Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

Лабораторная работа №5. Алгоритмы обработки последовательностей

тема

Преподаватель \_\_\_\_\_\_\_\_\_ Р.Ю. Царёв

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 \_\_\_\_\_\_\_\_\_ Е.В. Железкин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

Содержание

[Содержание 2](#_Toc33628970)

[1 Цель работы 3](#_Toc33628971)

[2 Задача работы 3](#_Toc33628972)

[3 Ход работы 3](#_Toc33628973)

[4 Вывод 10](#_Toc33628974)

# Цель работы

Изучение алгоритмов обработки последовательностей.

# Задача работы

Реализовать в программе один из алгоритмов (по выбору студента) из указанных ниже:

а) простого слияния,

б) естественного слияния,

в) многофазного слияния,

г) много путевого слияния.

Важно! При сортировке последовательностей должны использоваться файлы, но не массивы.

# Ход работы

Листинг 1 – содержание файла main.cpp проекта Lab\_5:

#include **<iostream>**#include **<string>**#include **"MultywaySort.h"  
  
using namespace** std;  
  
**int** main() {  
  
 cout << **"Введите имя сортируемого файла: \n"**;  
  
 string str = **"0"**;  
 *// getline(cin, str);* ofstream outfile(str, ios::binary | ios::out);  
 **int** temp = 0;  
 srand(time(**NULL**));  
 **if** (outfile)  
 {  
 **for** (**int** i = 1; i < **COUNT\_OF\_NUMBERS**; i++) {  
 temp = rand() % **MAX\_RANGE**;  
 outfile.write((**char** \*) &temp, **sizeof**(temp));  
 cout << temp << **" "**;  
 }  
 }  
  
 outfile.close();  
  
 MultywaySort(str);  
  
 FILE\* streamIn;  
 FILE\* streamOut;  
 **char** str2[2];  
 str2[1] = **'\0'**;  
 str2[0] = **char**(2 + **'0'**);  
 streamIn = fopen(str2, **"rd"**);  
 fseek(streamIn, 0, **SEEK\_END**);  
 **int** size = ftell(streamIn);  
 fclose(streamIn);  
  
 **if** (size == 0){  
 streamIn = fopen(**"1"**, **"rb"**);  
 streamOut = fopen(**"Out"**, **"wb"**);  
 **while** (fread(&temp, 1, **sizeof**(**int**), streamIn)){  
 fwrite(&temp, 1, **sizeof**(**int**), streamOut);  
 }  
 fclose(streamIn);  
 fclose(streamOut);  
 } **else** {  
 str2[0] = **char**(**COUNT\_OF\_CHANNELS** + **'0'** + 1);  
 streamIn = fopen(str2, **"rb"**);  
 streamOut = fopen(**"Out"**, **"wb"**);  
 **while** (fread(&temp, 1, **sizeof**(**int**), streamIn)){  
 fwrite(&temp, 1, **sizeof**(**int**), streamOut);  
 }  
 fclose(streamIn);  
 fclose(streamOut);  
 }  
  
  
  
 **return** 0;  
}

Листинг 2 – содержание файла MultywaySort.cpp проекта Lab\_5:

#include **"MultywaySort.h"  
  
using namespace** std;  
  
**void** MultywaySort(**const** string& Name){  
  
 **int** temp = 0;  
 **char** str[2];  
 str[1] = **'\0'**;  
 **int** last = **MAX**;  
 **int** i = 0;  
  
 ifstream in(Name, ios::binary | ios::in);  
 FILE \*stream = **nullptr**;  
  
 i = -1;  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 str[0] = **char**(j % **COUNT\_OF\_CHANNELS** + 1 + **'0'**);  
 stream = fopen(str, **"wb"**);  
 fclose(stream);  
 }  
  
 **while** (in.read((**char**\*) &temp, **sizeof**(temp))) {  
  
 **if** (last > temp) {  
 i++;  
 str[0] = **char**(i % **COUNT\_OF\_CHANNELS** + 1 + **'0'**);  
 fclose(stream);  
 stream = fopen(str, **"ab+"**);  
 fseek(stream, 0, **SEEK\_END**);  
 }  
  
 last = temp;  
 fwrite(&temp, 1, **sizeof**(**int**), stream);  
  
 }  
  
 fclose(stream);  
 in.close();  
  
 i = 0;  
 **int** pool;  
 **int** size = 0;  
  
 **while**(**true**){  
  
 pool = 0;  
 sortHelper(i % 2);  
 i++;  
  
 **for** (**int** j = 1; j < **COUNT\_OF\_CHANNELS**; j++){  
 str[0] = **char**(j + 1 + **'0'** + **COUNT\_OF\_CHANNELS** \* (i % 2));  
 stream = fopen(str, **"rb"**);  
 fseek(stream, 0, **SEEK\_END**);  
 size = ftell(stream);  
 **if** (size != 0) pool++;  
 fclose(stream);  
 }  
  
 stream = fopen(**"1"**, **"rb"**);  
 cout << **"\n1) "**;  
 **while** (fread(&temp, 1, **sizeof**(**int**), stream)){  
 cout << temp << **" "**;  
 }  
 fclose(stream);  
  
 stream = fopen(**"2"**, **"rb"**);  
 cout << **"\n2) "**;  
 **while** (fread(&temp, 1, **sizeof**(**int**), stream)){  
 cout << temp << **" "**;  
 }  
 fclose(stream);  
  
 stream = fopen(**"3"**, **"rb"**);  
 cout << **"\n3) "**;  
 **while** (fread(&temp, 1, **sizeof**(**int**), stream)){  
 cout << temp << **" "**;  
 }  
 fclose(stream);  
  
 stream = fopen(**"4"**, **"rb"**);  
 fseek(stream, 0, **SEEK\_SET**);  
 cout << **"\n\n4) "**;  
 **while** (fread(&temp, 1, **sizeof**(**int**), stream)){  
 cout << temp << **" "**;  
 }  
 fclose(stream);  
  
 stream = fopen(**"5"**, **"rb"**);  
 fseek(stream, 0, **SEEK\_SET**);  
 cout << **"\n5) "**;  
 **while** (fread(&temp, 1, **sizeof**(**int**), stream)){  
 cout << temp << **" "**;  
 }  
 fclose(stream);  
  
 stream = fopen(**"6"**, **"rb"**);  
 fseek(stream, 0, **SEEK\_SET**);  
 cout << **"\n6) "**;  
 **while** (fread(&temp, 1, **sizeof**(**int**), stream)){  
 cout << temp << **" "**;  
 }  
 fclose(stream);  
 cout << endl;  
  
 **for** (**int** j = 0; j < 20; j++)  
 cout << **"-----"**;  
  
 **if** (pool == 0) **return**;  
  
 }  
  
}  
  
**void** sortHelper(**bool** flag){  
  
 **if** (!flag){  
  
 **char** str[2];  
 str[1] = **'\0'**;  
  
 FILE \*stream = **nullptr**;  
  
 **int** i = 0;  
  
 **int** temp[**COUNT\_OF\_CHANNELS**];  
 **int** indents[**COUNT\_OF\_CHANNELS**];  
 **for** (**int** & indent : indents)  
 indent = 0;  
  
 **int** last = **MAX**;  
 **int** pool = 0;  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 str[0] = **char**(j + 1 + **'0'** + **COUNT\_OF\_CHANNELS**);  
 stream = fopen(str, **"wb"**);  
 fclose(stream);  
 }  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 str[0] = **char**(j + 1 + **'0'**);  
 stream = fopen(str, **"rb"**);  
 fseek(stream, 4 \* indents[j], **SEEK\_SET**);  
 **if** (fread(&temp[j], 1, **sizeof**(**int**), stream)) pool++;  
 fclose(stream);  
 }  
  
 **if** (pool == 0) **return**;  
  
 **int** min;  
 **int** pos = 0;  
 **int** size = 0;  
  
 **while** (**true**) {  
  
 pool = 0;  
 min = **MAX**;  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++)  
 {  
 **if** (min > temp[j]) {  
 min = temp[j];  
 pos = j;  
 }  
 }  
  
 **if** (last > temp[pos] && last != **MAX**) {  
 i++;  
 }  
  
 str[0] = **char**(i % **COUNT\_OF\_CHANNELS** + 1 + **'0'** + **COUNT\_OF\_CHANNELS**);  
 stream = fopen(str, **"ab+"**);  
 fseek(stream, 0, **SEEK\_END**);  
 last = temp[pos];  
 fwrite(&last, 1, **sizeof**(**int**), stream);  
 fclose(stream);  
  
 str[0] = **char**(pos + 1 + **'0'**);  
 stream = fopen(str, **"rb"**);  
 indents[pos]++;  
 fseek(stream, 4 \* indents[pos], **SEEK\_SET**);  
 fread(&temp[pos], 1, **sizeof**(**int**), stream);  
 fseek(stream, 0, **SEEK\_END**);  
 size = ftell(stream);  
  
 **if** (size / 4 <= indents[pos]) {  
 temp[pos] = **MAX**;  
 last = temp[pos];  
 }  
 fclose(stream);  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 **if** (temp[j] != **MAX**) pool++;  
 }  
  
 **if** (pool == 0) **break**;  
  
 }  
  
  
 } **else** {  
  
 **char** str[2];  
 str[1] = **'\0'**;  
  
 FILE \*stream = **nullptr**;  
 *//FILE \*streamOut = nullptr;* **int** i = 0;  
  
 **int** temp[**COUNT\_OF\_CHANNELS**];  
 **int** indents[**COUNT\_OF\_CHANNELS**];  
 **for** (**int** & indent : indents)  
 indent = 0;  
  
 **int** last = **MAX**;  
 **int** pool = 0;  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 str[0] = **char**(j + 1 + **'0'**);  
 stream = fopen(str, **"wb"**);  
 fclose(stream);  
 }  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 str[0] = **char**(j + 1 + **'0'** + **COUNT\_OF\_CHANNELS**);  
 stream = fopen(str, **"rb"**);  
 fseek(stream, 4 \* indents[j], **SEEK\_SET**);  
 **if** (fread(&temp[j], 1, **sizeof**(**int**), stream)) pool++;  
 fclose(stream);  
 }  
  
 **if** (pool == 0) **return**;  
  
 **int** min;  
 **int** pos = 0;  
 **int** size = 0;  
  
 **while** (**true**) {  
  
 pool = 0;  
 min = **MAX**;  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++)  
 {  
 **if** (min > temp[j]) {  
 min = temp[j];  
 pos = j;  
 }  
 }  
  
 **if** (last > temp[pos] && last != **MAX**) {  
 i++;  
 }  
  
 str[0] = **char**(i % **COUNT\_OF\_CHANNELS** + 1 + **'0'**);  
 stream = fopen(str, **"ab+"**);  
 fseek(stream, 0, **SEEK\_END**);  
 last = temp[pos];  
 fwrite(&last, 1, **sizeof**(**int**), stream);  
 fclose(stream);  
  
 str[0] = **char**(pos + 1 + **'0'** + **COUNT\_OF\_CHANNELS**);  
 stream = fopen(str, **"rb"**);  
 indents[pos]++;  
 fseek(stream, 4 \* indents[pos], **SEEK\_SET**);  
 fread(&temp[pos], 1, **sizeof**(**int**), stream);  
 fseek(stream, 0, **SEEK\_END**);  
 size = ftell(stream);  
  
 **if** (size / 4 <= indents[pos]) {  
 temp[pos] = **MAX**;  
 last = temp[pos];  
 }  
 fclose(stream);  
  
 **for** (**int** j = 0; j < **COUNT\_OF\_CHANNELS**; j++){  
 **if** (temp[j] != **MAX**) pool++;  
 }  
  
 **if** (pool == 0) **break**;  
  
 }  
  
 }  
  
  
}

Листинг 3 – содержание файла MultywaySort.h проекта Lab\_5:

#ifndef **LAB\_5\_MULTYWAYSORT\_H**#define **LAB\_5\_MULTYWAYSORT\_H**#define **MAX** 999999999  
#define **COUNT\_OF\_CHANNELS** 4  
#define **COUNT\_OF\_NUMBERS** 100  
#define **MAX\_RANGE** 1000  
  
#include **<string>**#include **<fstream>**#include **<cstdio>**#include **<cstdlib>**#include **<iostream>  
  
void** MultywaySort(**const** std::string&);  
**void** sortHelper(**bool**);  
  
#endif *//LAB\_5\_MULTYWAYSORT\_H*

1. Примеры работы:

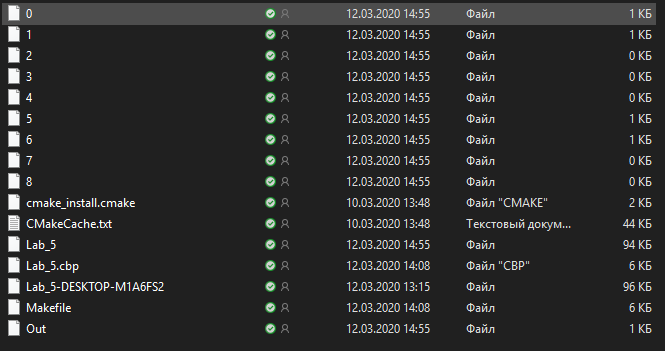


Рисунок 1 – результаты работы программы (исходный массив в файле «0»)

# Вывод

В ходе данной лабораторной работы были изучены и реализованы алгоритмы обработки последовательностей, описанные в задани.