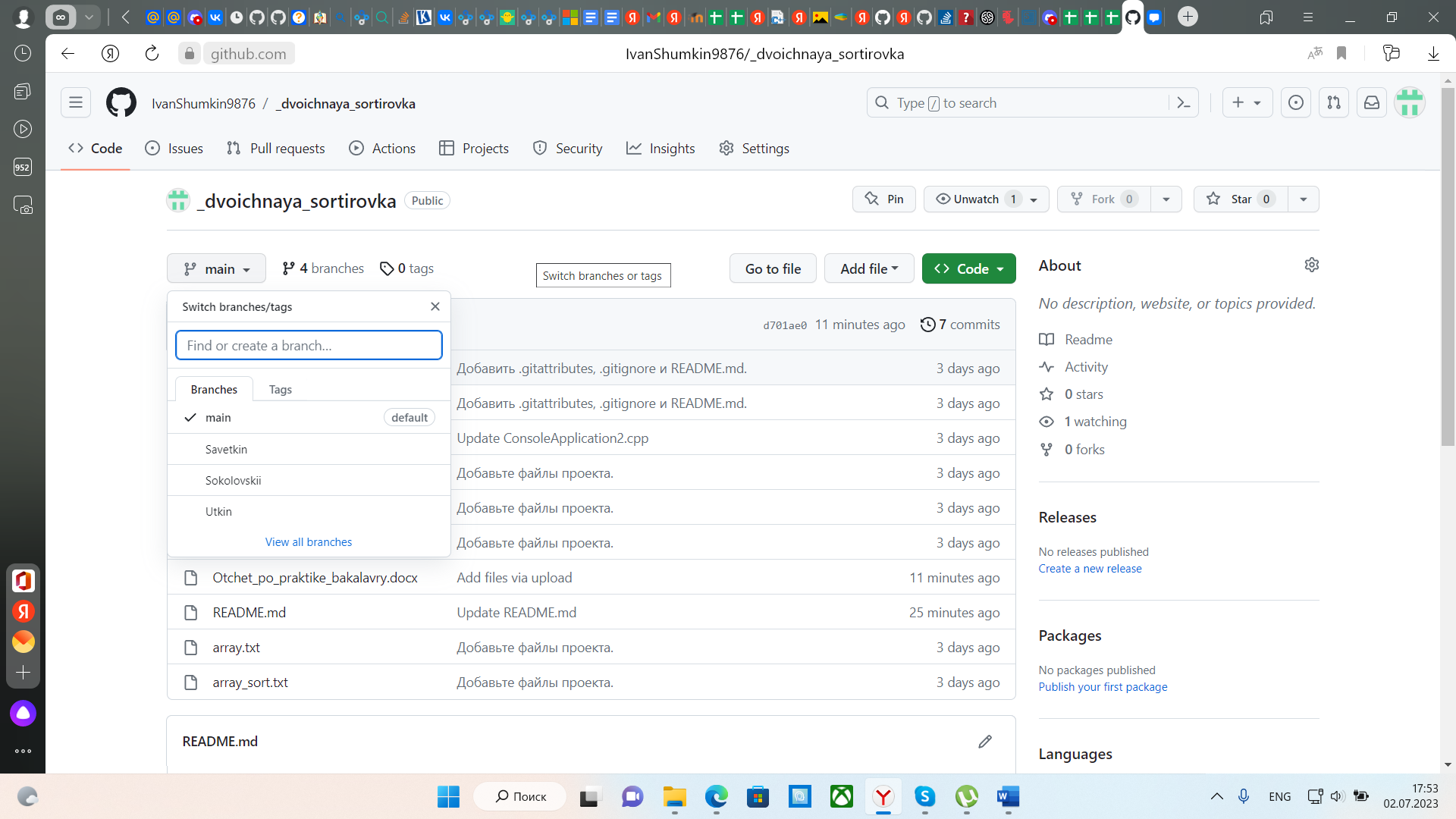
****

Рисунок 2 - Создание веток в GitHub

**6 Заключение**

На основании анализа данных, полученных в результате тестирования алгоритма двоичной сортировки, можно сделать вывод, что время, затраченное на работу программы относительно количества элементов увеличивается линейно, то есть с увеличением количества элементов пропорционально увеличивается время работы программы.

**Список литературы**

1. Стивен С. Скиена. Алгоритмы. Руководство по разработке, 2011 г.
2. Герберт Шилдт. С++ для начинающих. Шаг за шагом. 2013 г.
3. Стивен Прата. Язык программирования С++. Лекции и упражнения. 2018 г.

**Приложение А**

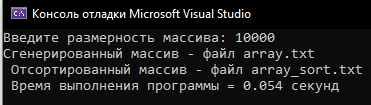


Рисунок А.1

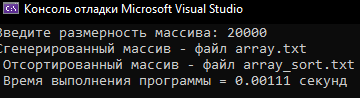


Рисунок А.2

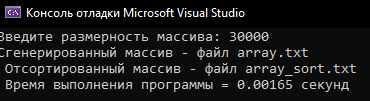


Рисунок А.3

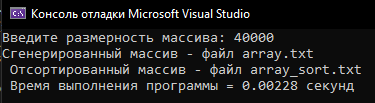


Рисунок А.4

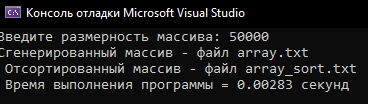


Рисунок А.5

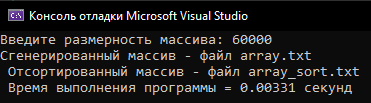


Рисунок А.6

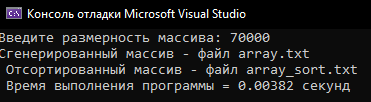


Рисунок А.7

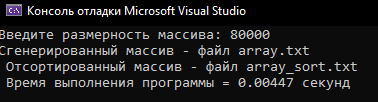


Рисунок А.8

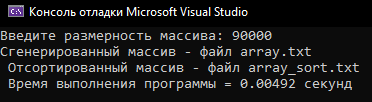


Рисунок А.9

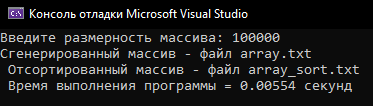


Рисунок А.10

**Приложение Б**

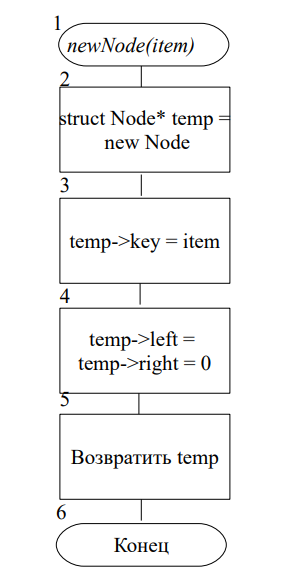
****

Рисунок Б.1 - Блок-схема функции newNode

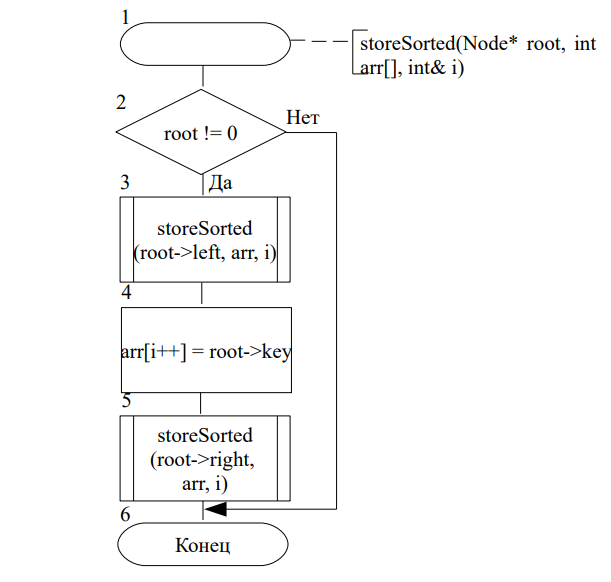


Рисунок Б.2 -Блок-схема функции storeSorted

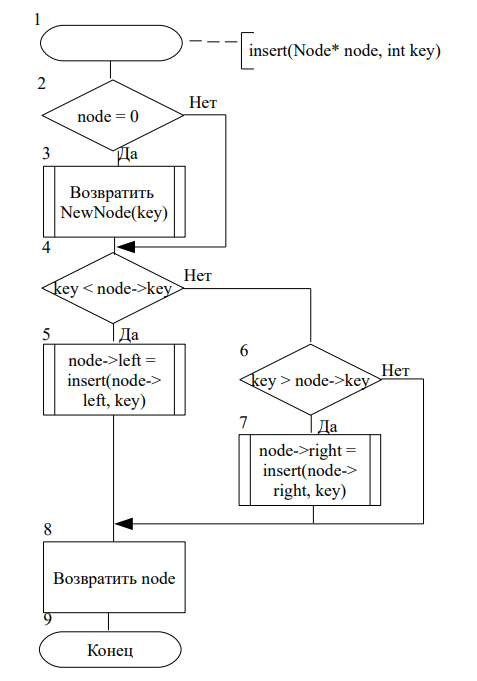


Рисунок Б.3 -Блок-схема функции insert

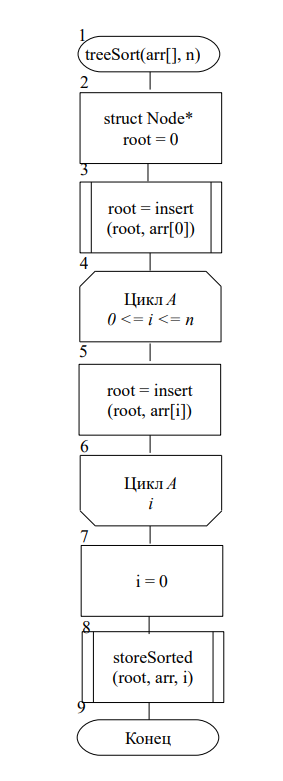


Рисунок Б.4 -Блок-схема функции treeSort

**Приложение В – Листинг**

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <ctime>

using namespace std;

struct Node

{

int key;

struct Node\* left, \* right;

};

struct Node\* newNode(int item)

{

struct Node\* temp = new Node;

temp->key = item;

temp->left = temp->right = 0;

return temp;

}

void storeSorted(Node\* root, int arr[], int& i)

{

if (root != 0)

{

storeSorted(root->left, arr, i);

arr[i++] = root->key;

storeSorted(root->right, arr, i);

}

}

Node\* insert(Node\* node, int key)

{

// Возвращается новый узел, если дерево "пустое"

if (node == 0) return newNode(key);

// Если дерево не пустое, то идет возврт вниз по дереву

if (key < node->key)

node->left = insert(node->left, key);

else if (key > node->key)

node->right = insert(node->right, key);

// Возвращается указатель узла

return node;

}

// Функция сортировки массива

void treeSort(int arr[], int n)

{

struct Node\* root = 0;

root = insert(root, arr[0]);

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

root = insert(root, arr[i]);

int i = 0;

storeSorted(root, arr, i);

}

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(0));

ofstream f1, f2;

int n = 0;

f1.open("array.txt");

f2.open("array\_sort.txt");

cout << "Введите размерность массива: ";

cin >> n;

int arr[100000];

double start\_time = clock() / 100000.0; // начальное время

for (int i = 0; i < n; i++) {

arr[i] = rand() % 100 + 1;

f1 << arr[i] << ", ";

}

treeSort(arr, n);

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

f2 << arr[i] << ", ";

f1.close();

f2.close();

double end\_time = clock() / 100000.0; // конечное время

double search\_time = end\_time - start\_time;

cout << "Сгенерированный массив - файл array.txt\n Отсортированный массив - файл array\_sort.txt\n ";

cout << "Время выполнения программы = " << search\_time << " секунд \n";

return 0;

}