Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №16**

Дисциплина: основы алгоритмизации и программирования

Тема: “Внешние сортировки”

Выполнил:

Студент группы РИС-20-2Б

Пономарёв Артём Викторович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

Отсортировать множество элементов по возрастанию методами естественного, сбалансированного и многофазного слияния.

2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать функции MergeSort() и Merging() для выполнения сбалансированной сортировки.

**1.2.** Организовать функцию EstesMergeSort(), которая отвечает за сортировку естественным слиянием.

**1.3.** Организовать функцию PolyphaseMergeSort(), которая отвечает за сортировку многофазного слияния.

**1.4.** Организовать функцию Fib(), которая генерирует количество серий для записи в файл, используя числа Фибоначчи (для многофазной сортировки).

**1.5.** Организовать функцию PrintSeries(), которая печатает серии из файлов в консоль (для многофазной сортировки).

**1.6.** Организовать функцию InsertionSort(), которая будет сортировать по возрастанию элементы в сериях (для многофазной сортировки).

**1.7.** Организовать функцию Transition(), которая будет перемещать серии из одних файлов в другие (для многофазной сортировки).

**1.8.** Организовать функцию ShowArr(), которая выводит отсортированное множество.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Переменная kol для хранения количества элементов множества.

int kol = -1;

while (kol < 1) {

cout << "Введите количество элементов в массиве >";

cin >> kol;

}

**2.2.** Вектор arr для хранения всех элементов множества.

vector<int>arr(kol);

**2.3.** Переменная method для выбора метода сортировки слияниям в ветвлении switch().

3

switch (method) {

case 1:

...

break;

case 2:

...

break;

case 3:

...

break;

}

**2.4.** Функция MergeSort() принимает в качестве параметров ссылку на вектор, в котором хранится множество элементов, переменные left и right хранят индексы на первый и последний элемент в множестве соответственно.

void MergeSort(vector<int>& arr, int left, int right, int kol)

**2.5.** Функция EstesMergeSort() принимает в качестве параметров множество элементов, количество

vector<int> EstesMergeSort(vector<int>& arr, int kol, vector<int>& resultvec)

Также используются два вспомогательных вектора tmp, tmp2; переменная ind для хранения индекса, от которого нужно заполнять tmp2.

vector<int>tmp, tmp2;

int ind = 0, count = 0;

bool flag = true;

**2.6.** В функции PolyphaseMergeSort() используется вектор векторов List для заполнения в него серий, векторы векторов F1, F2, F3 – файлы для заполнения, size1, size2 – количество серий в файлы F1, F2 с генерацией под числа Фибоначчи.

vector<vector<int>>List;

bool flag = true;

int i = 0;

…

vector<vector<int>>F1, F2, F3;

int size1, size2;

**2.7.** В функции Transition() в качестве параметров передаются файлы F1, F2, F3 с именами для параметров min, max, nullel в зависимости от положения (в каждой ситуации F1, F2, F3 могут меняться местами)

void Transition(vector<vector<int>>&min, vector<vector<int>>&max, vector<vector<int>>&nullel)

4

**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Вектор arr для хранения всех элементов множества.

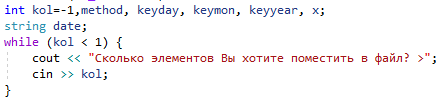
vector<int>arr(kol);

**3.2.** Используются два вспомогательных вектора tmp, tmp2

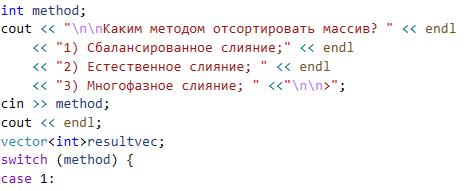
vector<int>tmp, tmp2;

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

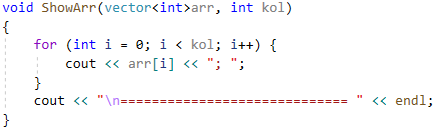
**4.1.** Для ввода количества элементов используется переменная kol. Ввод осуществляется командой cin.



**4.2.** Переменная method необходима для предоставления пользователю выбора метода поиска в массиве структур. Ввод осуществляется командой cin.

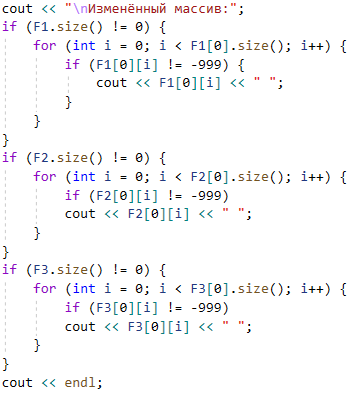


**4.3.** Вывод на консоль отсортированного множества осуществляется с помощью функции ShowArr()



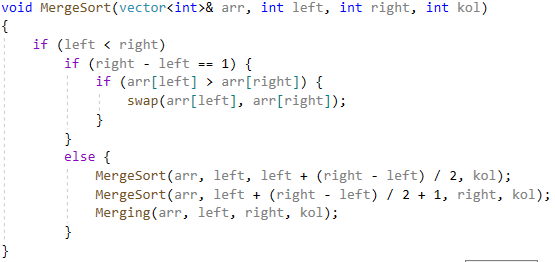
**4.4.** Вывод на консоль отсортированного множества осуществляется в циклах for для многофазной сортировки слиянием

5

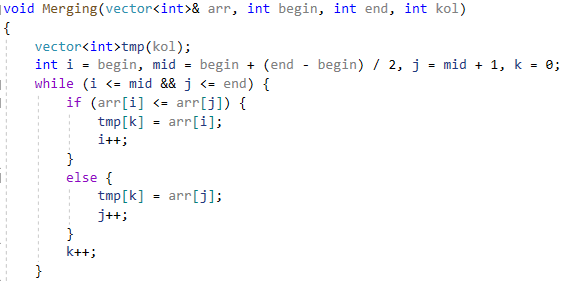


**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** Функция MergeSort() необходима для рекурсивного дробления пополам данного множества и последующего слияния с помощью функции Merging()

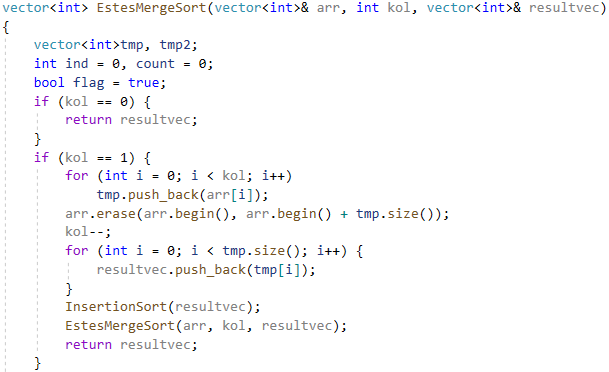


**5.2.** Функция Merging() для «сливания» частей множества.

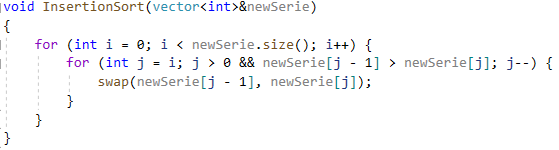


6

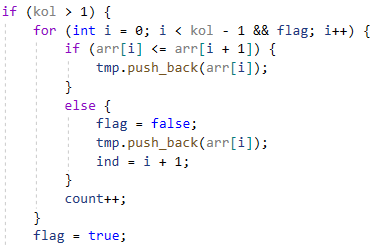
**5.3.** Функция EstesMergeSort() – рекурсивная, которая возвращает отсортированное множество resultvec. Kol=0 – признак окончания рекурсии. Kol=1 предпоследний вызов функции.



В цикле for заполняется последний оставшийся элемент множества, затем выполняется очистка изначального множества. В следующем цикле for выполняется заполнение resultvec[], значение которого выводит функция. Далее вызывается функция InsertionSort(), в котором происходит сортировка методом вставки.

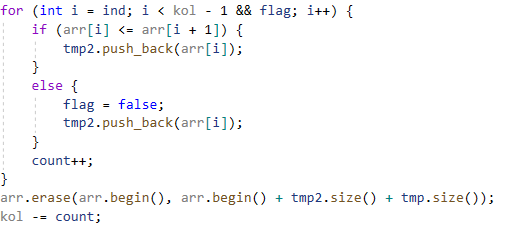


Пока kol>1 в цикле for заполняется вектор tmp[], пока элементы множества по возрастанию, где переменная ind необходима для запоминания индекса для заполнения следующей серии эллементов, а count – счётчик итераций цикла для вычитания из kol.

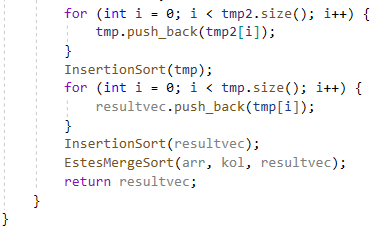


7

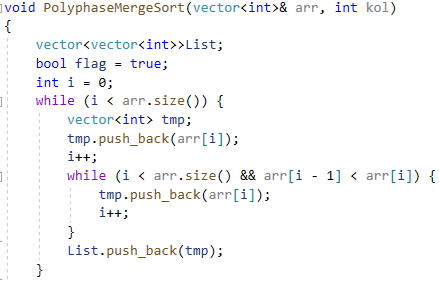
В следующем цикле for заполняется tmp2[] – вторая серия элеменов по возрастанию. Далее очищается изначальное множество arr[] (для заполнения tmp[] и tmp2[] в следующем вызове функции)



Далее заполняется tmp[] в цикле for из tmp2[]. Далее заполняется resultvec[] из tmp и сортируется через функцию InsertionSort().



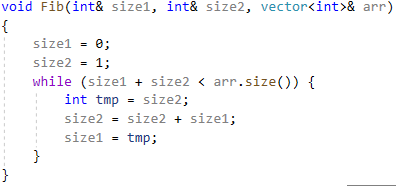
**5.4.** Функция PolyphaseMergeSort(), в цикле while производится заполнение вектора векторов List по сериям по признаку возрастания.



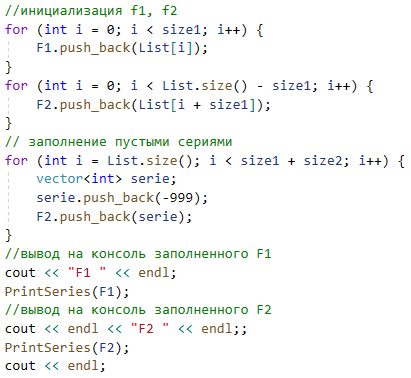
**5.5.** Генерация по количеству серий в файлах F1, F2, F3



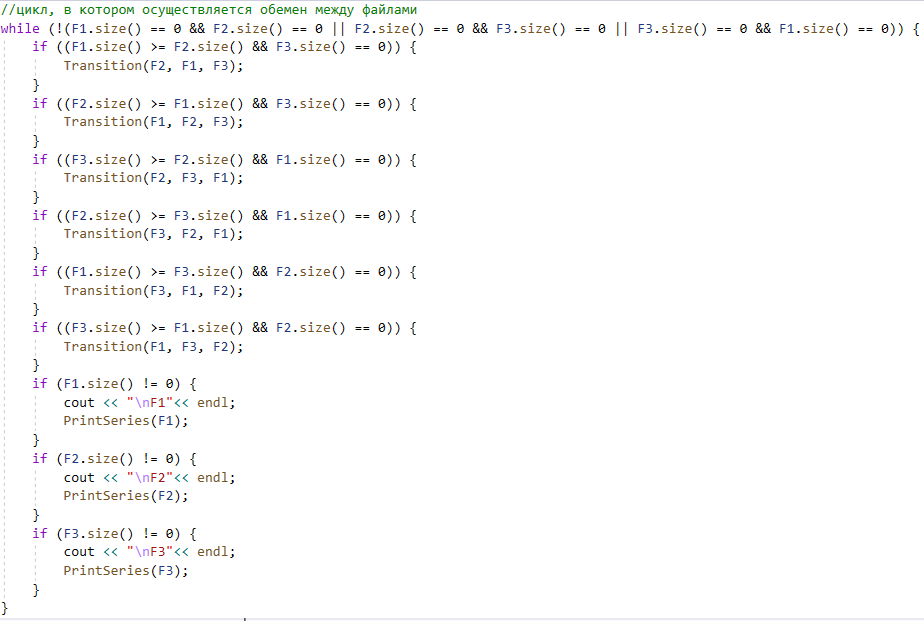
8



**5.6.** Заполнение файлов F1, F2 сериями.

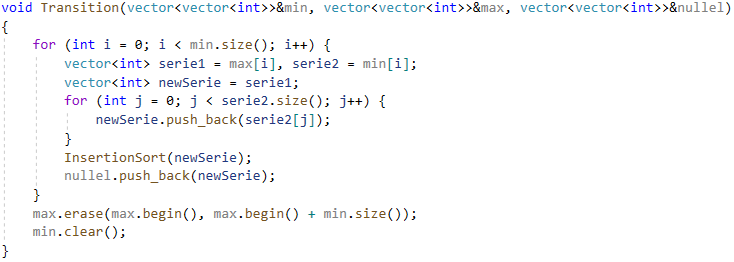


**5.7.** В цикле while осуществляется обмен между файлами до того момента пока два файла не станут пустыми (признак окончания сортировки).



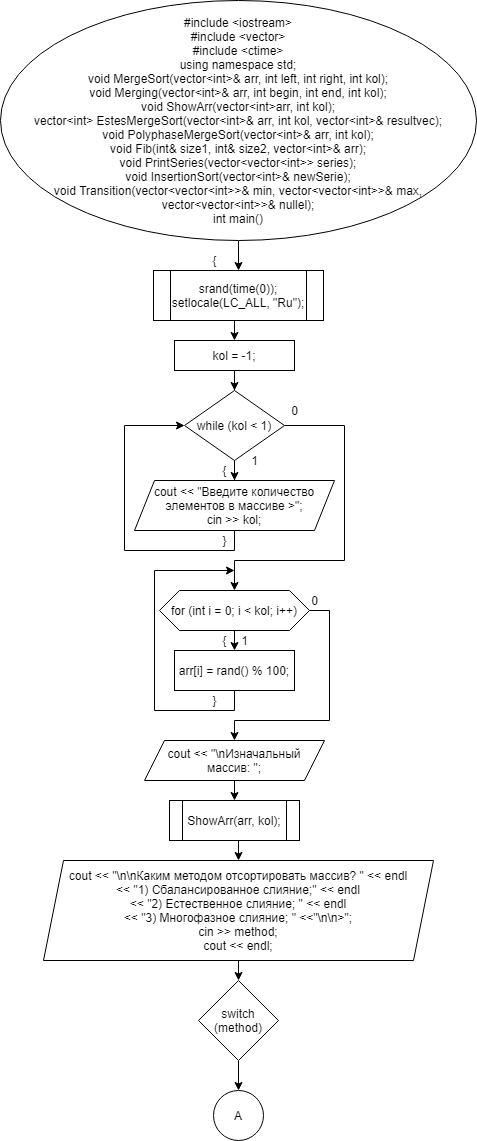
9

**5.8.** Обмен происходит в функции Transition(), где в цикле for выполняется заполнение пустого файла из файла, который заполнен меньшим (min) количеством серий по сравнению с другим (max) и удаление того же количества серий из файла с большим количеством серий (max)

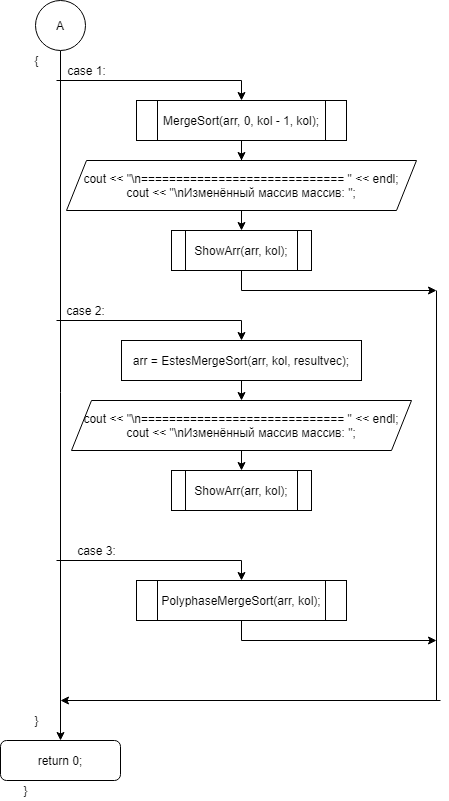


10

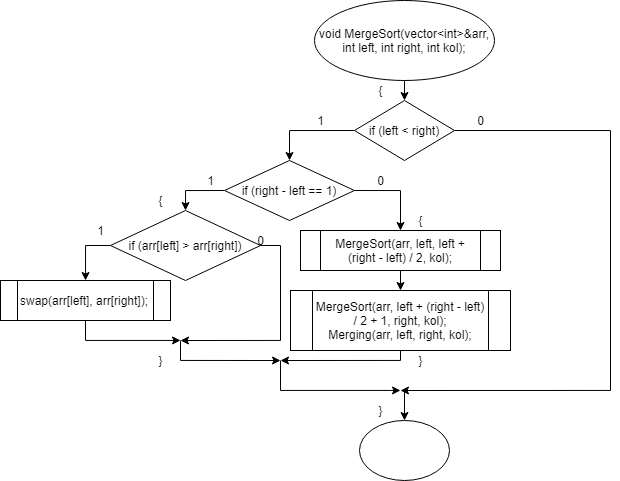
**Блок-схема**



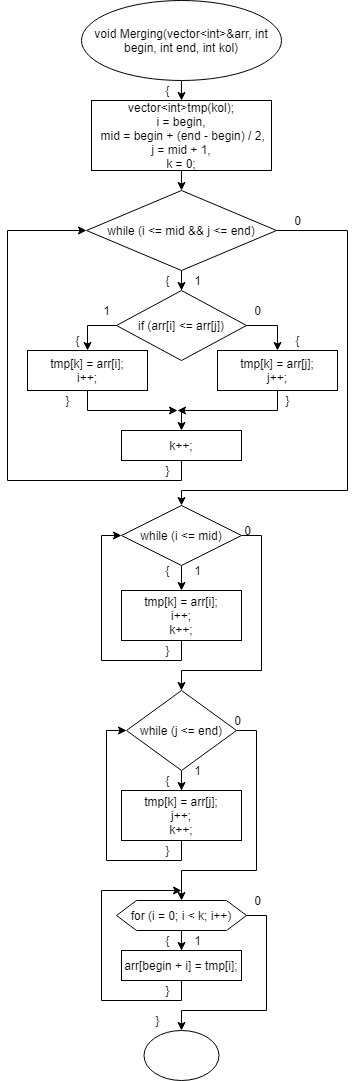
11



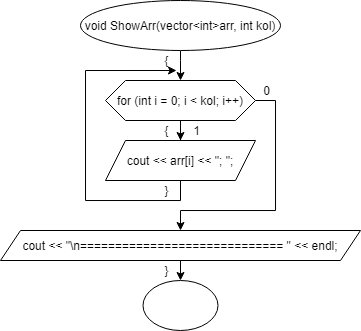
12



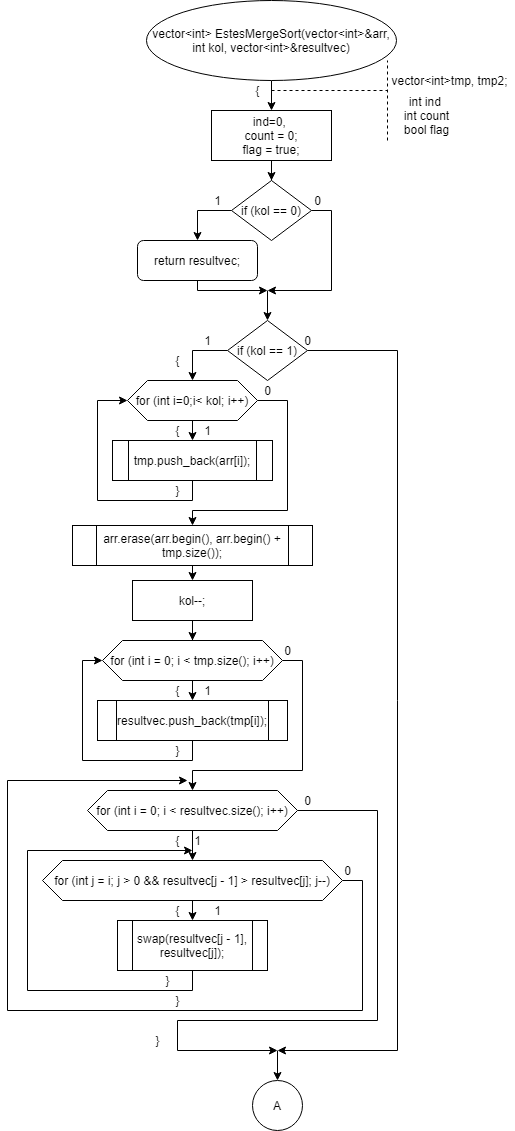
13



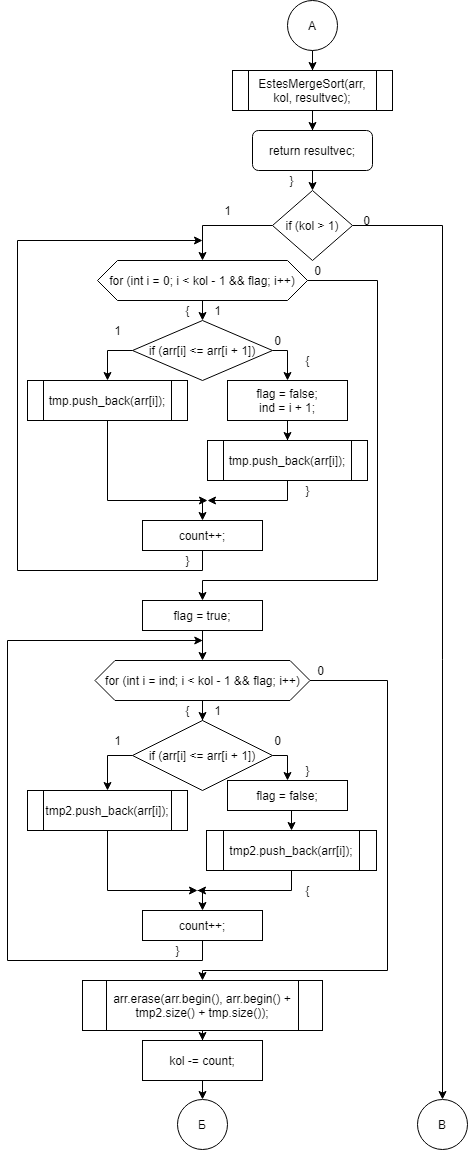
14



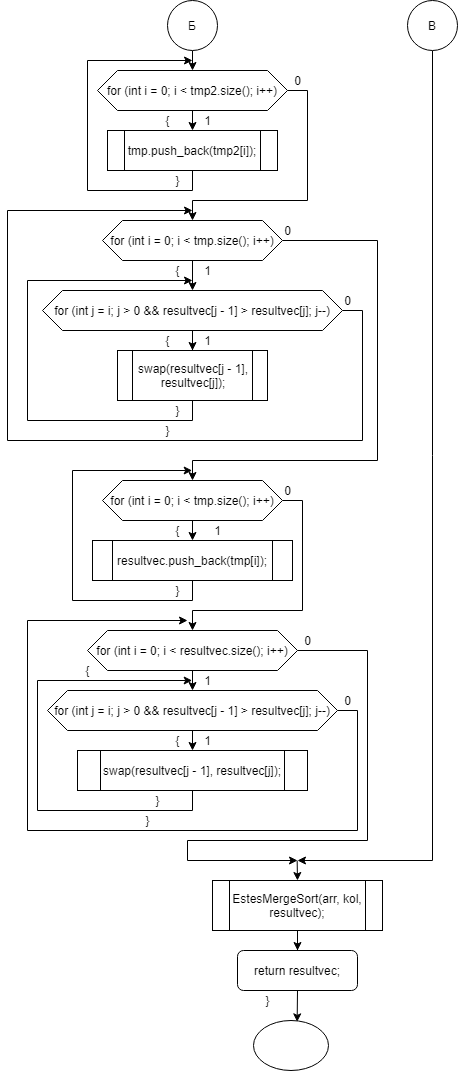
15



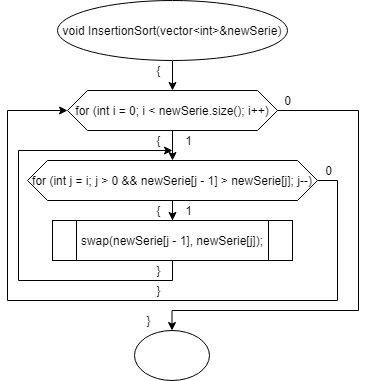
16



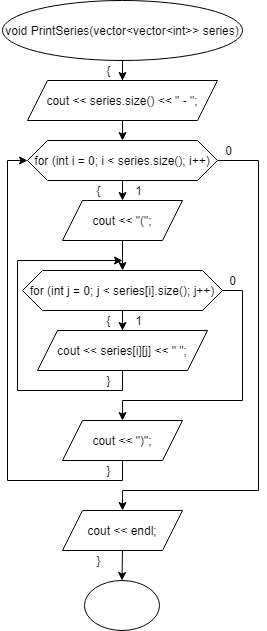
17



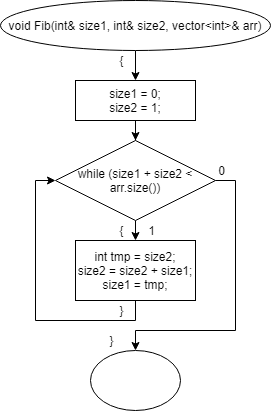
18



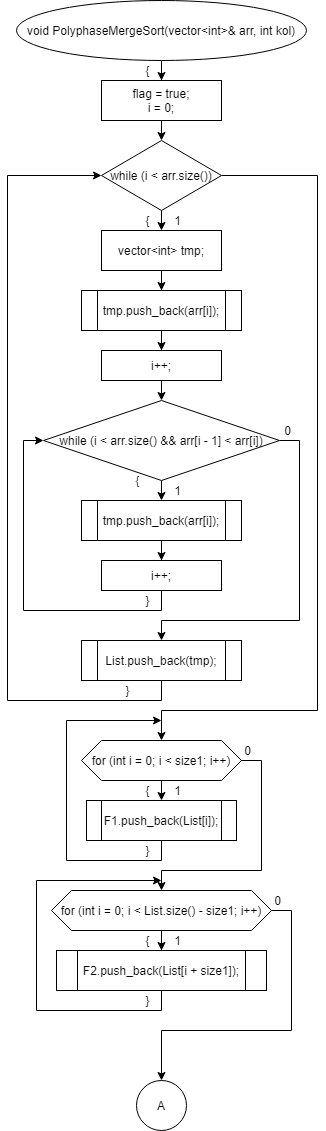
19



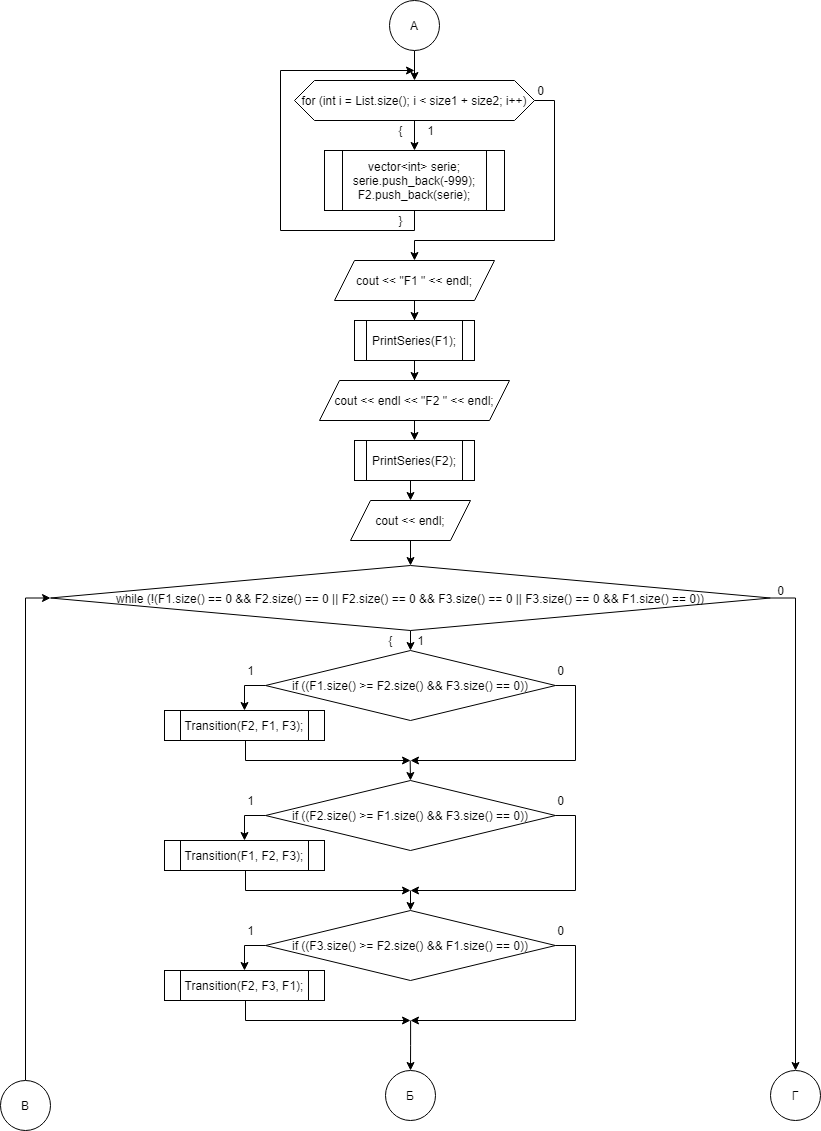
20



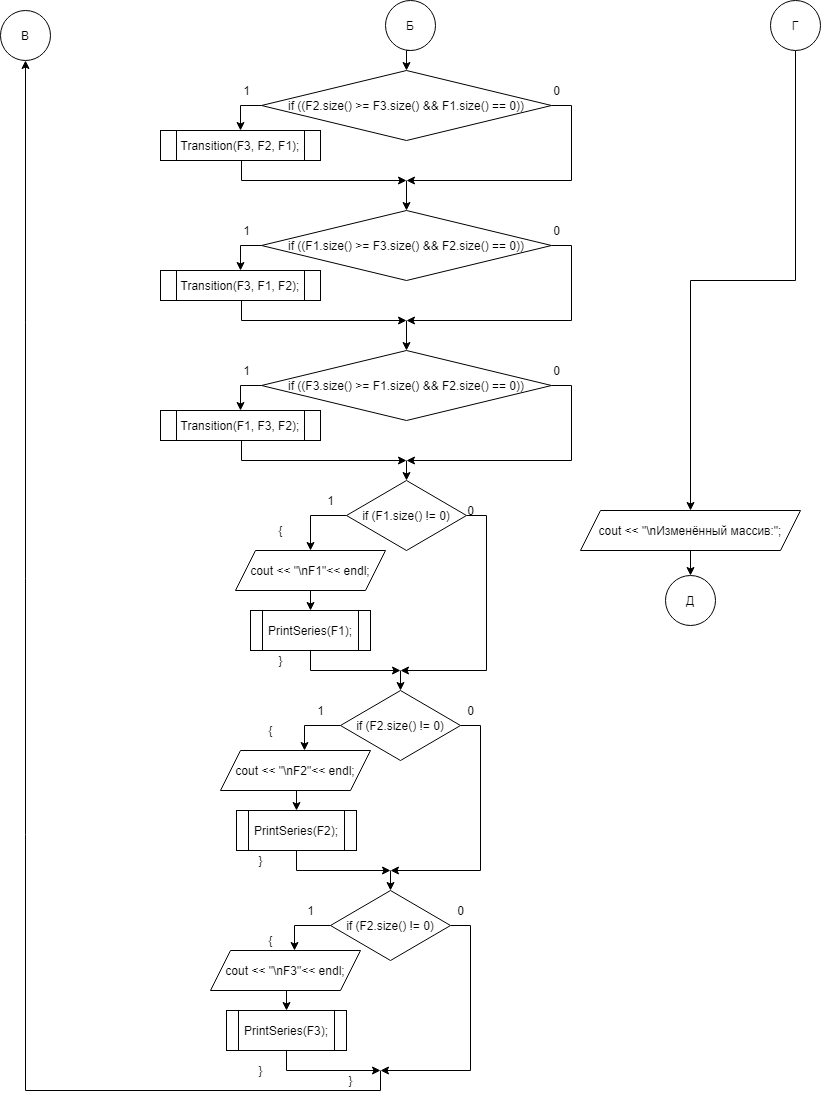
21



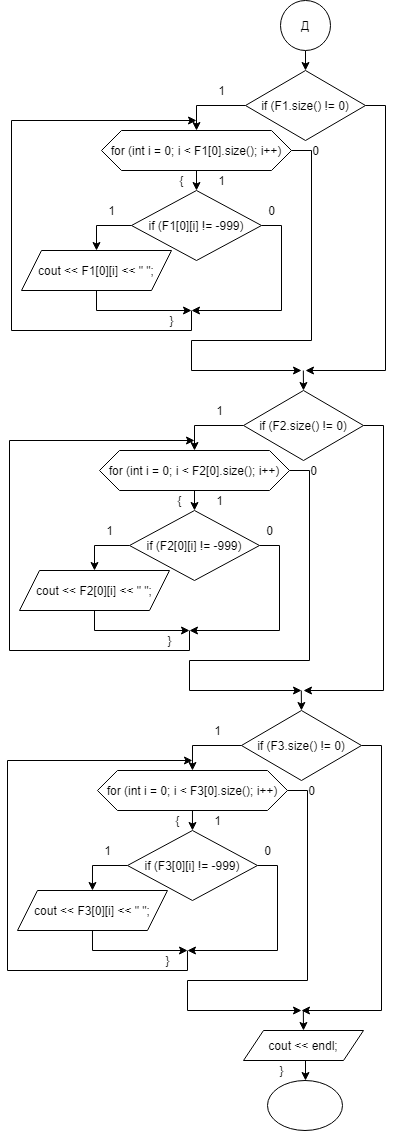
22



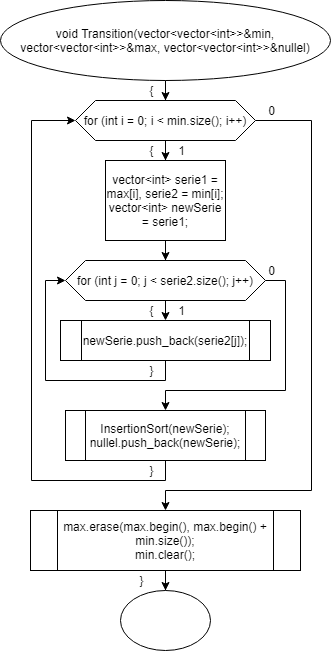
23



24



25



26

**Код**

#include <iostream>

#include <vector>

#include <ctime>

using namespace std;

void MergeSort(vector<int>& arr, int left, int right, int kol);

void Merging(vector<int>& arr, int begin, int end, int kol);

void ShowArr(vector<int>arr, int kol);

vector<int> EstesMergeSort(vector<int>& arr, int kol, vector<int>& resultvec);

void PolyphaseMergeSort(vector<int>& arr, int kol);

void Fib(int& size1, int& size2, vector<int>& arr);

void PrintSeries(vector<vector<int>> series);

void InsertionSort(vector<int>& newSerie);

void Transition(vector<vector<int>>& min, vector<vector<int>>& max, vector<vector<int>>& nullel);

int main()

{

srand(time(0));

setlocale(LC\_ALL, "Ru");

int kol = -1;

while (kol < 1) {

cout << "Введите количество элементов в массиве >";

cin >> kol;

}

vector<int>arr(kol);

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

arr[i] = rand() % 100;

}

cout << "\nИзначальный массив: ";

ShowArr(arr, kol);

int method;

cout << "\n\nКаким методом отсортировать массив? " << endl

<< "1) Сбалансированное слияние;" << endl

<< "2) Естественное слияние; " << endl

<< "3) Многофазное слияние; " <<"\n\n>";

cin >> method;

cout << endl;

vector<int>resultvec;

switch (method) {

case 1:

MergeSort(arr, 0, kol - 1, kol);

cout << "\n============================= " << endl;

cout << "\nИзменённый массив массив: ";

ShowArr(arr, kol);

break;

case 2:

arr = EstesMergeSort(arr, kol, resultvec);

cout << "\n============================= " << endl;

cout << "\nИзменённый массив массив: ";

ShowArr(arr, kol);

break;

case 3:

PolyphaseMergeSort(arr, kol);

break;

}

}

void ShowArr(vector<int>arr, int kol)

{

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

cout << arr[i] << "; ";

}

cout << "\n============================= " << endl;

27

}

void MergeSort(vector<int>& arr, int left, int right, int kol)

{

if (left < right)

if (right - left == 1) {

if (arr[left] > arr[right]) {

swap(arr[left], arr[right]);

}

}

else {

MergeSort(arr, left, left + (right - left) / 2, kol);

MergeSort(arr, left + (right - left) / 2 + 1, right, kol);

Merging(arr, left, right, kol);

}

}

void Merging(vector<int>& arr, int begin, int end, int kol)

{

vector<int>tmp(kol);

int i = begin, mid = begin + (end - begin) / 2, j = mid + 1, k = 0;

while (i <= mid && j <= end) {

if (arr[i] <= arr[j]) {

tmp[k] = arr[i];

i++;

}

else {

tmp[k] = arr[j];

j++;

}

k++;

}

while (i <= mid) {

tmp[k] = arr[i];

i++;

k++;

}

while (j <= end) {

tmp[k] = arr[j];

j++;

k++;

}

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

arr[begin + i] = tmp[i];

}

}

vector<int> EstesMergeSort(vector<int>& arr, int kol, vector<int>& resultvec)

{

vector<int>tmp, tmp2;

int ind = 0, count = 0;

bool flag = true;

if (kol == 0) {

return resultvec;

}

if (kol == 1) {

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

tmp.push\_back(arr[i]);

arr.erase(arr.begin(), arr.begin() + tmp.size());

kol--;

for (int i = 0; i < tmp.size(); i++) {

resultvec.push\_back(tmp[i]);

}

InsertionSort(resultvec);

EstesMergeSort(arr, kol, resultvec);

28

return resultvec;

}

if (kol > 1) {

for (int i = 0; i < kol - 1 && flag; i++) {

if (arr[i] <= arr[i + 1]) {

tmp.push\_back(arr[i]);

}

else {

flag = false;

tmp.push\_back(arr[i]);

ind = i + 1;

}

count++;

}

flag = true;

for (int i = ind; i < kol - 1 && flag; i++) {

if (arr[i] <= arr[i + 1]) {

tmp2.push\_back(arr[i]);

}

else {

flag = false;

tmp2.push\_back(arr[i]);

}

count++;

}

arr.erase(arr.begin(), arr.begin() + tmp2.size() + tmp.size());

kol -= count;

for (int i = 0; i < tmp2.size(); i++) {

tmp.push\_back(tmp2[i]);

}

InsertionSort(tmp);

for (int i = 0; i < tmp.size(); i++) {

resultvec.push\_back(tmp[i]);

}

InsertionSort(resultvec);

EstesMergeSort(arr, kol, resultvec);

return resultvec;

}

}

void PolyphaseMergeSort(vector<int>& arr, int kol)

{

vector<vector<int>>List;

bool flag = true;

int i = 0;

while (i < arr.size()) {

vector<int> tmp;

tmp.push\_back(arr[i]);

i++;

while (i < arr.size() && arr[i - 1] < arr[i]) {

tmp.push\_back(arr[i]);

i++;

}

List.push\_back(tmp);

}

vector<vector<int>>F1, F2, F3;

//числа фибоначи

int size1, size2;

Fib(size1, size2, arr);

//инициализация f1, f2

for (int i = 0; i < size1; i++) {

F1.push\_back(List[i]);

}

for (int i = 0; i < List.size() - size1; i++) {

29

F2.push\_back(List[i + size1]);

}

// заполнение пустыми сериями

for (int i = List.size(); i < size1 + size2; i++) {

vector<int> serie;

serie.push\_back(-999);

F2.push\_back(serie);

}

//вывод на консоль заполненного F1

cout << "F1 " << endl;

PrintSeries(F1);

//вывод на консоль заполненного F2

cout << endl << "F2 " << endl;

PrintSeries(F2);

cout << endl;

//цикл, в котором осуществляется обемен между файлами

while (!(F1.size() == 0 && F2.size() == 0 || F2.size() == 0 && F3.size() == 0 || F3.size() == 0 && F1.size() == 0)) {

if ((F1.size() >= F2.size() && F3.size() == 0)) {

Transition(F2, F1, F3);

}

if ((F2.size() >= F1.size() && F3.size() == 0)) {

Transition(F1, F2, F3);

}

if ((F3.size() >= F2.size() && F1.size() == 0)) {

Transition(F2, F3, F1);

}

if ((F2.size() >= F3.size() && F1.size() == 0)) {

Transition(F3, F2, F1);

}

if ((F1.size() >= F3.size() && F2.size() == 0)) {

Transition(F3, F1, F2);

}

if ((F3.size() >= F1.size() && F2.size() == 0)) {

Transition(F1, F3, F2);

}

if (F1.size() != 0) {

cout << "\nF1"<< endl;

PrintSeries(F1);

}

if (F2.size() != 0) {

cout << "\nF2"<< endl;

PrintSeries(F2);

}

if (F3.size() != 0) {

cout << "\nF3"<< endl;

PrintSeries(F3);

}

}

cout << "\nИзменённый массив:";

if (F1.size() != 0) {

for (int i = 0; i < F1[0].size(); i++) {

if (F1[0][i] != -999) {

cout << F1[0][i] << " ";

}

}

}

if (F2.size() != 0) {

for (int i = 0; i < F2[0].size(); i++) {

if (F2[0][i] != -999)

cout << F2[0][i] << " ";

}

}

if (F3.size() != 0) {

30

for (int i = 0; i < F3[0].size(); i++) {

if (F3[0][i] != -999)

cout << F3[0][i] << " ";

}

}

cout << endl;

}

void Fib(int& size1, int& size2, vector<int>& arr)

{

size1 = 0;

size2 = 1;

while (size1 + size2 < arr.size()) {

int tmp = size2;

size2 = size2 + size1;

size1 = tmp;

}

}

void PrintSeries(vector<vector<int>> series)

{

cout << series.size() << " - ";

for (int i = 0; i < series.size(); i++) {

cout << "(";

for (int j = 0; j < series[i].size(); j++) {

cout << series[i][j] << " ";

}

cout << ")";

}

cout << endl;

}

void InsertionSort(vector<int>&newSerie)

{

for (int i = 0; i < newSerie.size(); i++) {

for (int j = i; j > 0 && newSerie[j - 1] > newSerie[j]; j--) {

swap(newSerie[j - 1], newSerie[j]);

}

}

}

void Transition(vector<vector<int>>&min, vector<vector<int>>&max, vector<vector<int>>&nullel)

{

for (int i = 0; i < min.size(); i++) {

vector<int> serie1 = max[i], serie2 = min[i];

vector<int> newSerie = serie1;

for (int j = 0; j < serie2.size(); j++) {

newSerie.push\_back(serie2[j]);

}

InsertionSort(newSerie);

nullel.push\_back(newSerie);

}

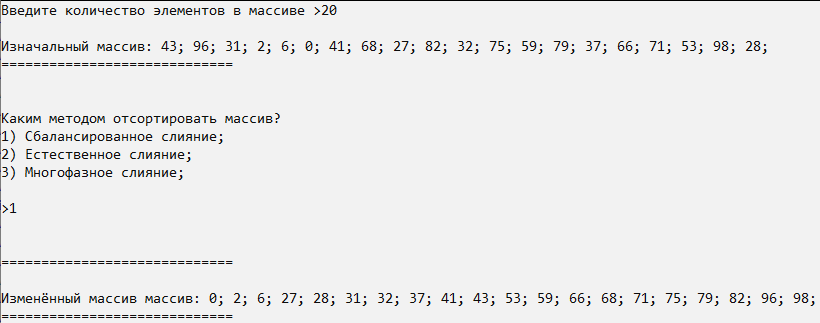
max.erase(max.begin(), max.begin() + min.size());

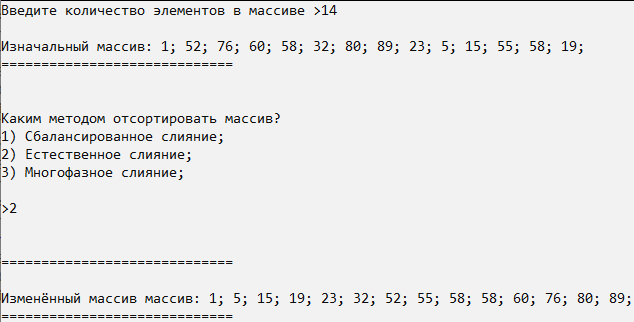
min.clear();

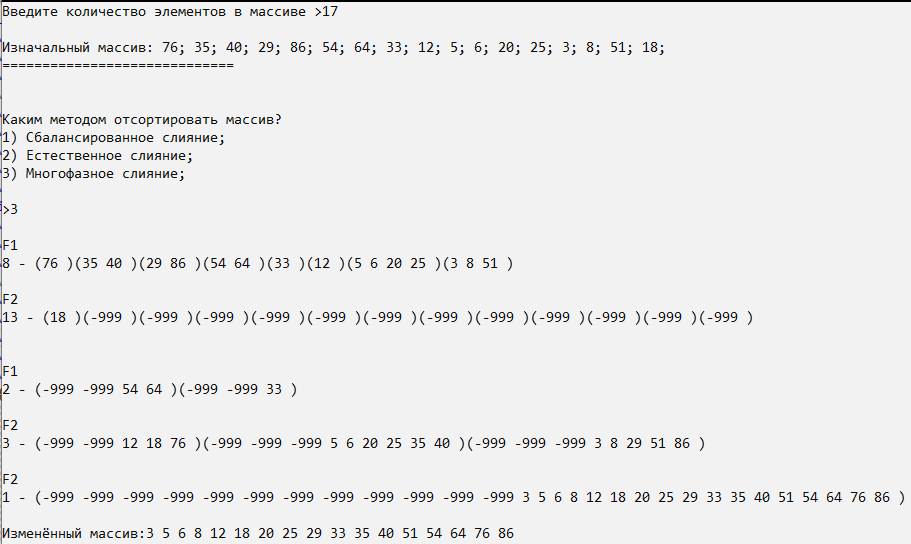
}

31

**Скриншоты**







32