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)

                                       Аргаткин Алексей Алексеевич

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 30.06.2023 по 08.07.2023

Кафедра «Вычислительная техника»

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

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

Руководитель практики к.т.н., доцент, 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)

                                       Аргаткин Алексей Алексеевич

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 30.06.2023 по 08.07.2023

Кафедра «Вычислительная техника»

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

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

Руководитель практики к.т.н., доцент, 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 | 29.06.2022 -  29.06.2022 |  |
| 2 | Подбор и изучение материала по теме работы | 15 | 30.06.2022 –  02.07.22 |  |
| 3 | Разработка алгоритма | 43 | 02.07.22 –  06.07.22 |  |
| 4 | Описание алгоритма и программы | 18 | 06.07.22 –  08.07.22 |  |
| 5 | Тестирование | 5 | 08.07.22 –  08.07.22 |  |
| 6 | Получение и анализ результатов | 10 | 08.07.22 –  10.07.22 |  |
| 7 | Оформление отчёта | 15 | 10.07.22 –  12.07.2022 |  |
|  | **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)

                                       Аргаткин Алексей Алексеевич

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 30.06.2022 по 08.07.2022

Кафедра «Вычислительная техника»

Горбатов К.В. выполнял практическое задание «Двоичная сортировка». На первоначальном этапе были изучен и проанализирован алгоритм данной сортировки, был выбран метод решения и язык программирования С, на котором была написана программа двоичной сортировки. Также была осуществлена работа с файлами. Оформил отчёт.

Бакалавр Аргаткин А.А. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 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)

                                       Аргаткин Алексей Алексеевич

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 30.06.2023 по 08.07.2023

Кафедра «Вычислительная техника»

В процессе выполнения практики Горбатов К.В. решал следующие задачи: создание алгоритма двоичной сортировки, анализ работы алгоритма.

За период выполнения практики были освоены основные понятия и технологии двоичной сортировки, реализован метод работы с файлами. Во время выполнения работы Горбатов К.В. 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 Горбатов К.В. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики к.т.н., доцент, Зинкин.С.А « » 2023 г.

Оглавление

[**1.Введение** 2](#_Toc139491868)

[**2.Постановка задачи** 5](#_Toc139491869)

[**3.Выбор решения** 8](#_Toc139491870)

[**4.Схема программы** 9](#_Toc139491871)

[**5.Тестирование программы** 10](#_Toc139491872)

[**6.Отладка** 11](#_Toc139491873)

[**7.Работа со средой Github** 13](#_Toc139491874)

[**8.Описание программы** 14](#_Toc139491875)

[**9.Заключение** 19](#_Toc139491876)

[**10.Список литературы:** 20](#_Toc139491877)

[**11.Приложение A. Листинг программы** 21](#_Toc139491878)

# **Введение**

Двоичная сортировка представляет собой алгоритм, построенный на бинарном дереве.

Суть алгоритма состоит в следующем: входными данными является последовательность чисел, которая на практике представляется в виде динамического списка. Во время заполнения списка алгоритм автоматически его сортирует от наименьшего к наибольшему. За первый элемент дерева берется первый элемент списка, после чего он сравнивается с дальнейшими числами списка, которые будут добавляться и располагаться в дереве в нужном им месте. Эти действия происходят пока происходят добавления в список новых элементов.

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Функциональная структура среды включает в себя:

1) редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3);

2) отладчик кода;

3) редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

4) веб-редактор;

5) дизайнер классов;

6) дизайнер схем баз данных.

Visual Studio также позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения).

В комплект входят следующие основные компоненты:

1. Visual Basic.NET – для разработки приложений на VisualBasic;
2. Visual C++ – на традиционном языке C++;
3. Visual C# – на языке C# (Microsoft);
4. Visual F# – на F# (Microsoft Developer Division).

Интегрированная среда разработки (IntegratedDevelopmentEnvironment – IDE) Visual Studio предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio:

1. Встроенный Web-сервер;
2. Поддержка множества языков при разработке;
3. Интуитивный стиль кодирования;
4. Более высокая скорость разработки;
5. Возможности отладки.

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

Своими корнями он уходит в язык Си, который был разработан в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи. В начале 1980-х годов датский программист Бьерн Страуструп, который в то время работал в компании Bell Labs, разработал С++ как расширение к языку Си. Фактически вначале C++ просто дополнял язык Си некоторыми возможностями объектно-ориентированного программирования. И поэтому сам Страуструп вначале называл его как "C with classes" ("Си с классами").

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где С++ тоже нашел свое применение. И даже в веб-разработке также можно использовать С++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем С++ – язык широкого пользования, на котором можно создавать практически любые виды программ.

# **Постановка задачи**

Необходимо разработать алгоритм двоичной сортировки. Конечная программа должна реализовывать непосредственно алгоритм, а также выполнять считывание входных данных из файла и запись результатов в файл и отображать все данные в интерфейсе.

Сортировка по двоичной вставке - это алгоритм сортировки, который похож на [сортировку по вставке](https://translated.turbopages.org/proxy_u/en-ru.ru.80d84ebb-62c46a5c-cf0b2608-74722d776562/www.geeksforgeeks.org/insertion-sort/), но вместо использования линейного поиска для поиска местоположения, в которое должен быть вставлен элемент, мы используем [двоичный поиск](https://translated.turbopages.org/proxy_u/en-ru.ru.80d84ebb-62c46a5c-cf0b2608-74722d776562/https/www.geeksforgeeks.org/binary-search/). Таким образом, мы уменьшаем сравнительную ценность вставки одного элемента с O (N) до O(logN).

**Бинарное (двоичное) дерево поиска** – это бинарное дерево, для которого выполняются следующие дополнительные условия (свойства дерева поиска):

1. оба поддерева – левое и правое, являются двоичными деревьями поиска;
2. у всех узлов левого поддерева произвольного узла X значения ключей данных меньше, чем значение ключа данных самого узла X;
3. у всех узлов правого поддерева произвольного узла X значения ключей данных не меньше, чем значение ключа данных узла X.

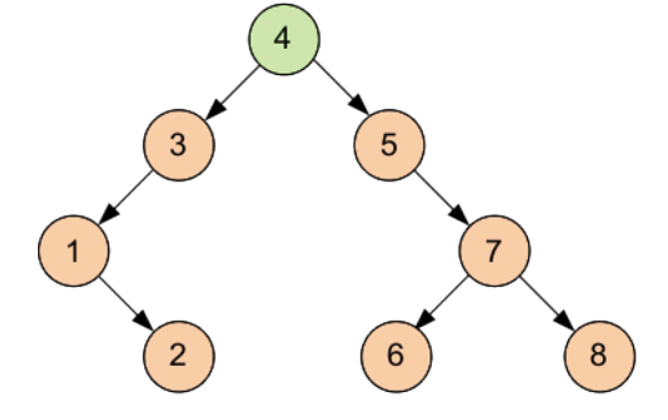
Данные в каждом узле должны обладать ключами, на которых определена операция сравнения меньше.

Для сортировки с помощью дерева исходная сортируемая последовательность представляется в виде структуры данных «дерево».

Например, исходная последовательность имеет вид:

**4, 3, 5, 1, 7, 8, 6, 2**

Корнем дерева будет начальный элемент последовательности. Далее все элементы, меньшие корневого, располагаются в левом поддереве, все элементы, большие корневого, располагаются в правом поддереве. Причем это правило должно соблюдаться на каждом уровне.  
После того, как все элементы размещены в структуре «дерево», необходимо вывести их, используя

  
рисунок 1- Дерево бинарной сортировки

**Достоинства алгоритма**

* Эффективен на небольших наборах данных
* Эффективен на уже частично отсортированной последовательности
* Может сортировать последовательность по мере её получения
* Алгоритм является устойчивым, т. е. уже отсортированная часть не изменяется

**Недостатки алгоритма**

* Не подходит для больших последовательностей
* Количество сравнений в сортировке вставками зависит от изначальной упорядоченности списка. Если список уже отсортирован, количество сравнений равно n - 1; в противном случае его производительность является величиной порядка n2 [2].

**Типичные сценарии использования**

Данная сортировка актуальна для использования в программах с небольшим количеством данных. Алгоритм можно применять для сортировки с целью ускорения поиска каких-либо элементов в программах, для упорядочивания элементов.

1. **Схема программы**

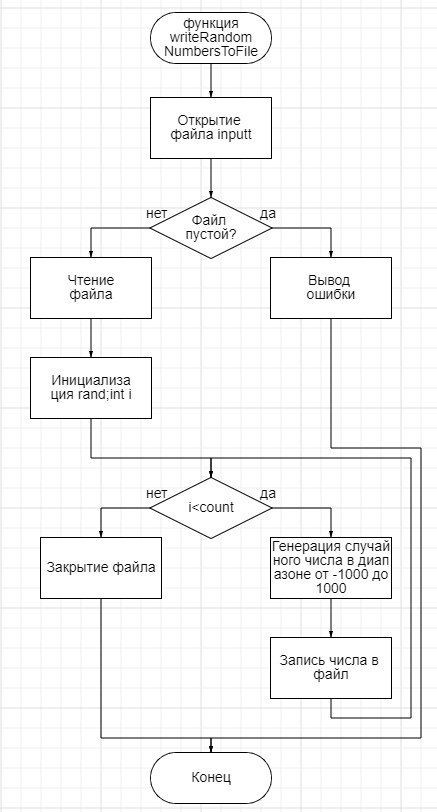


Рисунок 1.Схема Функции writeRandomNumbersToFile

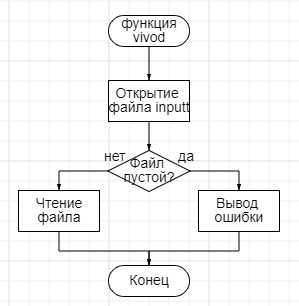
****

Рисунок 2.Схема Функции vivod

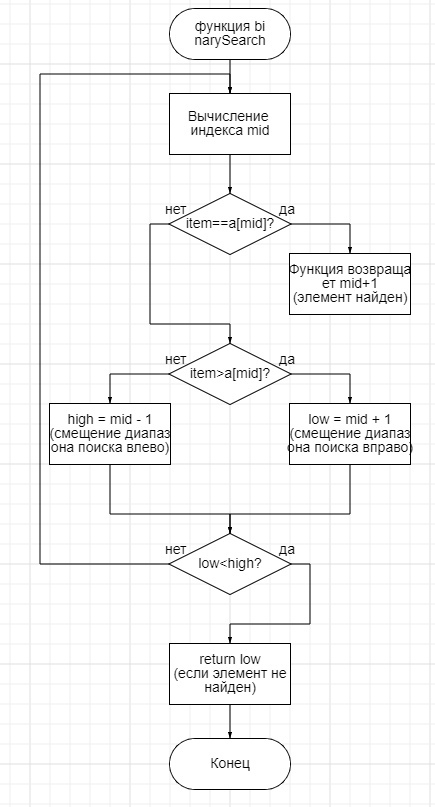


Рисунок 3.Схема Функции binarySearch

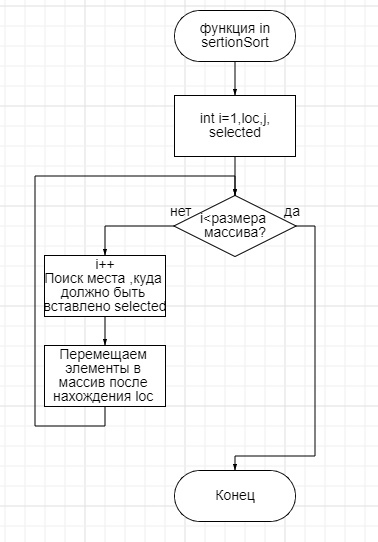


Рисунок 4. Схема функции insertionSort

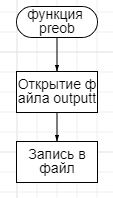
****

Рисунок 5. Схема Функции preob

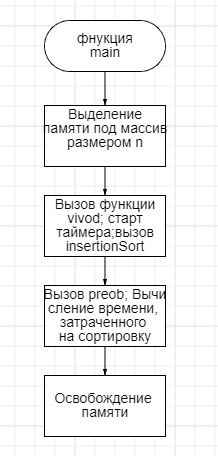


Рисунок 6. Схема программы

**5.Тестирование программы**

Тестовый набор данных представлен в таблице 1.

|  |  |  |
| --- | --- | --- |
| № Теста | Размер массива arr[] | Время выполнения сортировки в секундах |
| 1 | 10000 | 0.039 |
| 2 | 20000 | 0.135 |
| 3 | 30000 | 0.297 |
| 4 | 40000 | 0.518 |
| 5 | 50000 | 0.781 |
| 6 | 60000 | 1.125 |
| 7 | 70000 | 1.53 |
| 8 | 80000 | 1.991 |
| 9 | 90000 | 2.488 |
| 10 | 100000 | 3.086 |
| 11 | 110000 | 3.718 |

**Анализ полученных результатов тестирования**

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.

# **6.Отладка**

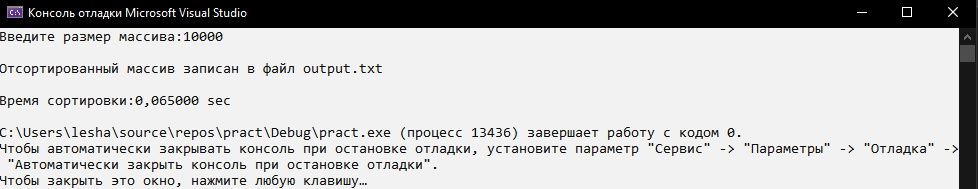
Выполнение работы программой  


Рисунок 6.1 – Результат работы программы

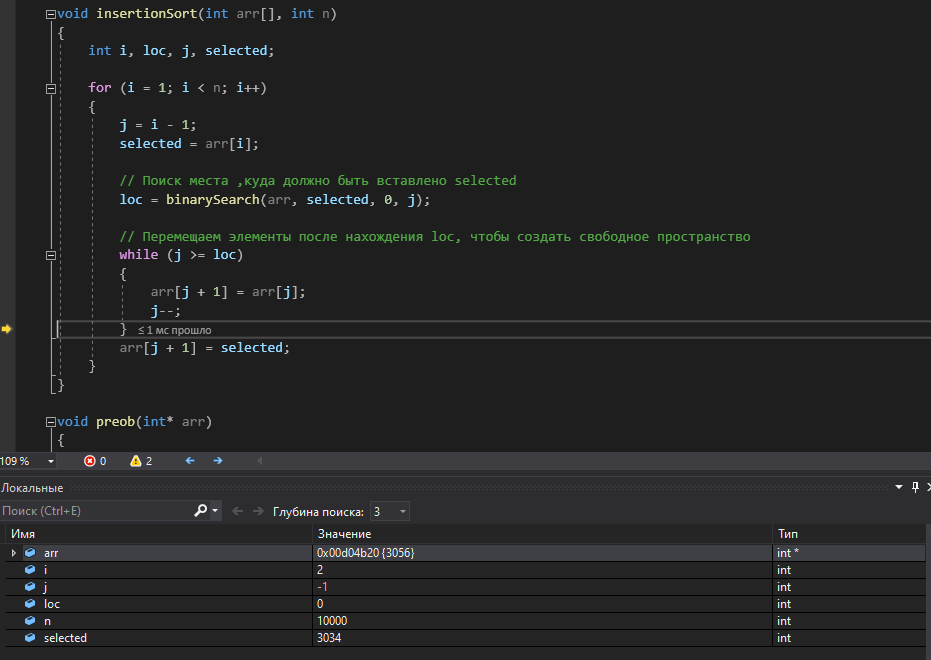
Трассировка  


Рисунок 6.2- Трассировочная таблица

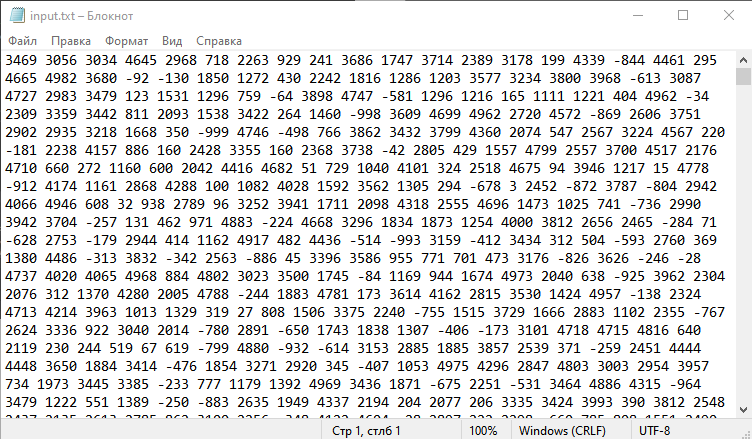
Запись в файлы

Рисунок 6.3- Входные данные

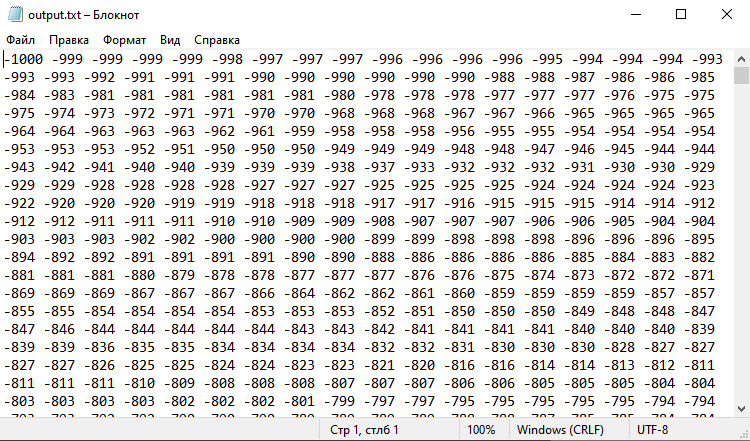


Рисунок 6.4- Выходные данные

# **7.Работа со средой Github**

# Я создал репозиторий с названием praktika и добавил в него код программы, затем я нашёл ошибку в коде и исправил её, после я добавил изменённый код в репозиторий.

# 

# **8.Описание программы**

В программе для двоичной сортировки подключены следующие заголовочные файлы:. stdio.h — это стандартный заголовочный файл ввода-вывода библиотеки языка Си, он содержит константы и объявления функций и типов, определения макросов, которые используются в операциях стандартного ввода и вывода, locale.h. используется для задач, связанных с локализацией (в данной программе использован русский язык). Заголовочный файл time.h содержит типы и функции для работы с датой и временем. Windows.h необходим для использования в программе функционала, предоставляемого операционной системой windows.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

#include <time.h>  
#include <locale.h>

Затем идет глобальная переменная n , которая служит для определения массива  
  
int n;//глобальная переменная для определения размерности массива  
void writeRandomNumbersToFile(const char\* ik, int count) {

FILE\* file = fopen("inputt.txt", "w");

if ("inputt.txt" == NULL) {

printf("Ошибка открытия файла.\n");

return;

}

srand(time(NULL)); // Инициализация генератора случайных чисел

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

int randomNumber = rand() % 2001 - 1000; // Генерация случайного числа в диапазоне от -1000 до 1000

fprintf(file, "%d\n", randomNumber); // Запись числа в файл

}

fclose(file);

}

После нее идет функция для чтения переменных из файла и заполнения массива этими переменными. Так же идет проверка на «пустой файл», если в указанном файле ничего не будет, то программа выдаст ошибку.  
  
FILE\* in = fopen("input.txt", "r");//открытие файла с функцией чтения

if (in == NULL)//проверка на пустой файл

{

printf("Error!\n");//вывод "Ошибка"

}

else

{

for (int i = 0; i < n; i++)//цикл

{

fscanf\_s(in, "%d ", &arr[i]);//чтение из файла и заполнение массива

}

fclose(in);//закрытие файла

printf("Массив до сортировки: \n");

for (int i = 0; i < n; i++)//цикл

{

printf("%d ", \*(arr + i));//вывод не отсортированного массива

}

printf("\n");

}

Далее идет функция бинарной сортировки, в которой определяется наименьшее, наибольшее и среднее значение, с которыми сравниваются элементы массива.  
   
int binarySearch(int a[], int item, int low, int high)//функция бинарной сортировки

{

while (low <= high)

{

int mid = low + (high - low) / 2;

if (item == a[mid])

return mid + 1;

else if (item > a[mid])

low = mid + 1;

else

high = mid - 1;

}

return low;

}

Вместе с ней работает функция по поиску места, в которое должно быть вставлена переменная , которая была принята по одному из трех условий в функции, описанной выше  
void insertionSort(int arr[], int n)

{

int i, loc, j, selected;

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

{

j = i - 1;

selected = arr[i];

// Поиск места ,куда должно быть вставлено selected

loc = binarySearch(arr, selected, 0, j);

// Перемещаем элементы после нахождения loc, чтобы создать свободное пространство

while (j >= loc)

{

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

j--;

}

arr[j + 1] = selected;

}

}

После идет функция для записи отсортированного массива в текстовый файл  
void preob(int\* arr)

{

FILE\* out = fopen("output.txt", "w");

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

{

fprintf(out, "%d ", \*(arr+i));

}

fclose(out);

}

И после всех вышеперечисленных функций, идет основное тело программы, в котором сначала идет изменение цвета консоли  
  
system("color F0");

После идет изменение языка консоли на русский язык  
  
setlocale(0, "");

Далее вводимый пользователем размер массива.  
  
printf("Введите размер массива:");

scanf("%d",&n);

Определение массива.  
int\* arr = new int[n];

Создаются также переменные для последующего подсчета времени, требуемого на сортировку массива.

clock\_t start;

double duration;

После идет вызов функции vivod(arr), которая заполняет массив переменными из файла.  
  
vivod(arr);

Запускается таймер перед вызовом функции двоичной сортировки.  
start = clock();

insertionSort(arr, n);

Затем вызов функции для записи отсортированного массива в файл и оповещение пользователя об этом.  
  
preob(arr);  
printf("Отсортированный массив записан в файл output.txt \n");

Находится и выводится время затраченное на сортировку массива и массив очищается  
  
duration = (double)(clock() - start) / CLOCKS\_PER\_SEC;

delete[] arr;

printf("\nВремя сортировки:%f sec\n", duration);

# **9.Заключение**

В ходе работы мы написали алгоритм сортировки вставками. Протестировали его на разных наборах данных. В результате тестирования мы пришли к выводу, что данный алгоритм подходит для сортировки небольших массивов данных, так как при увеличении количества элементов, увеличивается и время работы.

Таким образом сортировка вставками является эффективной или на небольших последовательностях, или на уже частично отсортированных, а также ее можно применять как вспомогательную. Реализация данного алгоритма проста, но высокая вычислительная сложность не позволяет применять данный алгоритм в повсеместном использовании.

# **10.Список литературы:**

1. Двоичная сортировка [Электронный ресурс] // Wikipedia  
   Режим доступа – URL. <https://ru.wikipedia.org/wiki/Сортировка_с_помощью_двоичного_дерева>
2. Сортировка с помощью древа [Электронный ресурс] // prog-cpp.ru  
   Режим доступа – URL. <https://prog-cpp.ru/sort-tree/>
3. Бинарная сортировка [Электронный ресурс] // geeksforgeeks.org   
   Режим доступа – URL. <https://translated.turbopages.org/proxy_u/en-ru.ru.1f7740d2-62c5796b-f73e1b86-74722d776562/https/www.geeksforgeeks.org/binary-insertion-sort/>
4. Двоичная сортировка [Электронный ресурс] // cyberforum.ru  
   Режим доступа – URL. <https://www.cyberforum.ru/cpp-beginners/thread549088.html>

# **11.Приложение A. Листинг программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

#include <time.h>

#include <locale.h>

int n;//глобальная переменная для определения размерности массива

void writeRandomNumbersToFile(const char\* ik, int count) {

FILE\* file = fopen("inputt.txt", "w");

if ("inputt.txt" == NULL) {

printf("Ошибка открытия файла.\n");

return;

}

srand(time(NULL)); // Инициализация генератора случайных чисел

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

int randomNumber = rand() % 2001 - 1000; // Генерация случайного числа в диапазоне от -1000 до 1000

fprintf(file, "%d\n", randomNumber); // Запись числа в файл

}

fclose(file);

}

void vivod(int\* arr)//функция для чтения из файла

{

FILE\* in = fopen("input.txt", "r");//открытие файла с функцией чтения

if (in == NULL)//проверка на пустой файл

{

printf("Error!\n");//вывод "Ошибка"

}

else

{

for (int i = 0; i < n; i++)//цикл

{

fscanf\_s(in, "%d ", &arr[i]);//чтение из файла и заполнение массива

}

fclose(in);//закрытие файла

printf("\n");

}

}

int binarySearch(int a[], int item, int low, int high)//функция бинарной сортировки

{

while (low <= high)

{

int mid = low + (high - low) / 2;

if (item == a[mid])

return mid + 1;

else if (item > a[mid])

low = mid + 1;

else

high = mid - 1;

}

return low;

}

void insertionSort(int arr[], int n)

{

int i, loc, j, selected;

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

{

j = i - 1;

selected = arr[i];

// Поиск места ,куда должно быть вставлено selected

loc = binarySearch(arr, selected, 0, j);

// Перемещаем элементы после нахождения loc, чтобы создать свободное пространство

while (j >= loc)

{

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

j--;

}

arr[j + 1] = selected;

}

}

void preob(int\* arr)

{

FILE\* out = fopen("output.txt", "w");

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

{

fprintf(out, "%d ", \*(arr + i));

}

fclose(out);

}

// Основное тело программы

int main()

{

system("color F0");

setlocale(0, "");

printf("Введите размер массива:");

scanf("%d", &n);

writeRandomNumbersToFile("random\_numbers.txt", n);

int\* arr = new int[n];

clock\_t start;

double duration;

vivod(arr);

start = clock();

insertionSort(arr, n);

preob(arr);

printf("Отсортированный массив записан в файл output.txt \n");

duration = (double)(clock() - start) / CLOCKS\_PER\_SEC;

delete[] arr;

printf("\nВремя сортировки:%f sec\n", duration);

return 0;

}