МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

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

**ОТЧЕТ ПО УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ**

(2022/2023 учебный год)

Куракин Никита Николаевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

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.)*

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(должность, ученая степень, ученое звание)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

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

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

                                          Куракин Никита Николаевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

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

Год обучения                  1                 семестр                 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.)*

Руководитель практики д.т.н., профессор, Зинкин С.А.

*(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 | 30.06.2023 –  02.07.23 |  |
| 3 | Разработка алгоритма | 43 | 02.07.23 –  06.07.23 |  |
| 4 | Описание алгоритма и программы | 18 | 6.07.23 –  08.07.23 |  |
| 5 | Тестирование | 5 | 08.07.23 –  08.07.23 |  |
| 6 | Получение и анализ результатов | 10 | 08.07.23 –  10.07.23 |  |
| 7 | Оформление отчёта | 15 | 10.07.23 –  12.07.2023 |  |
|  | **Общий объём часов** | 108 |  |  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЁТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

Куракин Никита Николаевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

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»

Куракин Н.Н. выполнял практическое задание «Двоичная сортировка». На первоначальном этапе был выбран метод решения и язык программирования С, на котором была написана программа сортировки с помощью двоичного дерева. Оформил отчёт.

Бакалавр Куракин Н.Н.        \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Руководитель Зинкин С.А.   \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2023 г.

практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКИ**

(2022/2023 учебный год)

Куракин Никита Николаевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Программное обеспечение средств вычислительной техники и автоматизированных систем»

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 практики Куракин Н.Н. решал следующие задачи: составление схем программы, реализация алгоритма двоичной сортировки, описание и отладка программы.

За период выполнения практики были освоены основные понятия и технологии двоичной сортировки, реализован метод работы с файлами. Во время выполнения работы Куракин Н.Н. показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике, программированию и сортировке.

За выполнение работы Куракин Н.Н. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики д.т.н., профессор, Зинкин С.А. « » 2023 г.

Содержание

[Введение 7](#_Toc139446455)

[1. Постановка задачи 8](#_Toc139446456)

[1.1 Достоинства алгоритма 8](#_Toc139446457)

[1.2 Недостатки алгоритма 8](#_Toc139446458)

[1.3 Типичные сценарии применения 8](#_Toc139446459)

[2. Выбор решения 9](#_Toc139446460)

[2.1 Выбор среды разработки 9](#_Toc139446461)

[2.2 Выбор языка программирования 10](#_Toc139446462)

[2.3 Выбор системы контроля версий 11](#_Toc139446463)

[3. Описание программы 12](#_Toc139446464)

[4. Схемы программы 14](#_Toc139446465)

[4.1 Блок-схема программы 14](#_Toc139446466)

[4.2 Блок-схема алгоритма 18](#_Toc139446467)

[5. Отладка 22](#_Toc139446468)

[6. Совместная работа 25](#_Toc139446469)

[Заключение 26](#_Toc139446470)

[Список используемой литературы 27](#_Toc139446471)

[Приложение А 28](#_Toc139446472)

[Приложение Б Листинг 30](#_Toc139446473)

[Файл main.c 30](#_Toc139446474)

[Файл main.h 31](#_Toc139446475)

Введение

Развитие человечества привело к появлению огромного количества информации, с которой каждый сталкивается или работает каждый день. Этому немало поспособствовало развитие вычислительной техники, которая в наше время способна обрабатывать информацию со скоростью до нескольких петабайт в секунду, что никак не сравниться с мозгом человека. Более того, скорость обработки человеком массива данных зависит от огромного количества факторов. Для упрощения восприятия человеком информации необходимо форматировать массив данных. Форматирование включает в себя множество действий, среди которых можно выделить сортировку данных по возрастанию или убыванию.

Сортировка – это процесс упорядочивания элементов массива или структуры данных в определенном порядке. Она является одной из основных задач в программировании и используется во многих приложениях, таких как базы данных, сортировка файлов. В данной практической работе мы рассмотрим алгоритм сортировки с помощью двоичного дерева, выделим достоинства и недостатки алгоритма и изучим его реализацию на языке программирования.

1. Постановка задачи

Сортировка с помощью двоичного дерева — универсальный алгоритм сортировки, заключающийся в построении двоичного дерева поиска по ключам массива, 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.

1.1 Достоинства алгоритма

а) Прост в реализации;

б) Хорошее быстродействие;

в) Её можно использовать в реальном времени и каждый раз просто считывать в правильном порядке.

1.2 Недостатки алгоритма

а) Из-за хранения дополнительной структуры используется дополнительная память, которая не используется в быстрой сортировке;

б) Необходимы дополнительные действия – преобразовать массив чисел в двоичное дерево.

1.3 Типичные сценарии применения

Типичные сценарии применения сортировки с помощью двоичного дерева включают в себя:

а) Сортировка массива целых чисел.

б) Сортировка строк текста.

в) Сортировка элементов односвязного/двусвязного списка.

г) Сортировка элементов графа.

д) Сортировка данных в базе данных.

е) Сортировка массивов в многопоточной среде.

1. Выбор решения

2.1 Выбор среды разработки

Для написания кода, отладки и трассировки программы требовалось выбрать интегрированную среду разработки. Среда разработки – комплекс программных средств, используемый для разработки программного обеспечения. Список некоторых интегрированных сред разработки:

а) Eclipse – бесплатная открытая среда разработки;

б) NetBeans – бесплатная открытая среда разработки;

в) Komodo – бесплатная среда разработки;

г) Code::Blocks – бесплатная открытая среда разработки;

д) Microsoft Visual Studio – условно-бесплатная среда разработки;

е) CLion – платная среда разработки.

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Microsoft Visual Studio – продукт компании Microsoft, включающих интегрированную среду разработки программного обеспечения и инструменты, позволяющие разрабатывать как консольные приложения, так и приложения с графическим интерфейсом. Ряд преимуществ среды разработки Visual Studio повлиял на выбор в его пользу:

а) Поддержка технологии IntelliSense, которая позволяет упростить написание программы за счет автоматического дописывания и быстрого доступа к документации;

б) Наличие встроенного отладчика, который позволяет выполнять трассировку программы и находить ошибки;

в) Поддержка плагинов, которые позволяют расширить функциональности среды разработки;

г) Поддержка различных систем контроля версий(в том числе Git), которые позволяют упростить управление версиями в Visual Studio.

Недостатком Visual Studio является её тяжеловесность: для выполнения задач могут потребоваться значительные ресурсы компьютера.

2.2 Выбор языка программирования

Для реализации алгоритмов и создания программы требовалось выбрать язык программирования. Язык программирования – формальная знаковая система, которая определяет набор лексических, синтаксических и сематических правил и предназначена для создания программного обеспечения. Список некоторых языков программирования:

а) Python – объектно-ориентированный высокоуровневый язык программирования;

б) C – процедурный низкоуровневый язык программирования;

в) C++ –объектно-ориентированный среднеуровневый язык программирования;

г) Java – объектно-ориентированный среднеуровневый язык программирования.

В качестве языка программирования был выбран язык C. C представляет собой универсальный язык программирования, сочетание возможностей языков программирования низкого и высокого уровней. Ряд преимуществ языка программирования C повлиял на выбор в его пользу:

а) Поддержка работы с памятью. Язык C позволяет взаимодействовать с оперативной памятью: выделять или занимать ячейки памяти, обращаться к ячейкам памяти по адресу;

б) Поддержка многомодульной программной структуры;

в) Простая языковая база, из которой вынесены многие существенные математические функции, функции ввода/вывода, функции работы с файлами;

г) Система типов, структур и объединений, позволяющая исключить бессмысленные операции. Наличие определяемых пользователем типов данных с возможностью управления ими.

Недостатком C является отсутствие контроля переполнения и выхода за разрешенные границы памяти, отсутствие механизма обработки ошибок.

2.3 Выбор системы контроля версий

Для реализации совместной работы и управления версиями проекта требовалось выбрать систему контроля версий. Система контроля версий – это система, регистрирующая изменения в файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Список некоторых языков программирования:

а) RCS – система управления пересмотрами версий;

б) CVS - система управления параллельными версиями;

в) Git – система управления версиями;

г) Monotone – система управления версиями.

В качестве системы контроля версий была выбрана система Git и веб-хостинг репозиториев GitHub. Git – система контроля версий, которая позволяет отслеживать любые изменения в файлах, хранить их версии и оперативно возвращаться в любое сохранённое состояние. GitHub – веб-хостинг версий, работающая в паре с системой контроля версий Git и позволяющая хранить версии “в облаке”. Ряд преимуществ системы Git повлиял на выбор в его пользу:

а) Наличие истории изменений версий, позволяющая осматривать ход разработки или изменения репозитория;

б) Возможность ветвления репозитория, позволяющая реализовывать процесс разработки с различными функциональными ответвлениями, которые в любой момент можно объединить;

в) Возможность “клонировать” репозиторий на свое устройство, что позволяет получить его последнюю версию и изменять его локально;

г) Возможность командной работы над репозиторием.

Недостатком системы контроля версий Git является отслеживание изменений всего проекта целиком, а не отдельных файлов.

1. Описание программы

Приложение BinaryTreeSort.exe является основным модулем программы. При запуске программы пользователю выводится меню, после чего ему предлагается выбрать пункт меню. В таблице 1 выполнено описание состояний программы при выборе пункта:

Таблица 1 – Описание состояний

|  |  |  |
| --- | --- | --- |
| Клавиша, вызывающие событие | Действие пользователя | Действие программы |
| 1 | Генерация | Поочередно запрашивает следующие параметры: название файла, количество генерируемых элементов, интервал генерации. Генерирует массив случайных чисел, в соответствие с указанными параметрами и заносит массив в файл. Выводит количество сгенерированных чисел и затраченное время, после ждет нажатие клавиши для возврата в меню. |
| 2 | Сортировка по возрастанию | Поочередно запрашивает следующие параметры: название входного и выходного файла. Двоичным методом сортирует массив входного файла по возрастанию и выводит отсортированный массив в выходной файл. Выводит количество отсортированных чисел и затраченное время, после ждет нажатие клавиши для возврата в меню. |
| 3 | Сортировка по убыванию | Поочередно запрашивает следующие параметры: название входного и выходного файла. Двоичным методом сортирует массив входного файла по убыванию и выводит отсортированный массив в выходной файл. Выводит количество отсортированных чисел и затраченное время, после ждет нажатие клавиши для возврата в меню. |
| 4 | Выход из программы | Закрывает программу. |

1. Схемы программы

4.1 Блок-схема программы

Рисунок 1 демонстрирует схему функции StageGenerate().



Рисунок 1 – блок схема функции StageGenerate()

Рисунок 2 демонстрирует схему функции StageSort().



Рисунок 2 – блок схема функции StageSort()

Рисунок 3 демонстрирует схему функции GenerateFile().



Рисунок 3 – блок схема функции GenerateFile()

Рисунок 4 демонстрирует схему функции FileToTreeInput().



Рисунок 4 – блок схема функции FileToTreeInput()

Рисунок 5 демонстрирует схему функции BinarySortOutput().



Рисунок 5 – блок схема функции BinarySortOutput()

4.2 Блок-схема алгоритма

Рисунок 6 демонстрирует схему алгоритма преобразования массива чисел в двоичное дерево.



Рисунок 6– блок схема алгоритма преобразования массива чисел в двоичное дерево

Рисунок 7 демонстрирует схему алгоритма сортировки с помощью двоичного дерева по возрастанию.



Рисунок 7 – блок схема алгоритма сортировки с помощью двоичного дерева по возрастанию

Рисунок 8 демонстрирует схему алгоритма сортировки с помощью двоичного дерева по убыванию.



Рисунок 8 – блок схема алгоритма сортировки с помощью двоичного дерева по убыванию

1. Отладка

Рисунки 9, 10 демонстрирует состояние программы при выборе пункта “Генерация”.

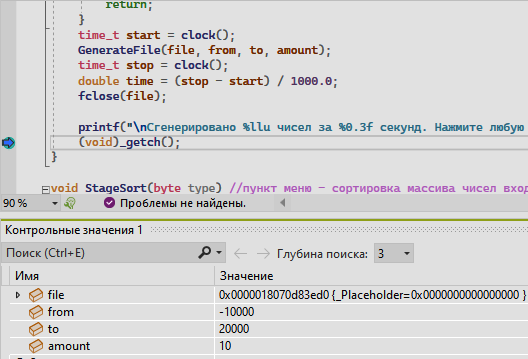


Рисунок 9 – состояние локальных переменных и точка останова

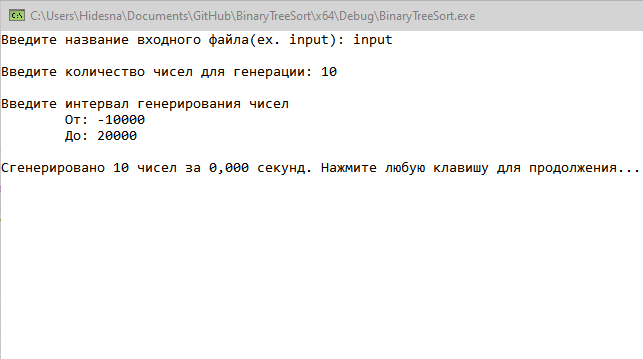


Рисунок 10 – состояние программы

Рисунки 11, 12, 13, 14 демонстрируют состояние программы при выборе пунктов “Сортировка по возрастанию”/“Сортировка по убыванию”.

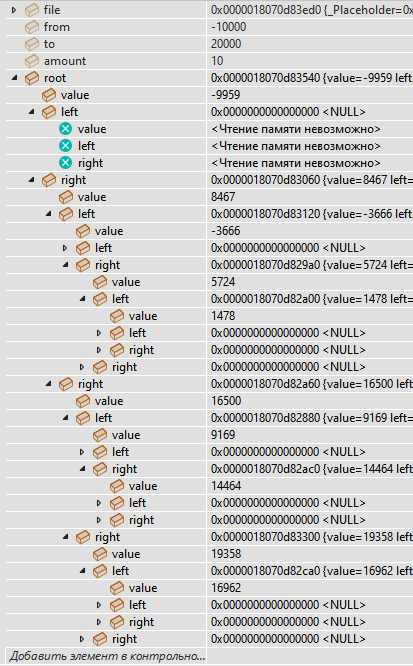


Рисунок 11 – часть преобразованного в двоичное дерево массива

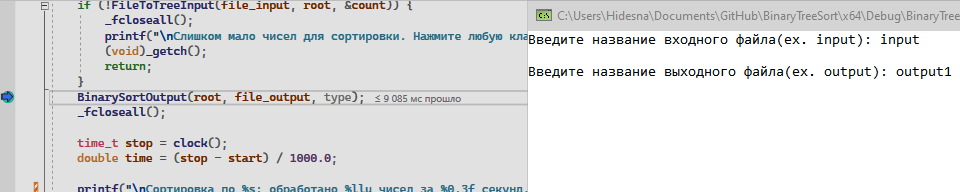


Рисунок 12 – состояние программы

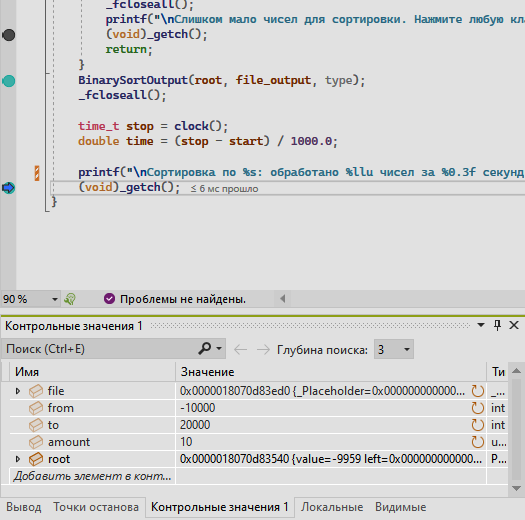


Рисунок 13 – состояние локальных переменных и точка останова

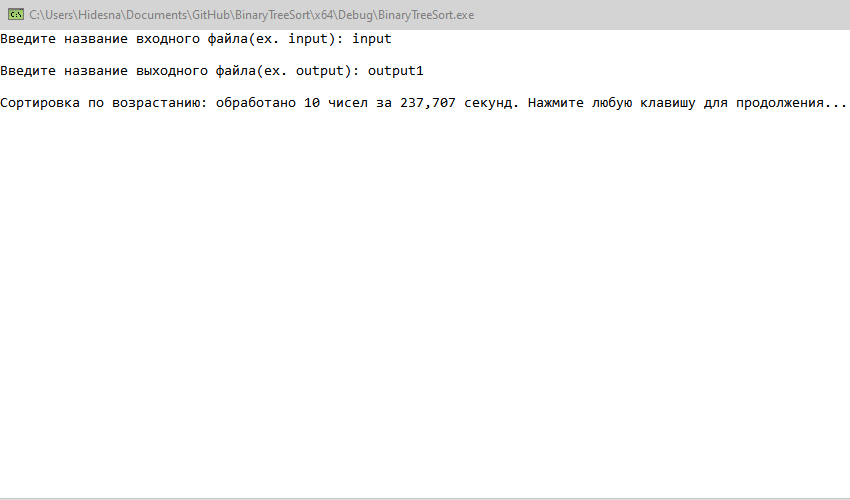


Рисунок 14 – состояние программы

1. Совместная работа

Совместная работа над практическим заданием обеспечивалась благодаря распределением ролей на каждого из участников бригады. В данной работе я выполнил ряд функций, которые поспособствовали завершению практической работы. Моя роль включала следующие задачи: выбор решения, разработка схем алгоритма, реализация алгоритма, а также отладка и трассировка программы.

Второй участник бригады, Майоров Никита, принял на себя роль объяснения алгоритма сортировки с помощью двоичного дерева, используемого в программе, реализации интерфейса программы, а также проведения тестирования с различными наборами данных, что позволило нам проверить работоспособность и эффективность программы, сделать выводы по использованному алгоритму.

В нашем взаимодействии я и второй участник бригады использовали систему контроля версий Git, что обеспечило надежность и эффективность в работе с кодом и алгоритмом.

Заключение

В ходе выполнения практического задания были получены навыки реализации алгоритма двоичной сортировки в программах, приобретен опыт работы с файлами и динамическим выделением памяти. Были отточены навыки отладки и трассировки программ, работы с контрольными и видимыми значениями. Была применена в ходе написания программы среда Microsoft Visual Studio на языке C.

Результатом нашей совместной работы стала программа, позволяющая сортировать массивы чисел, находящиеся в текстовых документах, определенным алгоритмом.

Список используемой литературы

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.

2. Плаугер П. Стандартная библиотека языка С. 1992 г.

3. Красиков И.В. Алгоритмы. Просто как дважды два. 2007 г.

Приложение А

Рисунок 15 демонстрирует функциональное меню приложения.



Рисунок 15 – Функциональное меню приложения

Рисунок 16 демонстрирует состояние программы после выбора пункта “Генерация”.

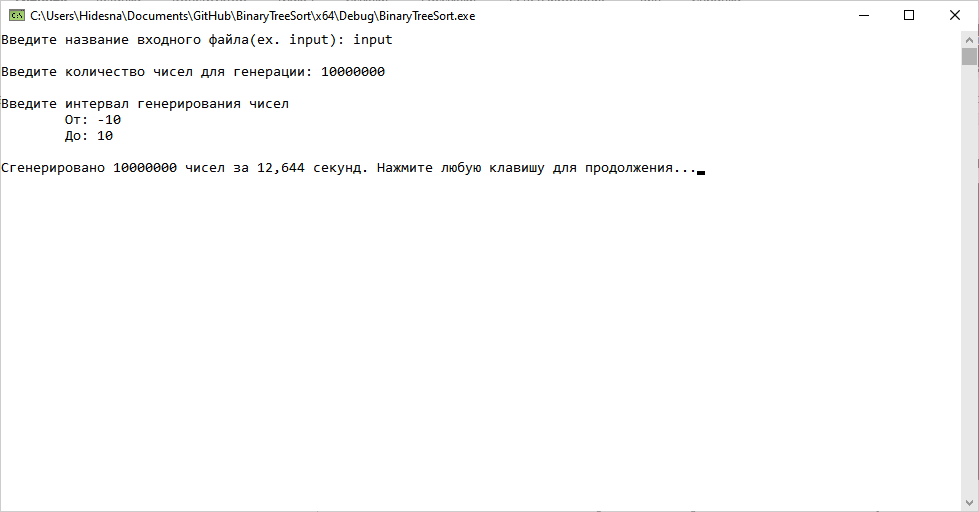


Рисунок 16 – состояние программы после выбора пункта “Генерация”

Рисунок 17 демонстрирует состояние программы после выбора пункта “Сортировка по возрастанию”.

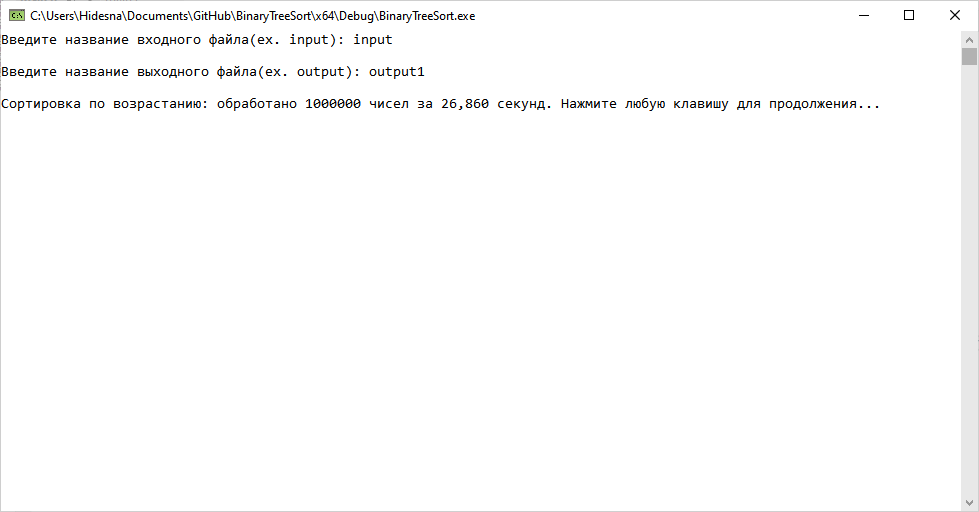


Рисунок 17 – состояние программы после выбора пункта “ Сортировка по возрастанию”

Рисунок 18 демонстрирует состояние программы после выбора пункта “Сортировка по убыванию”.

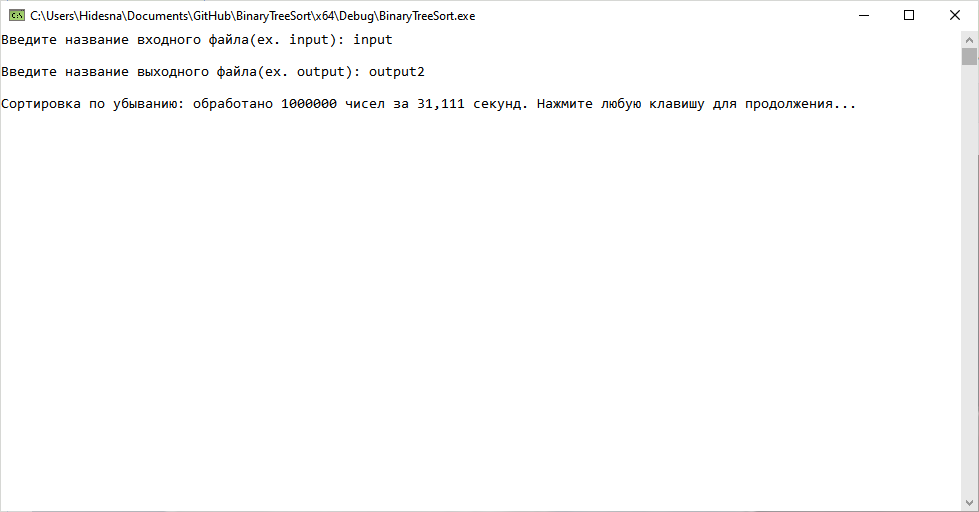


Рисунок 18 – состояние программы после выбора пункта “ Сортировка по убыванию”

Приложение Б Листинг

Файл main.c

#include "main.h"

int main() {

//формат кодировки github

setlocale(LC\_ALL, ".UTF8");

system("cls");

char menu[] = "\n\n\tМеню\n\n"

"\t\tИсходные данные\n"

"\t1) Генерация\n\n"

"\t\tСортировка\n"

"\t2) Сортировка по возрастанию\n"

"\t3) Сортировка по убыванию\n\n"

"\t\tПрограмма\n"

"\t4) Выход из программы\n";

printf(menu);

for (;;) {

switch (getch()) {

case '1':

//генерация массива чисел для входного файла

StageGenerate();

system("cls");

printf(menu);

break;

case '2':

//сортировка входного файла по возрастанию для выходного файла

StageSort(1);

system("cls");

printf(menu);

break;

case '3':

//сортировка входного файла по убыванию для выходного файла

StageSort(2);

system("cls");

printf(menu);

break;

case '4':

//выход из программы

return 0;

break;

}

}

}

Файл main.h

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#include <memory.h>

#include <time.h>

typedef char byte;

//структура узла

struct PARENT

{

int value;

struct PARENT\* left;

struct PARENT\* right;

};

//вспомогательные функции

//генерация данных

void GenerateFile(FILE\* file, int range\_from, int range\_to, unsigned long long amount)

{

for (long long i = 0; i != amount; i++) {

fprintf(file, "%d\n", range\_from + rand() % (range\_to - range\_from + 1));

}

}

//преобразование массива данных из файла в двоичное дерево

byte FileToTreeInput(FILE\* file, struct PARENT\* root, unsigned long long\* count)

{

struct PARENT\* current = root;

if (fscanf(file, "%i\n", &(root->value)) <= 0)

return 0;

int tmp = 0;

for (\*count = 1;;(\*count)++) {

if (fscanf(file, "%i\n", &tmp) <= 0)

return 1;

for (;;) {

if (tmp >= current->value) {

if (!current->right) {

current->right = (struct PARENT\*)malloc(sizeof(struct PARENT));

if (!current->right)

exit(EXIT\_FAILURE);

current->right->left = NULL; current->right->right = NULL;

current->right->value = tmp;

current = root;

break;

}

else

current = current->right;

}

else if (tmp < current->value) {

if (!current->left) {

current->left = (struct PARENT\*)malloc(sizeof(struct PARENT));

if (!current->left)

exit(EXIT\_FAILURE);

current->left->left = NULL; current->left->right = NULL;

current->left->value = tmp;

current = root;

break;

}

else

current = current->left;

}

}

}

}

//бинарная сортировка, вывод в выходной файл

void BinarySortOutput(struct PARENT\* root, FILE\* file, byte type)

{

if (!root)

return;

if (type == 1) {

BinarySortOutput(root->left, file, type);

fprintf(file, "%d\n", root->value);

BinarySortOutput(root->right, file, type);

}

else {

BinarySortOutput(root->right, file, type);

fprintf(file, "%d\n", root->value);

BinarySortOutput(root->left, file, type);

}

free(root);

}

//основные функции

//функция меню - генерация массива чисел для входного файла

void StageGenerate()

{

system("cls");

char filename[260] = "", \* tmp;

int from = 0, to = 0;

unsigned long long amount = 0;

for (;;) {

printf("Введите название входного файла(ex. input): ");

tmp = fgets(filename, sizeof(filename), stdin);

fseek(stdin, 0, SEEK\_END);

if (\*tmp != '\n')

break;

system("cls");

}

tmp = strchr(filename, '\n');

strcpy(tmp, ".txt");

do {

printf("\nВведите количество чисел для генерации: ");

(void)scanf("%llu", &amount);

} while (amount == 0);

do {

printf("\nВведите интервал генерирования чисел\n\tОт: ");

(void)scanf("%d", &from);

printf("\tДо: ");

(void)scanf("%d", &to);

} while (from>to);

FILE\* file = fopen(filename, "w");

if (!file) {

printf("\nНе удалось создать/открыть файл. Нажмите любую клавишу для продолжения...");

(void)\_getch();

return;

}

time\_t start = clock();

GenerateFile(file, from, to, amount);

time\_t stop = clock();

double time = (stop - start) / 1000.0;

fclose(file);

printf("\nСгенерировано %llu чисел за %0.3f секунд. Нажмите любую клавишу для продолжения...", amount, time);

(void)\_getch();

}

//пункт меню - сортировка массива чисел входного файла для выходного файла

void StageSort(byte type)

{

system("cls");

char filename\_input[260] = "", filename\_output[260] = "", \* tmp;

unsigned long long count = 0;

for (;;) {

printf("Введите название входного файла(ex. input): ");

tmp = fgets(filename\_input, sizeof(filename\_input), stdin);

fseek(stdin, 0, SEEK\_END);

if (\*tmp != '\n')

break;

system("cls");

}

tmp = strchr(filename\_input, '\n');

strcpy(tmp, ".txt");

FILE\* file\_input = fopen(filename\_input, "r");

if (!file\_input) {

printf("\nФайл не найден. Нажмите любую клавишу для продолжения...");

(void)\_getch();

return;

}

for (;;) {

printf("\nВведите название выходного файла(ex. output): ");

tmp = fgets(filename\_output, sizeof(filename\_output), stdin);

fseek(stdin, 0, SEEK\_END);

if (\*tmp != '\n')

break;

system("cls");

}

tmp = strchr(filename\_output, '\n');

strcpy(tmp, ".txt");

FILE\* file\_output = fopen(filename\_output, "w");

if (!file\_output) {

fclose(file\_input);

printf("\nНе удалось создать/открыть файл. Нажмите любую клавишу для продолжения...");

(void)\_getch();

return;

}

time\_t start = clock();

struct PARENT\* root = (struct PARENT\*)malloc(sizeof(struct PARENT));

if (!root)

exit(EXIT\_FAILURE);

root->left = NULL; root->right = NULL;

if (!FileToTreeInput(file\_input, root, &count)) {

\_fcloseall();

free(root);

remove(filename\_output);

printf("\nСлишком мало чисел для сортировки. Нажмите любую клавишу для продолжения...");

(void)\_getch();

return;

}

BinarySortOutput(root, file\_output, type);

\_fcloseall();

time\_t stop = clock();

double time = (stop - start) / 1000.0;

printf("\nСортировка по %s: обработано %llu чисел за %0.3f секунд. Нажмите любую клавишу для продолжения...", type == 1 ? "возрастанию" : "убыванию", count, time);

(void)\_getch();

}