Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Тульский государственный университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

СОРТИРОВКА СЛИЯНИЕМ

отчет о
лабораторной работе №5

по дисциплине
ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

ВАРИАНТ 1

Выполнила: ст. гр. 230711 Павлова В.С.
Проверил: асс. каф. ИБ Курбаков М.Ю.

Тула, 2022 г.

ЦЕЛЬ И ЗАДАЧА РАБОТЫ

Цель: изучить принципы сортировки слиянием файлов и массивов в памяти.

Задача: в данной работе требуется написать программу, демонстрирующую использование изученных принципов.

ЗАДАНИЕ НА РАБОТУ

Написать программу сортировки слиянием по возрастанию двух файлов разного размера, используя метод естественного двух путевого слияния.

СХЕМА ПРОГРАММЫ

Схема алгоритма для сортировки слиянием по возрастанию двух файлов разного размера с использованием метода естественного двух путевого слияния, представлена на рисунке 1.

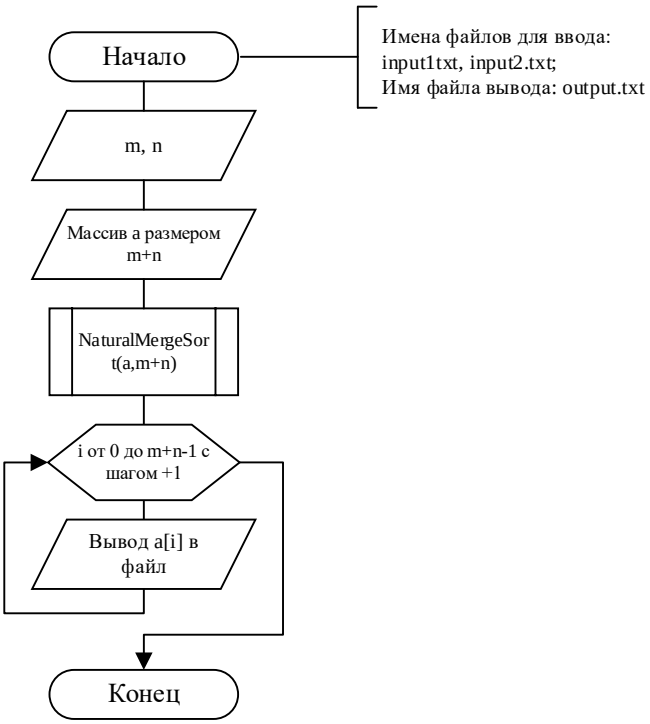

Рисунок 1 – Схема алгоритма сортировки слиянием по возрастанию двух файлов

Схема алгоритма слияния двух упорядоченных по возрастанию массивов, который является основной сортировки естественного двух путевого слияния, представлена на рисунке 2.

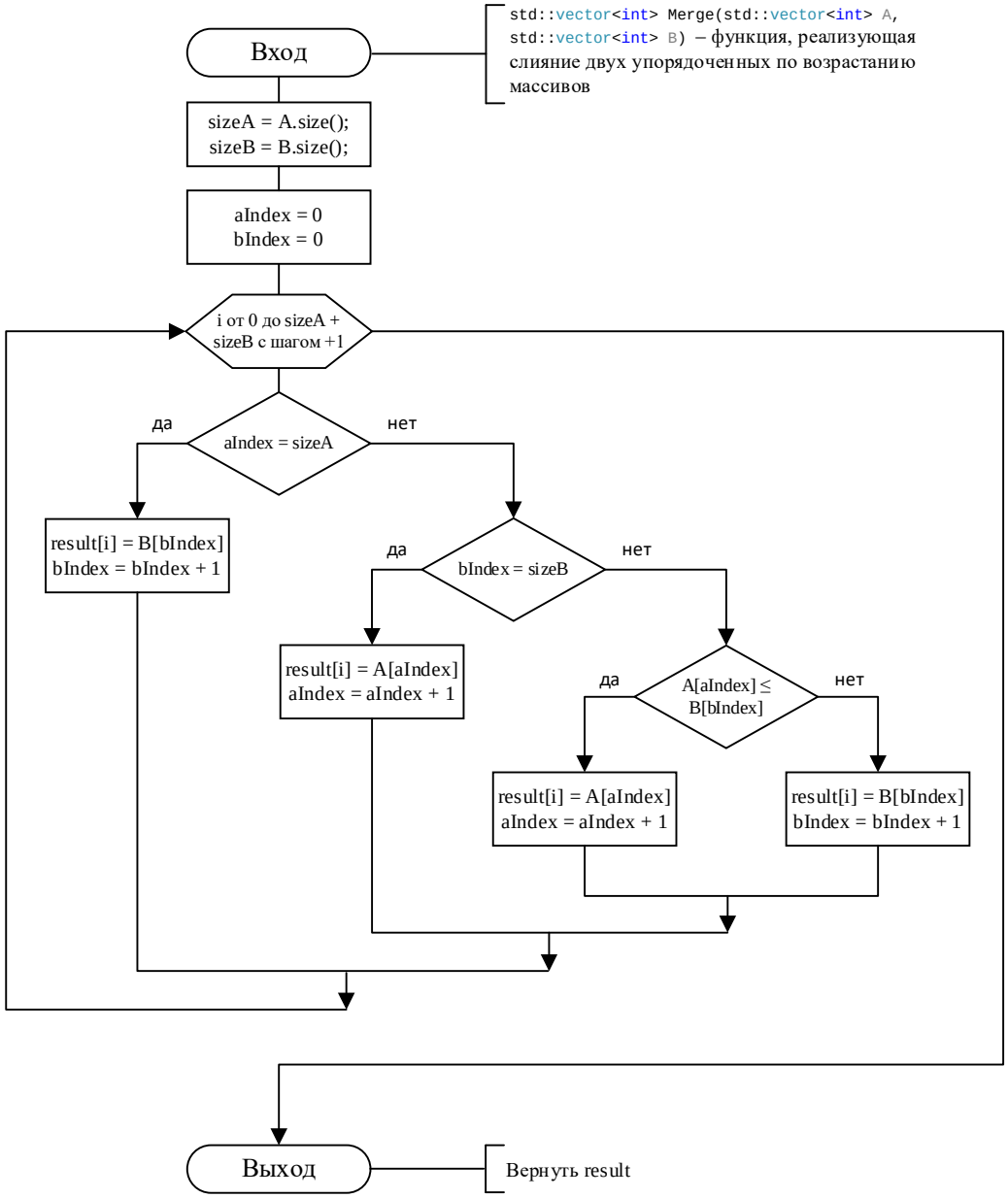

Рисунок 2 – Схема алгоритма слияния двух массивов

Схема алгоритма поиска длин отрезков возрастания на обоих концах последовательности представлена на рисунке 3.

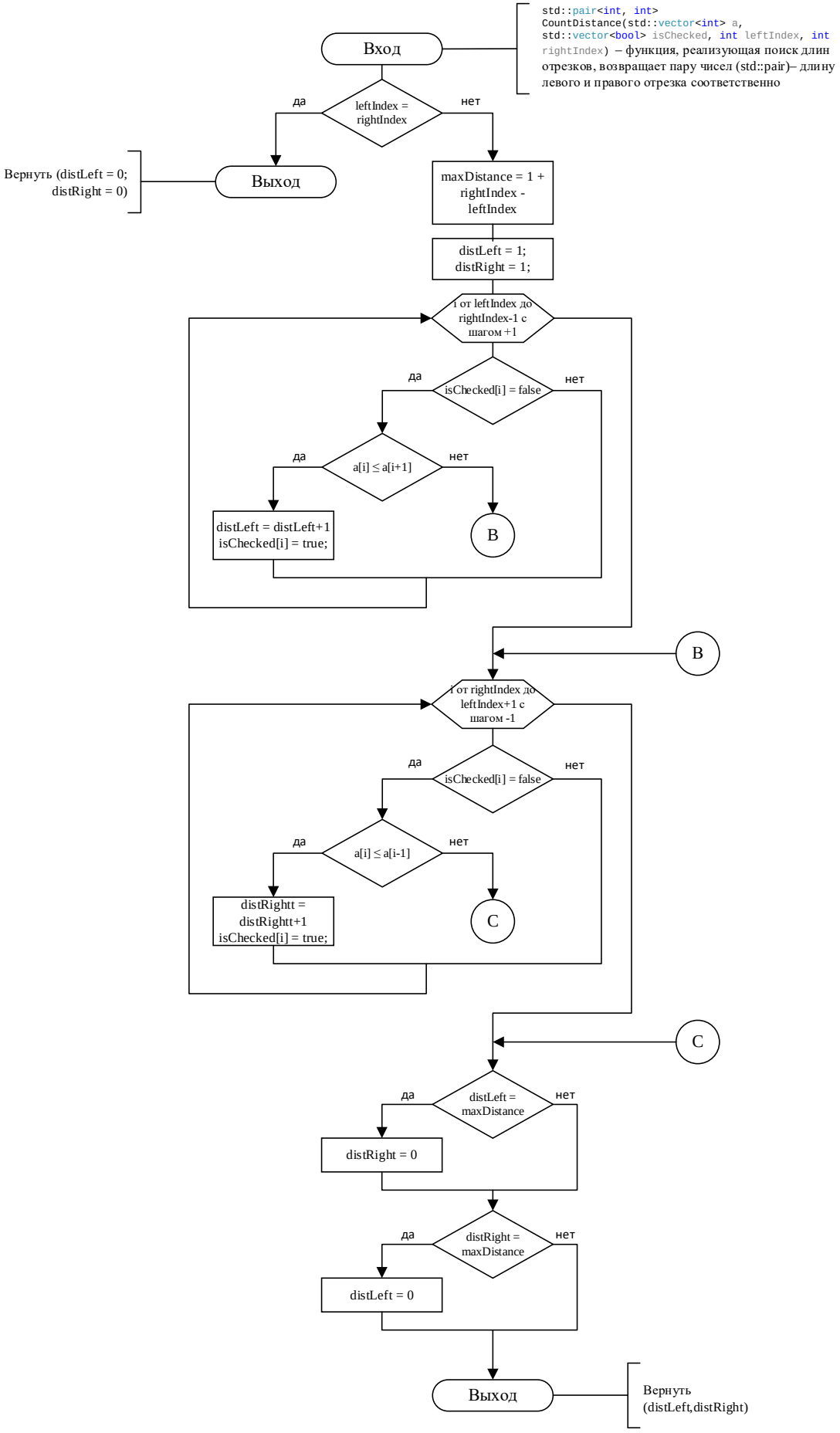

Рисунок 3 – Схема алгоритма поиска длин отрезков возрастания

Схема непосредственно самого алгоритма сортировки с использованием метода естественного двух путевого слияния, представлена на рисунке 4.

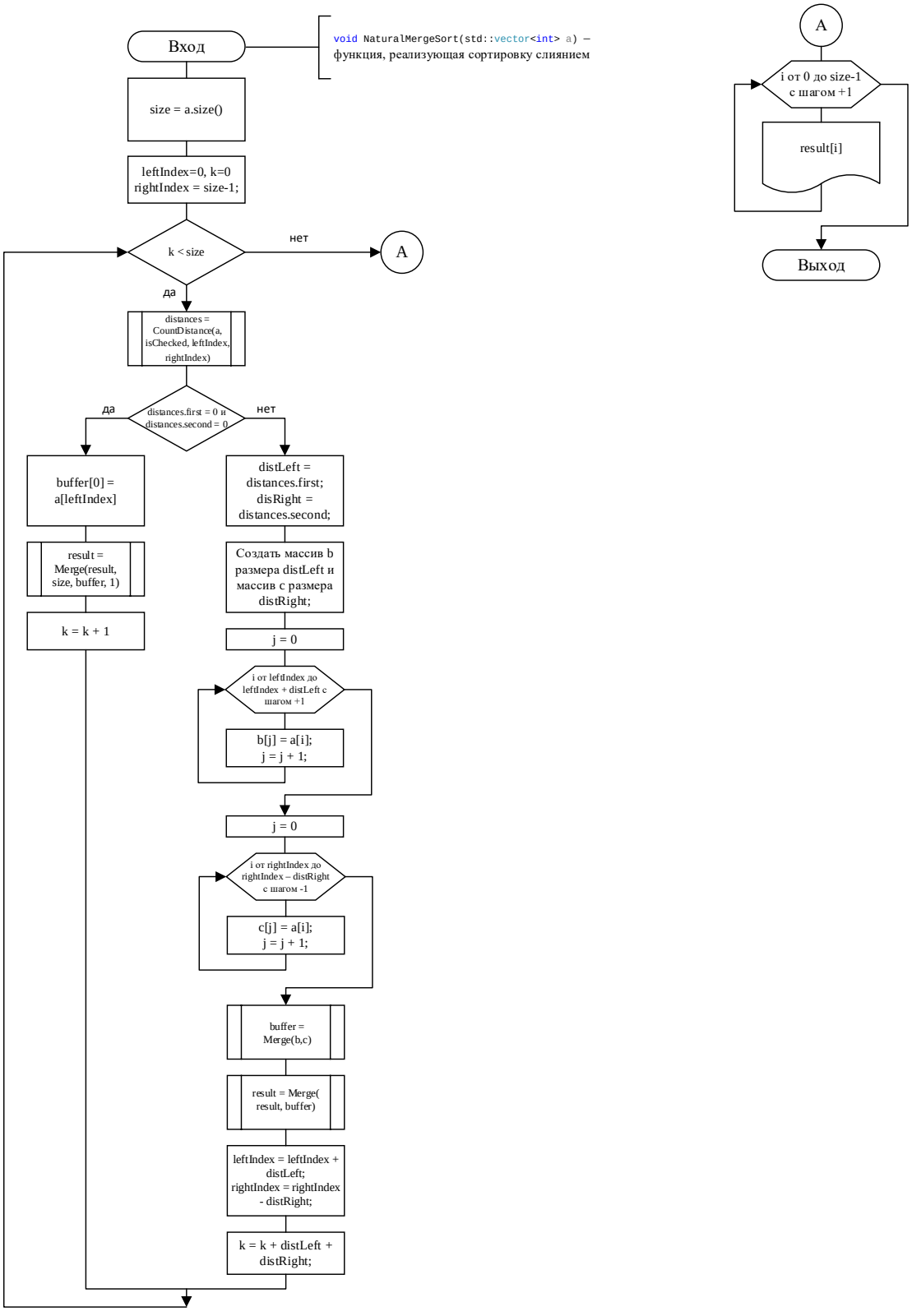

Рисунок 4 – Схема алгоритма сортировки слиянием по возрастанию двух файлов

ТЕКСТ ПРОГРАММЫ

Текст программы на языке программирования C++ для сортировки слиянием по возрастанию двух файлов с использованием метода естественного двух путевого слияния представлен в листинге 1.

Листинг 1. Текст программы

```
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <fstream>
#include <vector>
std::vector<int> Merge(std::vector<int> A, std::vector<int> B)
{
    int sizeA = A.size();
    int sizeB = B.size();
    int aIndex = 0, bIndex = 0;
    std::vector<int> result;
    for (int i = 0; i < sizeA + sizeB; i++)
    {
        if (aIndex == sizeA)
        {
            result.push_back(B[bIndex]);
            bIndex++;
        }
        else
        {
            if (bIndex == sizeB)
            {
                result.push_back(A[aIndex]);
                aIndex++;
            }
            else
            {
                if (A[aIndex] <= B[bIndex])
                {
                    result.push_back(A[aIndex]);
                    aIndex++;
                }
                else
                {
                    result.push_back(B[bIndex]);
                        bIndex++;
                }
            }
        }
    }
```

Листинг 1. Текст программы (продолжение)

```
    return result;
}
std::pair<int, int> CountDistance(std::vector<int> a,
std::vector<bool> isChecked, int leftIndex, int rightIndex)
{
    if (leftIndex == rightIndex)
        return std::make_pair(0, 0);
    int maxDistance = 1 + rightIndex - leftIndex;
    int distLeft = 1, distRight = 1;
    for (int i = leftIndex; i < rightIndex; i++) //поиск длины
левого отрезка
    {
        if (!isChecked[i])
        {
            if (a[i] <= a[i + 1])
            {
                distLeft++;
                isChecked[i] = true;
            }
            else break;
        }
    }
    for (int i = rightIndex; i > leftIndex; i--) //поиск длины
правого отрезка
    {
        if (!isChecked[i])
        {
            if (a[i] <= a[i - 1])
            {
                distRight++;
                isChecked[i] = true;
            }
            else break;
        }
    }
    if (distLeft == maxDistance) distRight = 0;
    if (distRight == maxDistance) distLeft = 0;
    return std::make_pair(distLeft, distRight);
}
void NaturalMergeSort(std::vector<int> a)
{
    int size = a.size();
    std::vector<int> buffer;
    std::vector<int> result;
    std::vector<bool> isChecked(size, false);
    int leftIndex = 0;
    int rightIndex = size - 1;
    int distLeft, distRight, k = 0;
    while (k < size)
    {
        auto distances = CountDistance(a, isChecked,
leftIndex, rightIndex);
```

Листинг 1. Текст программы (продолжение)

```
        if (distances.first == 0 && distances.second == 0)
        {
            buffer.resize(1, 0);
            buffer[0] = a[leftIndex];
            result = Merge(result, buffer);
            k++;
        }
        else
        {
            distLeft = distances.first;
            distRight = distances.second;
            std::vector<int> b (distLeft, -1e6);
            int j = 0;
            for (int i = leftIndex; i < leftIndex + distLeft;
i++)
            {
                b[j] = a[i];
                j++;
            }
            j = 0;
            std::vector<int> c(distRight, -1e6);
            for (int i = rightIndex; i > rightIndex -
distRight; i--)
            {
                c[j] = a[i];
                j++;
            }
            buffer.resize(distRight + distLeft);
            buffer = Merge(b, c);
            result = Merge(result, buffer);
            leftIndex += distLeft;
            rightIndex -= distRight;
            k += distLeft + distRight;
        }
    }
    for (size_t i = 0; i < size; i++)
    {
        std::cout << result[i] << std::setw(5);
    }
    return;
}
int main()
{
    setlocale(LC_ALL, "RUSSIAN");
    srand((unsigned)time(NULL));
    std::ofstream input1("input1.txt");
    std::ifstream input("input.txt");
    std::ofstream input2("input2.txt");
    std::ofstream output("output.txt");
    int n, m;
    std::cout << "\t\tВведите количество чисел для первого
файла: ";
```

Листинг 1. Текст программы (продолжение)

```
    std::cin >> n;
    std::cout << "\t\tВведите количество чисел для второго
файла: ";
    std::cin >> m;
    std::vector <int> a;
    std::cout << "\t\tПоследовательность до
сортировки:\n\n\t\t";
    for (int i = 0; i < n; i++)
    {
        a.push_back(rand() % 100 - 30);
        std::cout << a[i] << std::setw(5);
        input1 << a[i] << " ";
    }
    for (int i = n; i < n + m; i++)
    {
        a.push_back(rand() % 10);
        std::cout << a[i] << std::setw(5);
        input2 << a[i] << " ";
    }
    std::cout << "\n\n";
    std::cout << "\t\tСодержимое файлов после
сортировки:\n\n\t\t";
    NaturalMergeSort(a);
    for (size_t i = 0; i < m + n; i++)
        output << a[i] << " ";
    input1.close();
    input2.close();
    output.close();
    return 0;}
```

ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Данная программа предназначена для сортировки слиянием по возрастанию двух файлов разного размера с использованием метода естественного двух путевого слияния. Пользователю предлагается ввести количество случайных числовых значений для первого и второго файла соответственно. После происходит сортировка, и результат слияния выводится в файл и на экран.

ИНСТРУКЦИЯ ПРОГРАММИСТА

Данная программа заполняет два файла случайными числами и сортирует слиянием по возрастанию. Структуры данных, используемые в программе, приведены в таблице 1.

Таблица 1 – Структуры данных в программе

Имя	Тип (класс)	Предназначение
input1, input2, output	file	Имена файлов с входными и выходными данными соответственно
n, m	int	Длины последовательностей в файлах
a	int*	Общая последовательность для сортировки

В программе имеются следующие функции:

1) `void NaturalMergeSort(std::vector<int> a)` – функция, непосредственно реализующая сортировку.

Таблица 2 – Структуры данных, используемые в подпрограмме `NaturalMergeSort`

Имя	Тип	Предназначение
		формальные параметры
a	std::vector<int>	Массив для сортировки
		локальные переменные
buffer	int*	Буфер для хранения упорядоченных подмассивов
result	Int*	Массив для хранения конечной последовательности
isChecked	std::vector<bool>	Массив флагов, сообщающих о том, рассмотрен элемент или ещё нет
leftIndex	int	Индекс левого конца рабочей области
rightIndex	int	Индекс правого конца рабочей области
distLeft	int	Длина отрезка слева
distRight	int	Длина отрезка справа
k	int	Счётчик отсортированных элементов
j	int	Локальный счётчик элементов
size	int	Размер массива a

2) `std::pair<int, int> CountDistance(std::vector<int> a, std::vector<bool> isChecked, int leftIndex, int rightIndex)` – функция для подсчёта длин отрезков.

Таблица 3 – Структуры данных, используемые в подпрограмме `CountDistance`

Имя	Тип	Предназначение
		формальные параметры
a	std::vector<int>	Массив для подсчёта
isChecked	std::vector<bool>	Массив флагов
leftIndex	int	Левая граница области поиска
rightIndex	int	Правая граница области поиска
		локальные переменные
maxDistance	int	Максимально возможная длина отрезка
distLeft	int	Длина отрезка слева
distRight	int	Длина отрезка справа

3) `std::vector <int> Merge(std::vector <int> A, std::vector <int> B)` – функция для слияния массивов.

Таблица 4 – Структуры данных, используемые в подпрограмме `Merge`

Имя	Тип	Предназначение
		формальные параметры
A	std::vector <int>	Первый массив
B	std::vector <int>	Второй массив
		локальные переменные
aIndex	int	Индекс текущего элемента в первом массиве
bIndex	int	Индекс текущего элемента во втором массиве
result	std::vector <int>	Итоговый массив
sizeA	int	Размер первого массива
sizeB	int	Размер второго массива

ДЕМОНСТРАЦИОННЫЙ ПРИМЕР

Сперва отсортируем входные данные вручную. Имеется два файла разного размера, содержащие случайные числовые значения. Необходимо отсортировать последовательность чисел, полученную из содержимого этих файлов, двух путевым естественным слиянием.

Пусть в каждом файле содержится по 6 чисел. В первом файле содержится последовательность 7 9 13 1 8 4, а во втором – 10 11 5 3 6 2. Тогда рассмотрим такую последовательность:

7	9	13	1	8	4	10	11	5	3	6	2

1) Слева имеется возрастающий участок 7 9 13, а справа, если читать справа налево (то есть рассматривая отрезки от края к центру), есть участок 2 6. Слияние этих двух фрагментов дает отрезок 2 6 7 9 13.
2) Слиянием следующих двух неубывающих отрезков 1 8 и 3 5 11 получим неубывающую последовательность 1 3 5 8 11.
3) Полученную пару отрезков объединим в неубывающую последовательность 1 2 3 4 5 6 7 8 11 13. Теперь осталось рассмотреть последние два участка – 10 и 4. Из них получаем 4 10.
4) Последнее слияние дает 1 2 3 4 5 6 7 8 9 10 11 13. Последовательность отсортирована.

Теперь проверим результат работы программы на соответствие полученной последовательности. Он представлен на рисунке 5. Как видно по рисунку, последовательность отсортирована верно.

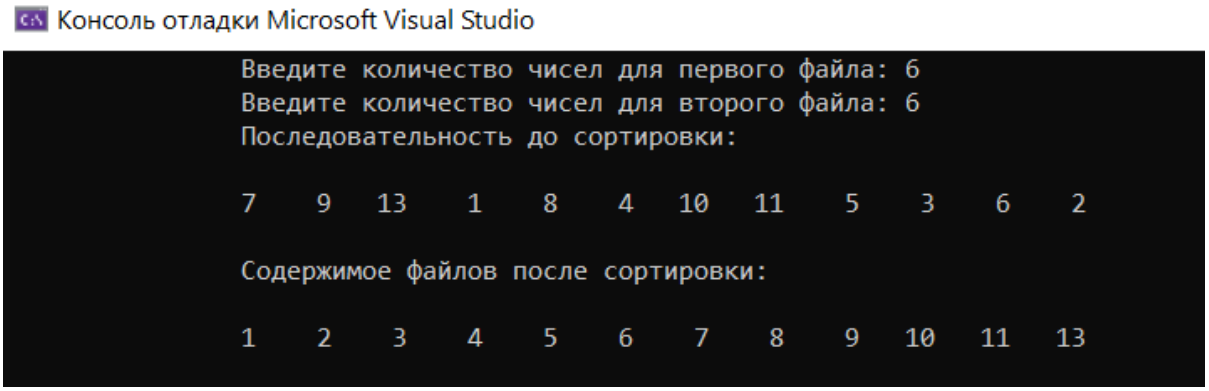

Рисунок 5 – Результат работы программы

Как видно на рисунке 6, данные в выходном файле также соответствуют ожидаемому результату сортировки.

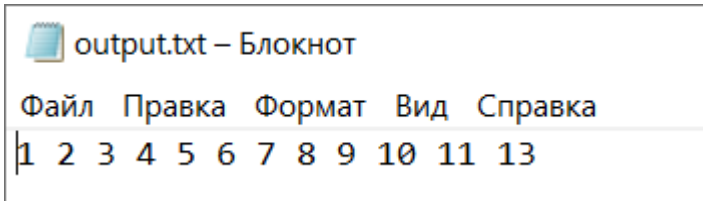

Рисунок 6 – Выходные данные, содержащиеся в файле output.txt

ВЫВОДЫ

В ходе данной лабораторной работы был изучен принцип работы сортировки слиянием по возрастанию двух файлов разного размера с использованием метода естественного двух путевого слияния. Такой подход обладает тем возможным преимуществом, что исходные файлы с преобладанием возрастающего или убывающего расположения элементов могут обрабатываться очень быстро, но при этом замедляется основной цикл вычислений.

Для демонстрации полученных знаний была написана программа для сортировки указанным методом, результат работы которой был проверен аналитически. По результатам проверки можно сделать вывод о том, что программа работает корректно.