Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра **«Информационные технологии и автоматизированные системы»**

направление подготовки: 09.03.04 - «Программная инженерия»

Лабораторная работа

По теме **«Внешние сортировки»**

Выполнял:

студент группы РИС-24-1б

Морозова Н.С.

Проверял:

Доцент кафедры ИТАС

Полякова О.А.

Постановка задачи:

1. Создать 3 массива из случайных чисел, вывести их.
2. Применить к каждому массиву определённую сортировку используя файлы.
3. Вывести отсортированные данные.

Анализ задачи:

1. Используя метод random заполнить массивы.
2. Для каждого метода сортировок создать функцию.
3. Переодически заносить данные в файлы.
4. Написать функцию show для вывода массива.

Решение

Код

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

#include <vector>

#include <algorithm>

#include <ctime>

using namespace std;

// Разделение на серии

void splitN(const string& input, const string& tape1, const string& tape2)

{

ifstream in(input);

ofstream out1(tape1), out2(tape2);

int current, next;

bool switchTape = false;

if (in >> current)

{

(switchTape ? out2 : out1) << current << " ";

while (in >> next) {

if (next < current)

switchTape = !switchTape;

(switchTape ? out2 : out1) << next << " ";

current = next;

}

}

in.close(); out1.close(); out2.close();

}

// Слияние двух файлов

void merge(const string& output, const string& tape1, const string& tape2)

{

ifstream in1(tape1), in2(tape2);

ofstream out(output);

int val1, val2;

bool hasVal1 = static\_cast<bool>(in1 >> val1), hasVal2 = static\_cast<bool>(in2 >> val2);

while (hasVal1 && hasVal2)

{

if (val1 <= val2) {

out << val1 << " ";

hasVal1 = !!(in1 >> val1);

}

else {

out << val2 << " ";

hasVal2 = !!(in2 >> val2);

}

}

while (hasVal1) { out << val1 << ' '; hasVal1 = static\_cast<bool>(in1 >> val1); }

while (hasVal2) { out << val2 << ' '; hasVal2 = static\_cast<bool>(in2 >> val2); }

in1.close(); in2.close(); out.close();

}

void write(const string& sourceFile, const string& destFile)

{

ifstream in(sourceFile);

ofstream out(destFile);

if (!in.is\_open()) {

cerr << "Ошибка открытия исходного файла " << sourceFile << endl;

return;

}

if (!out.is\_open()) {

cerr << "Ошибка создания файла " << destFile << endl;

return;

}

int k;

while (in >> k) {

out << k << ' ';

}

in.close();

out.close();

}

// Основная функция

void natur(const string& inputFile) {

string tape1 = "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\tape1.txt";

string tape2 = "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\tape2.txt";

string tempFile = "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\temp\_sorted.txt";

while (true)

{

splitN(inputFile, tape1, tape2);

ifstream check(tape2);

if (check.peek() == ifstream::traits\_type::eof())

{ // Если tape2 пуст

check.close();

write(tape1, "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_natur.txt");

break;

}

check.close();

merge(tempFile, tape1, tape2);

write(tempFile, inputFile);

}

// Удаляем временные файлы

remove(tape1.c\_str());

remove(tape2.c\_str());

remove(tempFile.c\_str());

}

const int max\_tap = 50;

// Слияние отсортированных файлов

void mergSort(const string& output, const string tapes[], int k)

{

ifstream\* streams = new ifstream[k];

int\* curValue = new int[k]; //текущие значения из каждого файла

bool\* isActive = new bool[k](); // Активен ли поток

int count = 0;

// Открываем файлы и читаем первые значения

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

{

streams[i].open(tapes[i]);

if (streams[i] >> curValue[i]) {

isActive[i] = true;

count++;

}

}

ofstream out(output);

while (count > 0)

{// Находим минимальное значение среди активных потоков

int minVal = INT\_MAX;

int minIndex = -1;

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

if (isActive[i] && curValue[i] < minVal) {

minVal = curValue[i];

minIndex = i;

}

}

out << minVal << " ";

// Читаем следующее значение из потока с минимальным элементом

if (!(streams[minIndex] >> curValue[minIndex])) {

isActive[minIndex] = false;

count--;

}

}

for (int i = 0; i < k; ++i) // Закрываем все потоки

streams[i].close();

delete[] streams;

delete[] curValue;

delete[] isActive;

}

// Основная функция

void balance(const string& inputFile, int k = 2)

{

if (k > max\_tap) {

cout << "Ошибка: k слишком большое (максимально " << max\_tap << ")\n";

return;

}

// Распределение данных по лентам

string tapes[max\_tap];

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

tapes[i] = "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\tape" + to\_string(i) + ".txt";

ifstream in(inputFile);

if (!in.is\_open()) {

cout << "Невозможно открыть файл" << inputFile << endl;

return;

}

ofstream outs[max\_tap];

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

outs[i].open(tapes[i]);

if (!outs[i].is\_open()) {

cout << "Невозможно открыть файл" << tapes[i] << endl;

return;

}

}

// Чтение и распределение данных

int val, idx = 0;

while (in >> val) {

outs[idx] << val << ' ';

outs[idx].flush(); // Сброс буфера

idx = (idx + 1) % k;

}

in.close();

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

outs[i].close();

// Слияние

mergSort("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_balance.txt", tapes, k);

for (int i = 0; i < k; ++i) // Удаление временных файлов

remove(tapes[i].c\_str());

}

// Распределение серий по лентам с оптимизацией Фибоначчи

void distrub(const string& input, const string\* tapes, int k)

{

ifstream in(input);

if (!in.is\_open()) return;

// Генерация чисел Фибоначчи

int\* fib = new int[k + 1] {0, 1};

for (int i = 2; i <= k; ++i)

fib[i] = fib[i - 1] + fib[i - 2];

ofstream\* outs = new ofstream[k];

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

outs[i].open(tapes[i]);

if (!outs[i].is\_open()) {

cout << "Ошибка";

// Освобождаем ресурсы при ошибке

for (int j = 0; j < i; ++j) outs[j].close();

delete[] outs;

delete[] fib;

in.close();

return;

}

}

int current, next, tapeIdx = 0;

int\* runsPerTape = new int[k] {0};

if (in >> current)

{

outs[tapeIdx] << current << " ";

runsPerTape[tapeIdx]++;

while (in >> next)

{

if (next < current)

{

// Выбираем ленту с наименьшим числом серий

tapeIdx = 0;

for (int i = 1; i < k; ++i)

if (runsPerTape[i] < runsPerTape[tapeIdx])

tapeIdx = i;

}

outs[tapeIdx] << next << " ";

current = next;

}

}

}

// Основная функция сортировки

void multi(const string& inputFile, int k = 3)

{

if (k < 2) return; // ошибка

string\* tapes = new string[k];

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

tapes[i] = "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\tape" + to\_string(i) + ".txt";

distrub(inputFile, tapes, k);

string tempFile = "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\temp.txt";

int activTape = k;

while (activTape > 1)

{

// Находим непустые ленты

int nonEmpty1 = -1, nonEmpty2 = -1;

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

ifstream f(tapes[i]);

if (f.peek() != ifstream::traits\_type::eof()) {

if (nonEmpty1 == -1) nonEmpty1 = i;

else if (nonEmpty2 == -1) nonEmpty2 = i;

else break;

}

f.close();

}

if (nonEmpty2 == -1) break; // Осталась одна лента с данными

merge(tempFile, tapes[nonEmpty1], tapes[nonEmpty2]);

remove(tapes[nonEmpty1].c\_str());

remove(tapes[nonEmpty2].c\_str());

rename(tempFile.c\_str(), tapes[nonEmpty1].c\_str());

activTape--;

}

// Определяем файл с результатом

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

ifstream f(tapes[i]);

if (f.peek() != ifstream::traits\_type::eof()) {

f.close();

remove("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_multi.txt");

if (rename(tapes[i].c\_str(), "C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_multi.txt") != 0)

perror("Failed to rename result file");

break;

}

f.close();

}

// Удаляем временные файлы

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

remove(tapes[i].c\_str());

delete[] tapes;

}

void print(int\* arr, int n)

{

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

cout << arr[i] << ' ';

cout << endl << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(unsigned(time(nullptr)));

const int n = 20;

int masN[n], masB[n], masM[n];

ofstream fileN("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\natur.txt");

ofstream fileB("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\balance.txt");

ofstream fileM("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\multi.txt");

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

{

masN[i] = rand() % 100 - 50;

fileN << masN[i] << endl;

masB[i] = rand() % 100 - 50;

fileB << masB[i] << endl;

masM[i] = rand() % 100 - 50;

fileM << masM[i] << endl;

}

cout << "Массив для сортировки естественного слияния:" << endl;

print(masN, n);

natur("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\natur.txt");

ifstream sortN("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_natur.txt");

cout << "Результат сортировки: ";

int k;

while (sortN >> k)

cout << k << ' ';

cout << endl << endl;

sortN.close();

cout << "Массив для сортировки сбалансированного слияния:" << endl;

print(masB, n);

balance("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\balance.txt", 10);

ifstream sortB("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_balance.txt");

cout << "Результат сортировки: ";

while (sortB >> k)

cout << k << ' ';

cout << endl << endl;

sortB.close();

cout << "Массив для многофазной сортировки сортировки:" << endl;

print(masM, n);

multi("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\multi.txt", 10);

ifstream sortM("C:\\Users\\user\\Desktop\\C++\\Внеш\_сорт\\sorted\_multi.txt");

cout << "Результат сортировки: ";

while (sortM >> k)

cout << k << ' ';

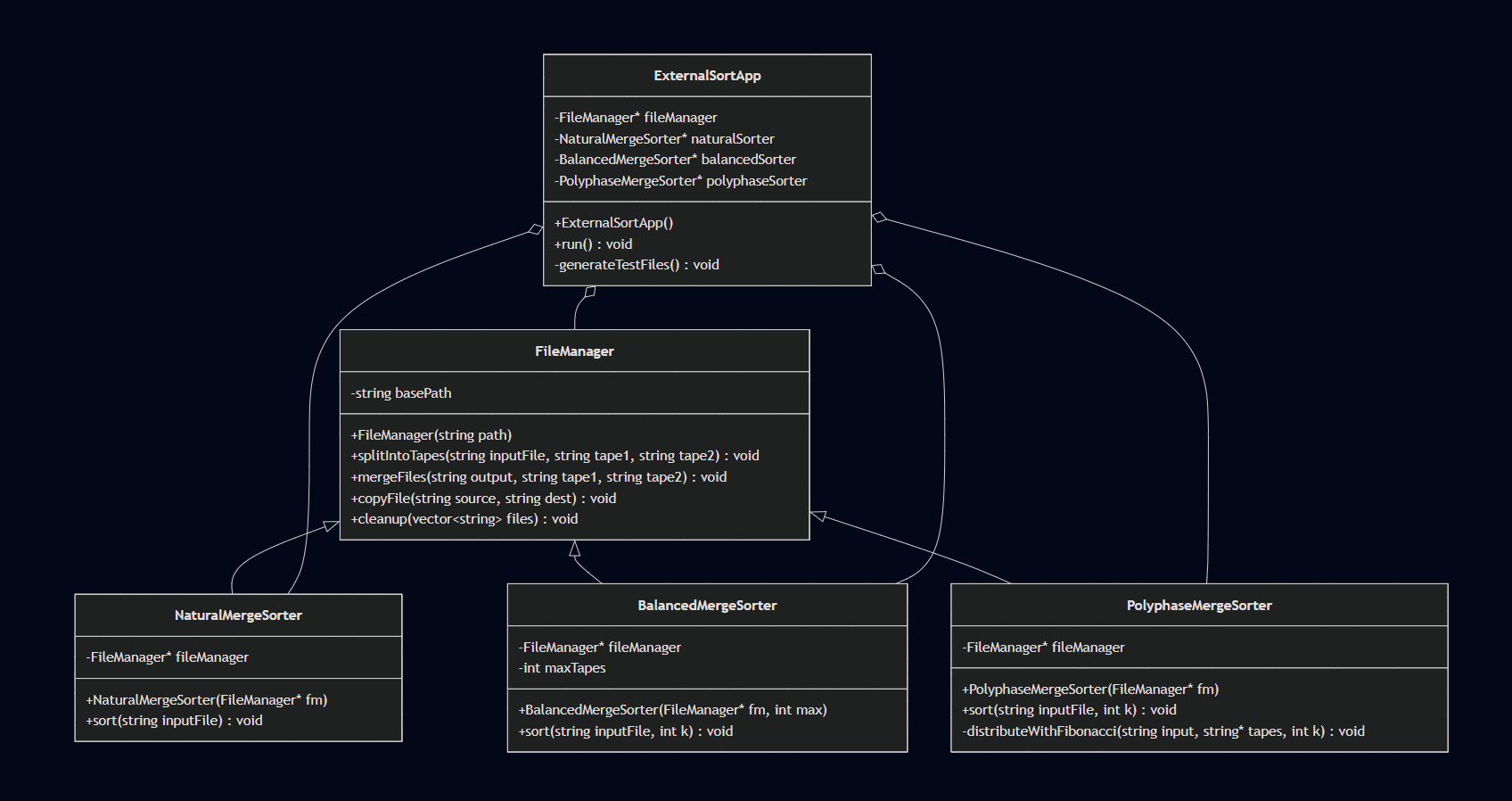
cout << endl << endl;

sortM.close();

return 0;

}

UML-диаграмма



Результат работы программы:

