Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Алгоритмы обработки последовательностей
Тема

Преподаватель _____ Р. Ю. Царев
подпись, дата | инициалы, фамилия

Студент КИ19-17/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки | подпись, дата | инициалы, фамилия

Красноярск 2020

1 Цель работы

Изучение некоторых алгоритмов обработки последовательностей.

2 Задачи

Написать программу, реализующую алгоритм внешней сортировки простым слиянием.

Предъявлены следующие требования к выполнению работы.

1. Строгое соответствие программы и результатов ее работы с полученным заданием.

2. Самостоятельные тестирование и отладка программы.

3. Устойчивость работы программы при любых воздействиях, задаваемых пользователем через интерфейс программы.

4. Предоставление демонстрационного примера и исходного текста программы для защиты.

5. Предоставление отчета по практическому заданию, содержащего описание реализованного алгоритма, программы, результатов работы программы (отчет необходимо загрузить на сайт курса).

6. При сортировке последовательностей должны использоваться файлы, но не массивы.

3 Описание реализованного алгоритма

Реализован алгоритм сортировки простым слиянием. При сортировке используются файлы.

4 Описание программы

Для решения задачи была написана программа на языке C. Программа берет данные из файла origin.txt и сохраняет результат работы там же, используя вспомогательные файлы a.txt и b.txt . В файле origin.txt должны содержаться целые числа, разделенные символом переноса строки.

Листинг 1 – Код программы

```
#include <stdio.h>
#include <stdbool.h>

bool SplitSequence(FILE* origin, FILE* a, FILE* b, int step)
{
    int i;
    i = 0;
    int current;

    while (fscanf(origin, "%d", &current) != EOF)
    {
        if (!((i / step) % 2))
        {
            fprintf(a, "%d\n", current);
        }
        else
        {
            fprintf(b, "%d\n", current);
        }
        i++;
    }
    return i > step;
}

void MergeSequences(FILE* a, FILE* b, FILE* result, int step)
{
    FILE* switchFile;
    switchFile = NULL;
    int tempA;
    int tempB;
    int next;
    int aPassed;
    int bPassed;
    bool aFinish;
    bool bFinish;
    int lastScan;
    aFinish = false;
    bFinish = false;
    aPassed = -1;
    bPassed = -1;
```

Продолжение листинга 1

```
while (!aFinish || !bFinish)
{
    if (!aFinish && switchFile != a)
    {
        lastScan = fscanf(a, "%d", &tempA);
        if (lastScan == EOF)
        {
            aFinish = true;
        }
        aPassed++;
    }
    if (!bFinish && switchFile != b)
    {
        lastScan = fscanf(b, "%d", &tempB);
        if (lastScan == EOF)
        {
            bFinish = true;
        }
        bPassed++;
    }
    if (aFinish && bFinish)
    {
        break;
    }
    if (aFinish)
    {
        fprintf(result, "%d\n", tempB);
        switchFile = NULL;
        continue;
    }
    if (bFinish)
    {
        fprintf(result, "%d\n", tempA);
        switchFile = NULL;
        continue;
    }
    if (aPassed / step == bPassed / step)
    {
        if (tempA > tempB)
        {
            next = tempA;
```

Продолжение листинга 1

```
                switchFile = b;
            }
            else
            {
                next = tempB;
                switchFile = a;
            }
        }
        else
        {
            if (aPassed > bPassed)
            {
                next = tempB;
                switchFile = a;
            }
            else
            {
                next = tempA;
                switchFile = b;
            }
        }
        fprintf(result, "%d\n", next);
    }
}

int main()
{
    int step = 1;
    int intToPrint;
    bool notSorted;
    FILE* originFile = fopen("origin.txt", "r");
    printf("Origin set:\n");
    while (fscanf(originFile, "%d", &intToPrint) != EOF)
    {
        printf("%d ", intToPrint);
    }
    fclose(originFile);

    do
    {
        originFile = fopen("origin.txt", "r");
```

Продолжение листинга 1

```
        FILE* aFile = fopen("a.txt", "w");
        FILE* bFile = fopen("b.txt", "w");

        notSorted = SplitSequence(originFile, aFile, bFile, step);

        fclose(originFile);
        fclose(aFile);
        fclose(bFile);

        aFile = fopen("a.txt", "r");
        bFile = fopen("b.txt", "r");

        FILE* destFile = fopen("origin.txt", "w");

        MergeSequences(aFile, bFile, destFile, step);

        fclose(aFile);
        fclose(bFile);
        fclose(destFile);

        step *= 2;
    } while (notSorted);
    printf("\nSorted set:\n");
    originFile = fopen("origin.txt", "r");
    while (fscanf(originFile, "%d", &intToPrint) != EOF)
    {
        printf("%d ", intToPrint);
    }
    fclose(originFile);
    return 0;
}#include <stdio.h>
#include <stdbool.h>

bool SplitSequence(FILE* origin, FILE* a, FILE* b, int step)
{
    int i;
    i = 0;
    int current;

    while (fscanf(origin, "%d", &current) != EOF)
    {
```

Продолжение листинга 1

```
        if (!((i / step) % 2))
        {
            fprintf(a, "%d\n", current);
        }
        else
        {
            fprintf(b, "%d\n", current);
        }
        i++;
    }
    return i > step;
}

void MergeSequences(FILE* a, FILE* b, FILE* result, int step)
{
    FILE* switchFile;
    switchFile = NULL;
    int tempA;
    int tempB;
    int next;
    int aPassed;
    int bPassed;
    bool aFinish;
    bool bFinish;
    int lastScan;
    aFinish = false;
    bFinish = false;
    aPassed = -1;
    bPassed = -1;

    while (!aFinish || !bFinish)
    {
        if (!aFinish && switchFile != a)
        {
            lastScan = fscanf(a, "%d", &tempA);
            if (lastScan == EOF)
            {
                aFinish = true;
            }
            aPassed++;
        }
```

Продолжение листинга 1

```
        if (!bFinish && switchFile != b)
        {
            lastScan = fscanf(b, "%d", &tempB);
            if (lastScan == EOF)
            {
                bFinish = true;
            }
            bPassed++;
        }
        if (aFinish && bFinish)
        {
            break;
        }
        if (aFinish)
        {
            fprintf(result, "%d\n", tempB);
            switchFile = NULL;
            continue;
        }
        if (bFinish)
        {
            fprintf(result, "%d\n", tempA);
            switchFile = NULL;
            continue;
        }
        if (aPassed / step == bPassed / step)
        {
            if (tempA > tempB)
            {
                next = tempA;
                switchFile = b;
            }
            else
            {
                next = tempB;
                switchFile = a;
            }
        }
        else
        {
            if (aPassed > bPassed)
```

Продолжение листинга 1

```
            {
                next = tempB;
                switchFile = a;
            }
            else
            {
                next = tempA;
                switchFile = b;
            }
        }
        fprintf(result, "%d\n", next);
    }
}

int main()
{
    int step = 1;
    int intToPrint;
    bool notSorted;
    FILE* originFile = fopen("origin.txt", "r");
    printf("Origin set:\n");
    while (fscanf(originFile, "%d", &intToPrint) != EOF)
    {
        printf("%d ", intToPrint);
    }
    fclose(originFile);

    do
    {
        originFile = fopen("origin.txt", "r");
        FILE* aFile = fopen("a.txt", "w");
        FILE* bFile = fopen("b.txt", "w");

        notSorted = SplitSequence(originFile, aFile, bFile, step);

        fclose(originFile);
        fclose(aFile);
        fclose(bFile);

        aFile = fopen("a.txt", "r");
        bFile = fopen("b.txt", "r");
```

Окончание листинга 1

```
        FILE* destFile = fopen("origin.txt", "w");

        MergeSequences(aFile, bFile, destFile, step);

        fclose(aFile);
        fclose(bFile);
        fclose(destFile);

        step *= 2;
    } while (notSorted);
    printf("\nSorted set:\n");
    originFile = fopen("origin.txt", "r");
    while (fscanf(originFile, "%d", &intToPrint) != EOF)
    {
        printf("%d ", intToPrint);
    }
    fclose(originFile);
    return 0;
}
```

5 Результаты работы программы

На следующем рисунке приведен скриншот с результатами работы программы.

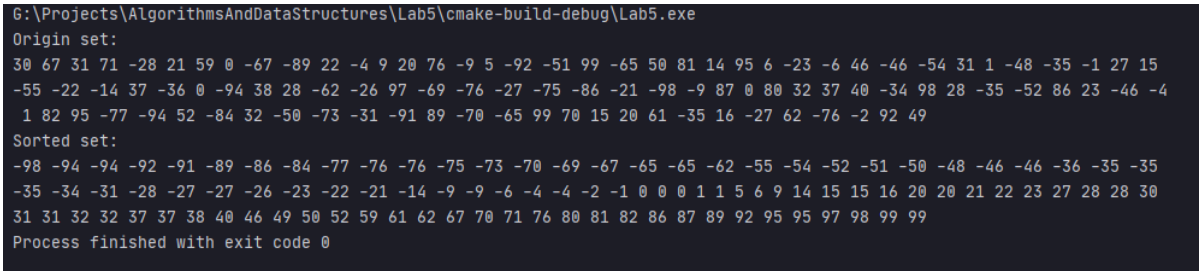

Рисунок 1 – Результаты работы программы