**“ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ**

**УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ” (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

Отчет по лабораторной работе №2

«Сортировка данных»

По дисциплине СиАОДвЭВМ

Выполнил студент гр. 438-3:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Канаев О.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Проверил

Доктор технических наук,

доцент каф. АСУ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Горитов А.Н

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Томск 2020

**Цель работы:**

Получить первичные навыки создания алгоритмов сортировки: турнирная сортировка.

**Задание:**

8. Пусть даны три файла вещественных чисел, состоящих из 100, 1000 и 10000 чисел. Выполнить сортировку данных с помощью метода турнирной сортировки. Подсчитать количество операций сравнения для каждого из трех файлов.

**Теория:**

**Алгоритм сортировки** — это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

Сортировки применяется для сокращения времени поиска элементов в

упорядоченном множестве.

**Сортировка** – это упорядочивание набора однотипных данных по возрастанию или убыванию. **Ключ сортировки** – это часть данных, определяющая порядок элементов.

Метод турнирной сортировки основан на повторяющихся поисках наименьшего ключа среди п элементов, среди оставшихся п— 1 элементов и т.д.

**Алгоритм:**

1. Составить дерево, где числа входного сортируемого множества расположены в листьях этого дерева. Эти элементы располагаются на уровне k, где 2k >= n
2. Предполагаем, что список сортируется по возрастанию. Из каждой соседней пары выбираем наименьший элемент и заносим его в родительскую вершину этих двух элементов.
3. Определили наименьший ключ, который находится в корне дерева. Перенесем этот ключ в выходную последовательность (массив).
4. Для нахождения следующего по величине ключа, спустимся от корня по пути, ведущему к листу с наименьшим значением ключа и заменим его на +∞ (1 в моем случае). За единицу я принимаю +∞, тк функция Random, которая генирирует текстовые файлы с рандомными числами с плавающей точкой в диапазоне от 0 до 1

**Решение:**

#include <iostream>

#include <locale.h>

#include <string>

#include <fstream>

#include <cstdlib>

#include <ctime>

using namespace std;

struct BTree

{

float data;

BTree \*left, \*right;

};

int t=1, num2=0;

float \*arr; //Выходная последовательность (массив)

void DeleteT(BTree \*&root); //Удаление дерева

int Add\_to\_T(BTree \*&root, float num); //Добавление элемента

void Show(BTree \*&root, int level); //Вывод на экран

void Random(int choice); //Функция, составяющая тектовые файлы

void Comparison\_func(BTree \*&root, int level); //Функция сравнения

void Sorting\_func(BTree \*&root,int num); //Сортирующая функция

int main()

{

srand(time(NULL));

system("chcp 1251");

BTree \*Root = new BTree;

if (Root == nullptr)

{

cout << "Память не выделилась";

return 0;

} //Проверка на выделение памяти

Root = nullptr;

Random(1);

Random(2);

Random(3);

int num;

cout << "Выберите:" << endl;

cout << "1-Файл из 10 (сделан для показа функционала программы)" << endl;

cout << "2-Файл из 100" << endl;

cout << "3-Файл из 1000" << endl;

cout << "4-Файл из 10000" << endl;

cout << "Выбор: ";

cin >> num;

cout << "\n\n";

switch (num)

{

case(1):

{

ifstream file;

float number;

file.open("10.txt");

arr = new float[10];

if (!file.is\_open())

{

cout << "Ошибка открытия" << endl;

}

while (!file.eof())

{

if (t > (pow(2, 4) - 1)) file >> number;

else {

number = 1;

t++;

}

Add\_to\_T(Root, number);

}

Show(Root, 0);

cout << "\n\n";

Comparison\_func(Root, 0);

Show(Root, 0);

Sorting\_func(Root, num);

cout << "Выходная последовательность: ";

for (int i = 0; i < 10; i++) cout << arr[i] << " "; //Ключи в порядке сортировки

num2 /= 2;

cout << "\nКол-во сравнений: " << num2 << "\n";

file.close();

break;

}

case(2):

{

ifstream file;

float number;

file.open("100.txt");

arr = new float[100];

if (!file.is\_open())

{

cout << "Ошибка открытия" << endl;

}

while (!file.eof())

{

if (t >(pow(2, 7)-1)) file >> number;

else {

number = 2;

t++;

}

Add\_to\_T(Root, number);

}

Comparison\_func(Root, 0);

//Show(Root, 0);

Sorting\_func(Root, num);

cout << "Выходная последовательность: ";

for (int i = 0; i < 100; i++) cout << arr[i] << " "; //Ключи в порядке сортировки

num2 /= 2;

cout << "\nКол-во сравнений: " << num2<<"\n";

file.close();

break;

}

case(3):

{

ifstream file;

float number;

arr = new float[1000];

file.open("1000.txt");

if (!file.is\_open())

{

cout << "Ошибка открытия" << endl;

}

while (!file.eof())

{

if (t > (pow(2, 10) - 1)) file >> number;

else {

number = 0;

t++;

}

Add\_to\_T(Root, number);

}

Comparison\_func(Root, 0);

//Show(Root, 0);

Sorting\_func(Root, num);

cout << "Выходная последовательность: ";

for (int i = 0; i < 1000; i++) cout << arr[i] << " "; //Ключи в порядке сортировки

num2 /= 2;

cout << "\nКол-во сравнений: " << num2 << "\n";

file.close();

break;

}

case(4):

{

ifstream file;

float number;

arr = new float[10000];

file.open("10000.txt");

if (!file.is\_open())

{

cout << "Ошибка открытия" << endl;

}

while (!file.eof())

{

if (t > (pow(2, 14) - 1)) file >> number;

else {

number = 0;

t++;

}

Add\_to\_T(Root, number);

}

Comparison\_func(Root, 0);

//Show(Root, 0);

Sorting\_func(Root, num);

cout << "Выходная последовательность: ";

for (int i = 0; i < 10000; i++) cout << arr[i] << " "; //Ключи в порядке сортировки

num2 /= 2;

cout << "\nКол-во сравнений: " << num2 << "\n";

file.close();

break;

}

}

DeleteT(Root);

delete[] arr;

system("pause");

return 0;

}

void DeleteT(BTree \*&root)

{

if (root != nullptr)

{

DeleteT(root->left);

DeleteT(root->right);

delete root;

root = nullptr;

}

}

int Add\_to\_T(BTree \*&root, float num)

{ //Заполнение слева направо последнего уровня (куча)

if (root == nullptr) //Если корень пуст

{

root = new BTree;

if (root == nullptr) return 0; //Проверка на выделение памяти

root->data = num;

root->left = root->right = nullptr;

}

else

{

if (root->left == nullptr) //Если слева пусто

{

root->left = new BTree;

if (root->left == nullptr) return 0; //Проверка на выделение памяти

root->left->left = root->left->right = nullptr;

root->left->data = num;

}

else if (root->right == nullptr) //Если справа пусто

{

root->right = new BTree;

if (root->right == nullptr) return 0; //Проверка на выделние памяти

root->right->left = root->right->right = nullptr;

root->right->data = num;

}

else if ((root->left->left == nullptr) || (root->left->right == nullptr)) Add\_to\_T(root->left, num);

else if ((root->right->left == nullptr) || (root->right->right == nullptr)) Add\_to\_T(root->right, num);

else {

int i = 0, j = 0;

BTree \*temp = root;

while (temp != nullptr) {

if (i == 0) temp = temp->left;

else temp = temp->right;

i++;

}

temp = root;

while (temp != nullptr) {

temp = temp->right;

j++;

}

if (i == j) Add\_to\_T(root->left, num);

else if (i > j) Add\_to\_T(root->right, num);

}

}

return 0;

}

void Show(BTree \*&root, int level)

{

if (root)

{

Show(root->right, level + 1);

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

cout << " ";

cout << root->data << endl;

Show(root->left, level + 1);

}

}

void Random(int choice) {

ofstream file;

float num;

switch (choice)

{

case(1):

{

file.open("10.txt");

for (int i = 1; i <= 10; i++)

{

num = 0.01 \* (rand() % 101);

file << num;

if (i % 25 == 0) file << '\n';

else if (i != 10) file << ' ';

}

break;

}

case(2):

{

file.open("100.txt");

for (int i = 1; i <= 100; i++)

{

num = 0.01 \* (rand() % 101);

file << num;

if ((i % 25 == 0) && (i != 100)) file << '\n';

else if (i != 100) file << ' ';

}

break;

}

case(3):

{

file.open("1000.txt");

for (int i = 1; i <= 1000; i++)

{

num = 0.01 \* (rand() % 101);

file << num;

if ((i % 25 == 0) && (i != 1000)) file << '\n';

else if (i != 1000) file << ' ';

}

break;

}

case(4):

{

file.open("10000.txt");

for (int i = 1; i <= 10000; i++)

{

num = 0.01 \* (rand() % 101);

file << num;

if ((i % 25 == 0) && (i != 10000)) file << '\n';

else if (i != 10000) file << ' ';

}

break;

}

}

file.close();

}

void Comparison\_func(BTree \*&root, int level)

{

if (root)

{

if ((root->left != nullptr) && (root->right != nullptr))

{

num2++;

if (root->left->data >= root->right->data) root->data = root->right->data;

else root->data = root->left->data;

}

Comparison\_func(root->right, level + 1);

Comparison\_func(root->left, level + 1);

if ((root->left != nullptr) && (root->right != nullptr))

{

num2++;

if (root->left->data >= root->right->data) root->data = root->right->data;

else root->data = root->left->data;

}

}

}

void Sorting\_func(BTree \*&root, int num)

{

int number;

if (num == 1) number = 10;

else if (num == 2) number = 100;

else if (num == 3) number = 1000;

else if (num == 4) number = 10000;

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

{

BTree \*temp = root;

arr[i] = temp->data;

while ((temp->left!=nullptr) && (temp->right!=nullptr))

{

if (temp->left->data >= temp->right->data) temp = temp->right;

else temp = temp->left;

}

temp->data = 1;

Comparison\_func(root, 0);

if (num == 1) { //Чтобы не нагружать компьютер

cout << "\n\\\\\\\\\\\\\\\\\\\\\\\\\\" << i << "-я итерация\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\";

Show(root, 0);

}

}

}

**Результат работы:**

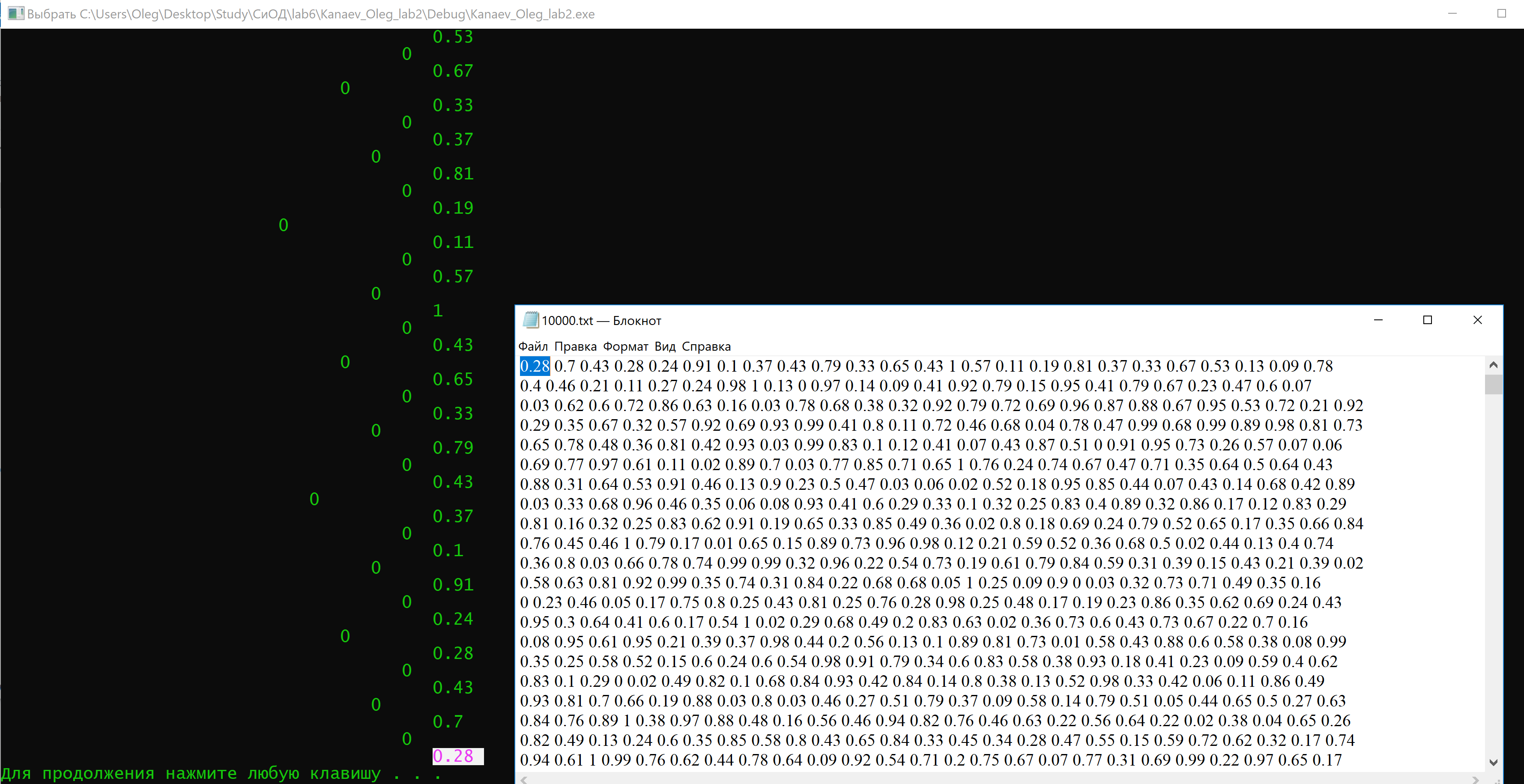


Рисунок №1 —Данные из файла 10000.txt, представленные в листьях дерева

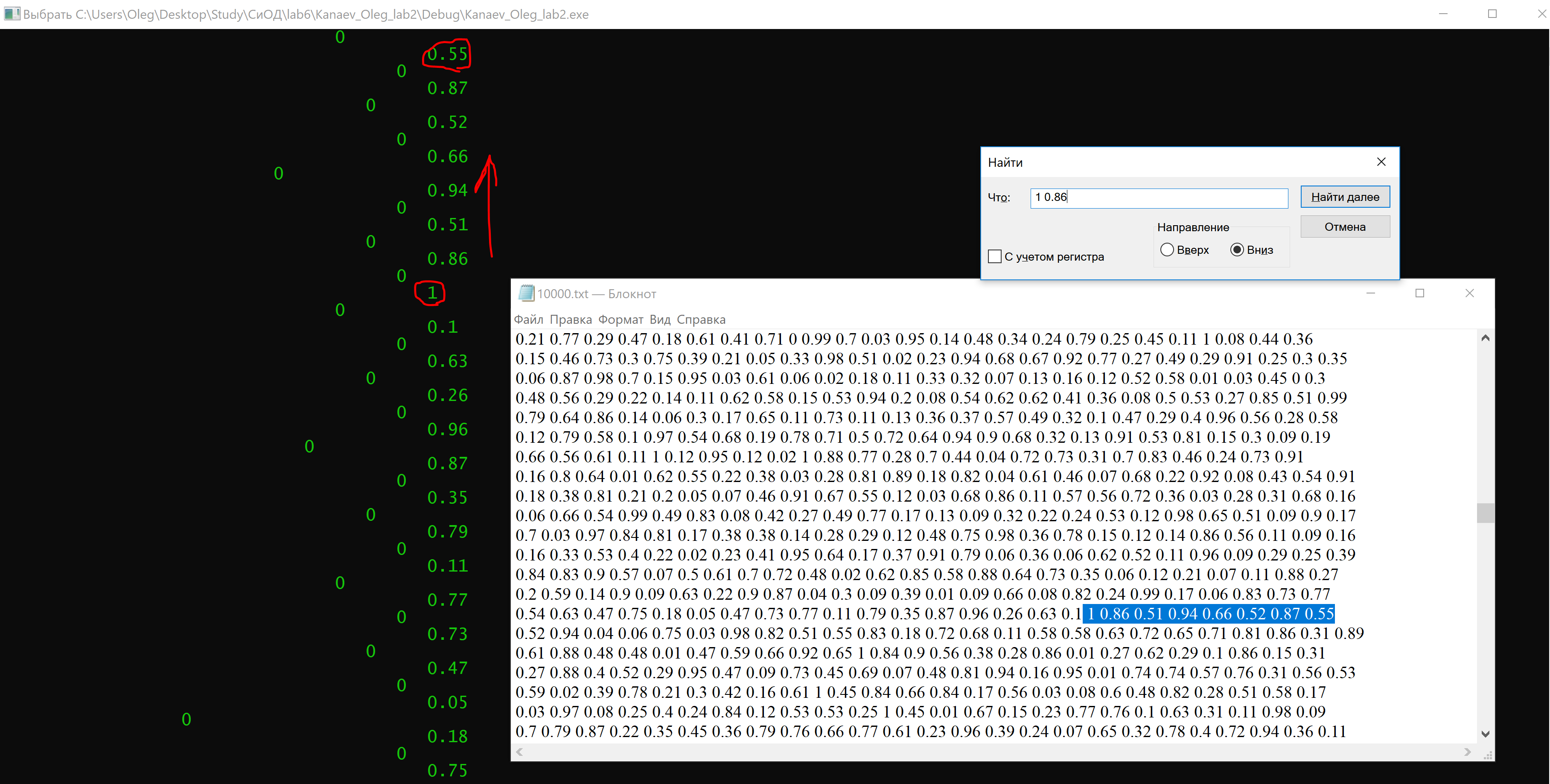


Рисунок №2 —Данные из файла 10000.txt. Проверка на правильность заполнения.

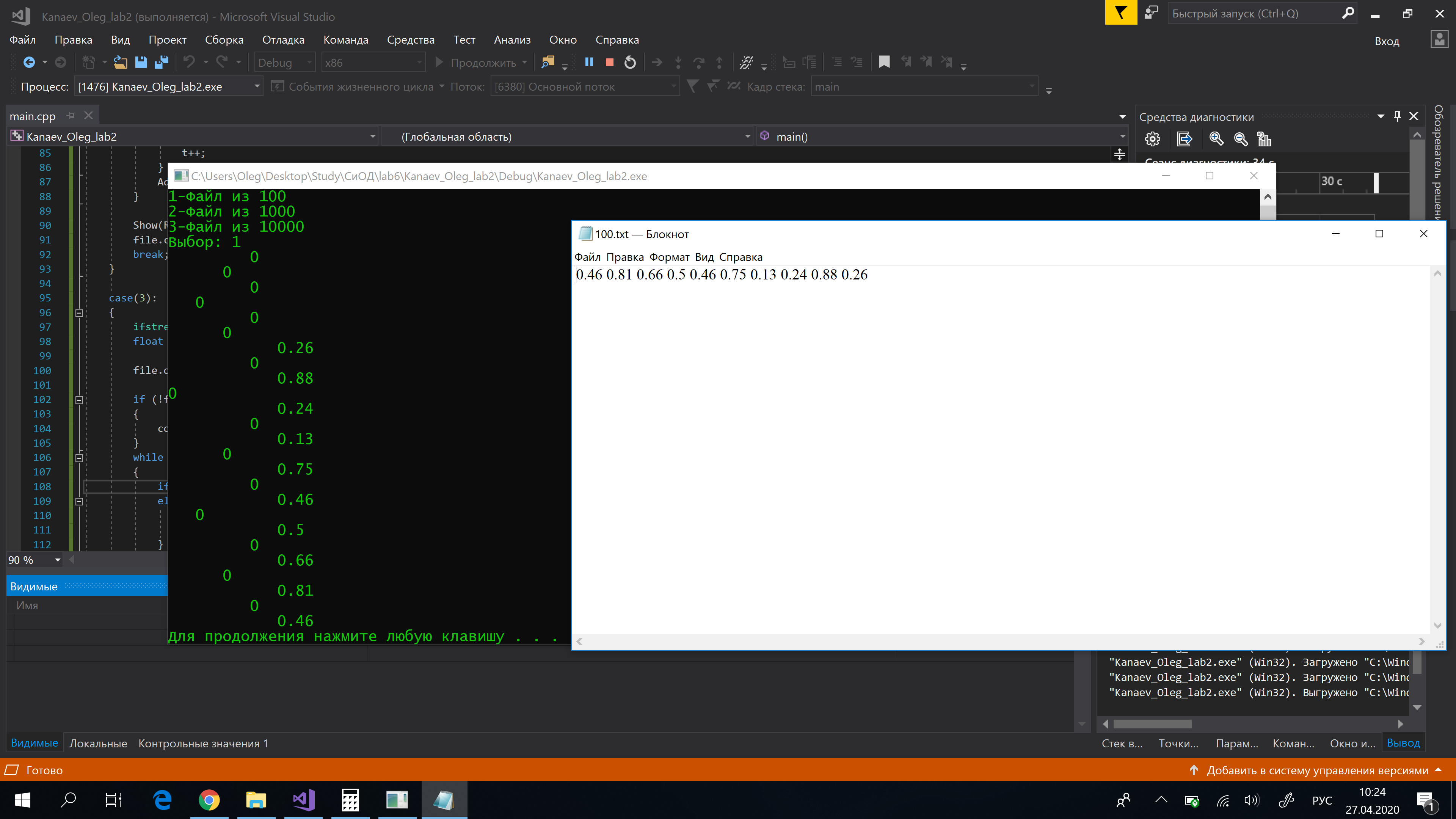


Рисунок №3 —Данные из файла 10.txt, представленные в листьях дерева

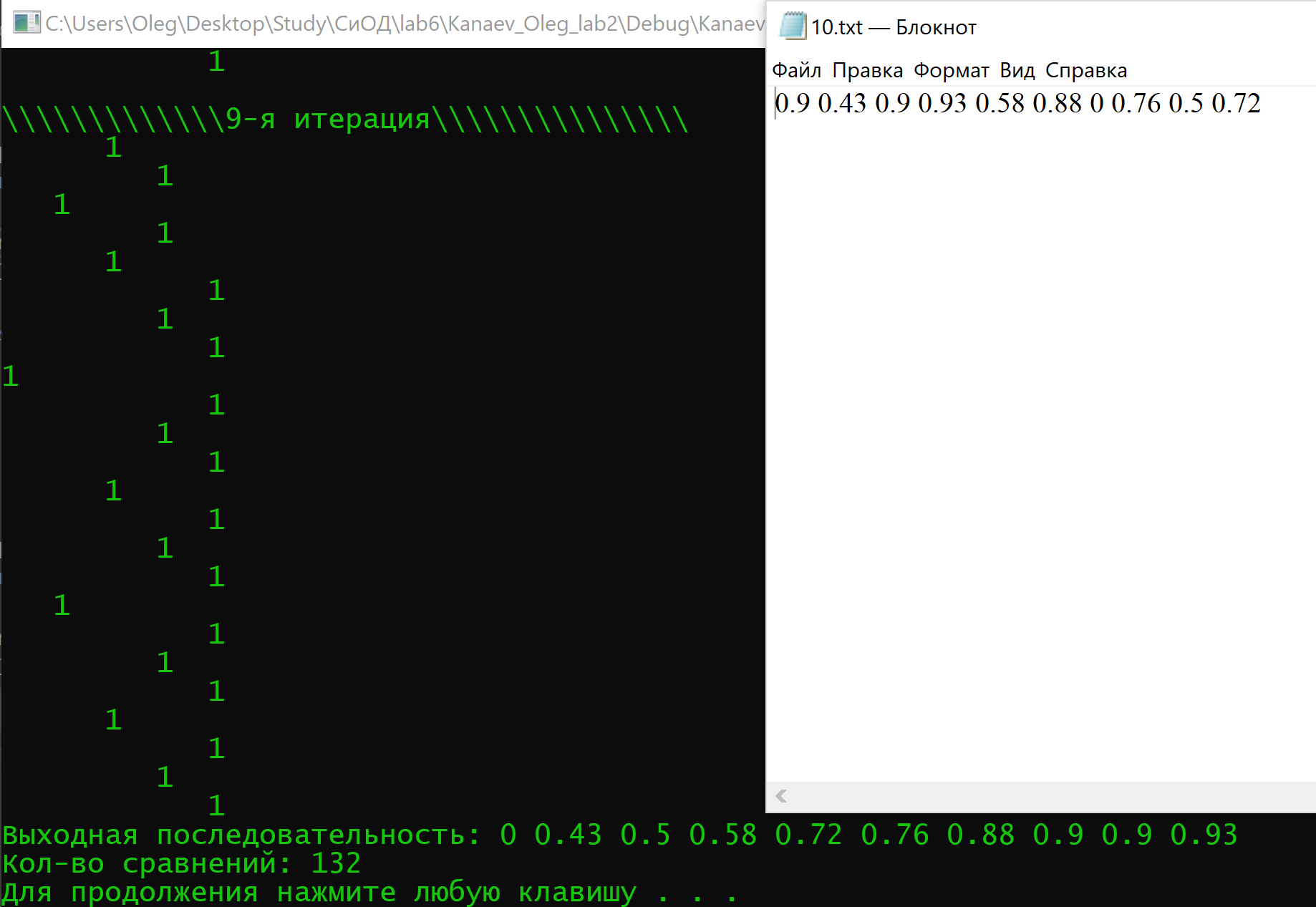


Рисунок №4 — Результат работы с файлом 10.txt

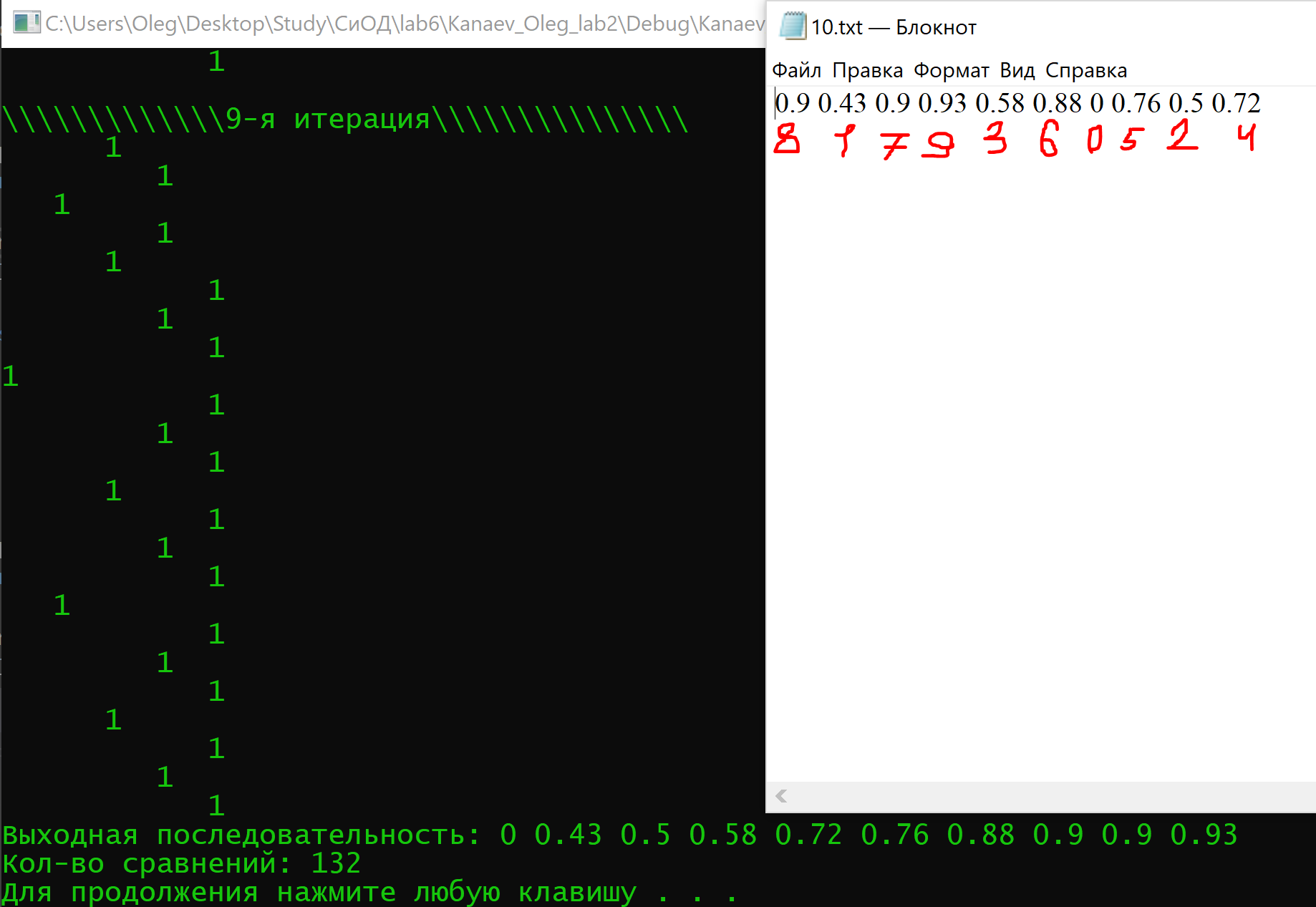


Рисунок №5 — Результат работы с файлом 10.txt

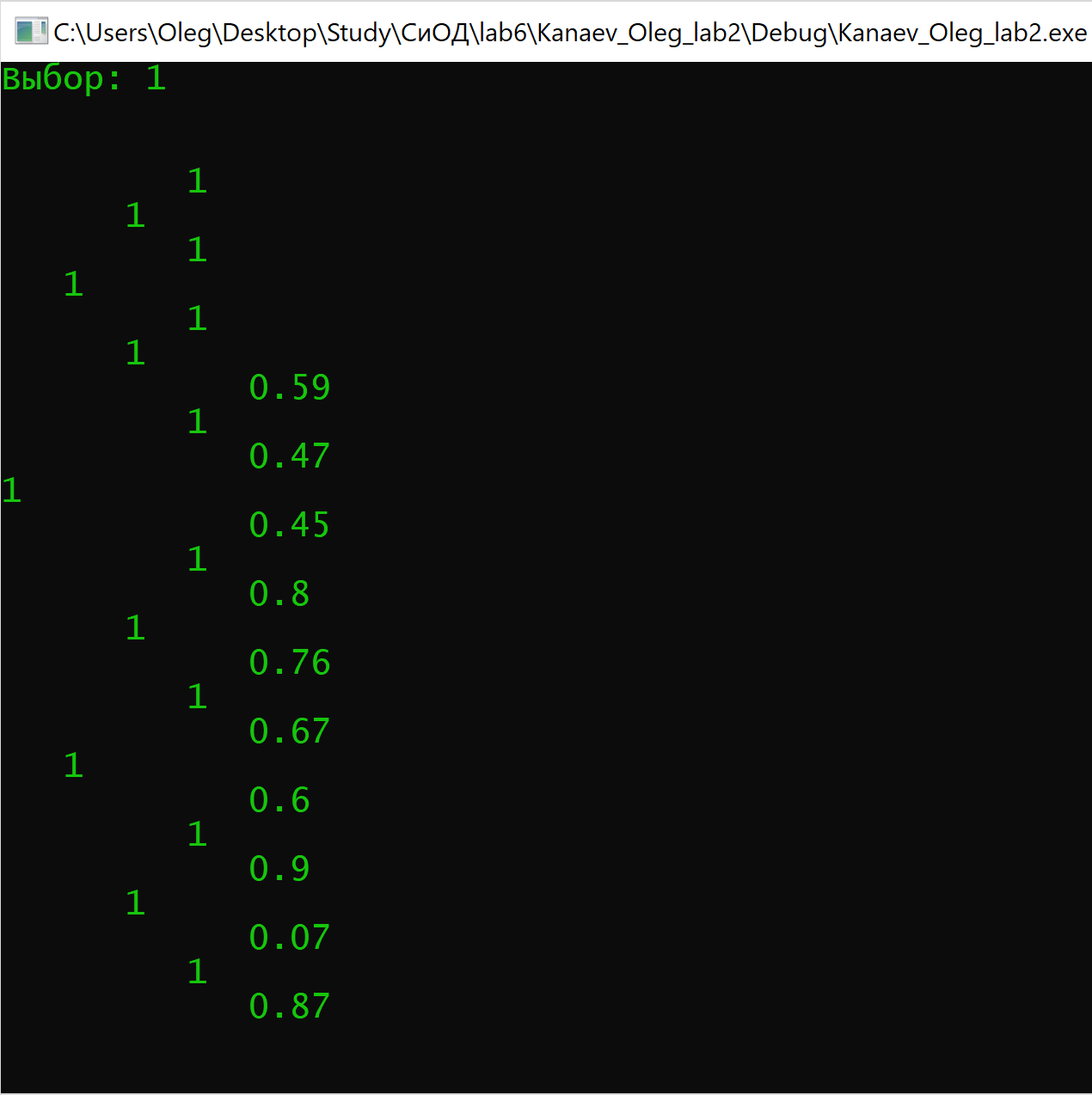


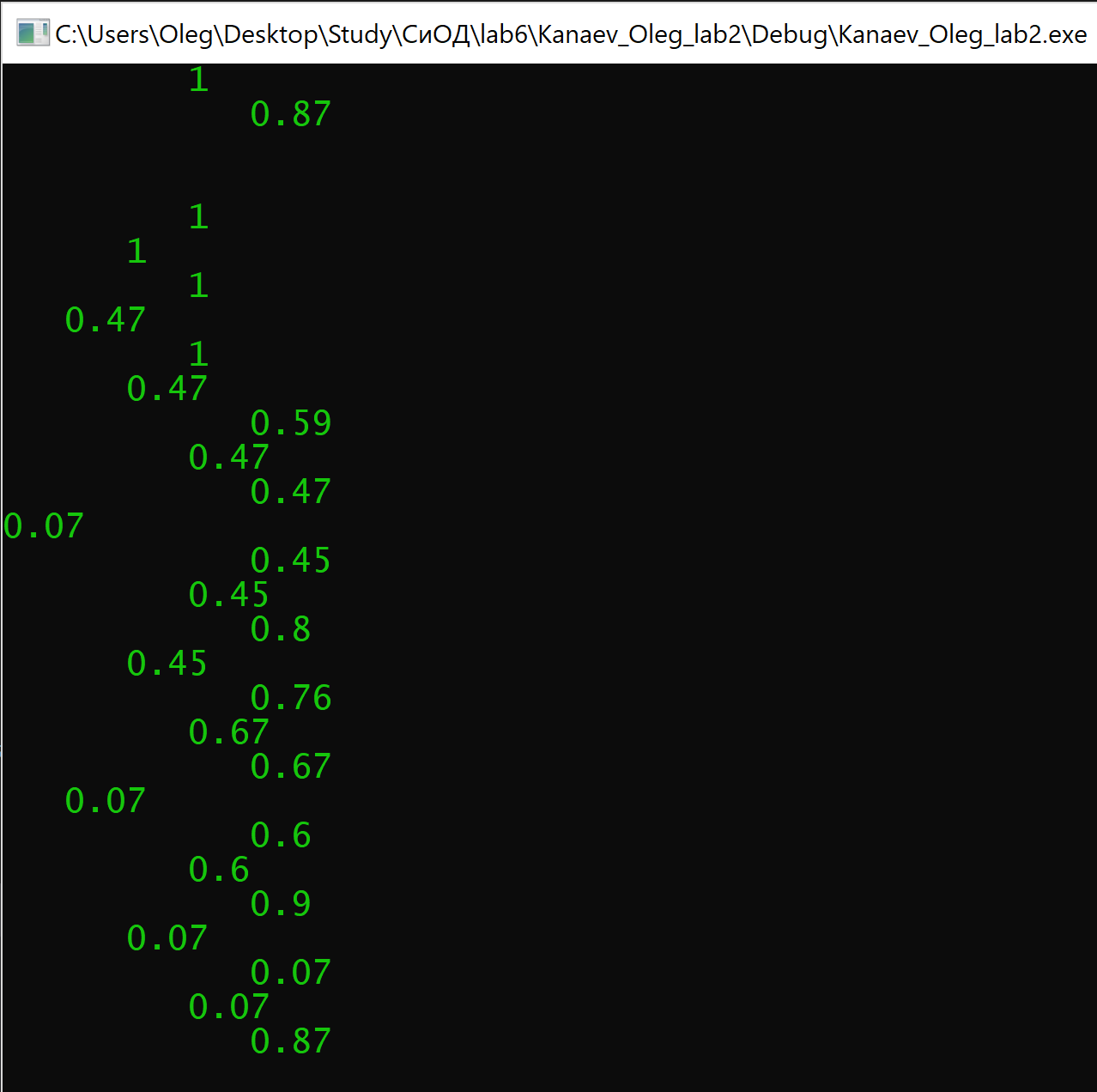
Рисунок №6 — Первый этап: входной массив в листьях дерева. 

Рисунок №7 — Сравнние элементов и запись наименьшего в родителя.

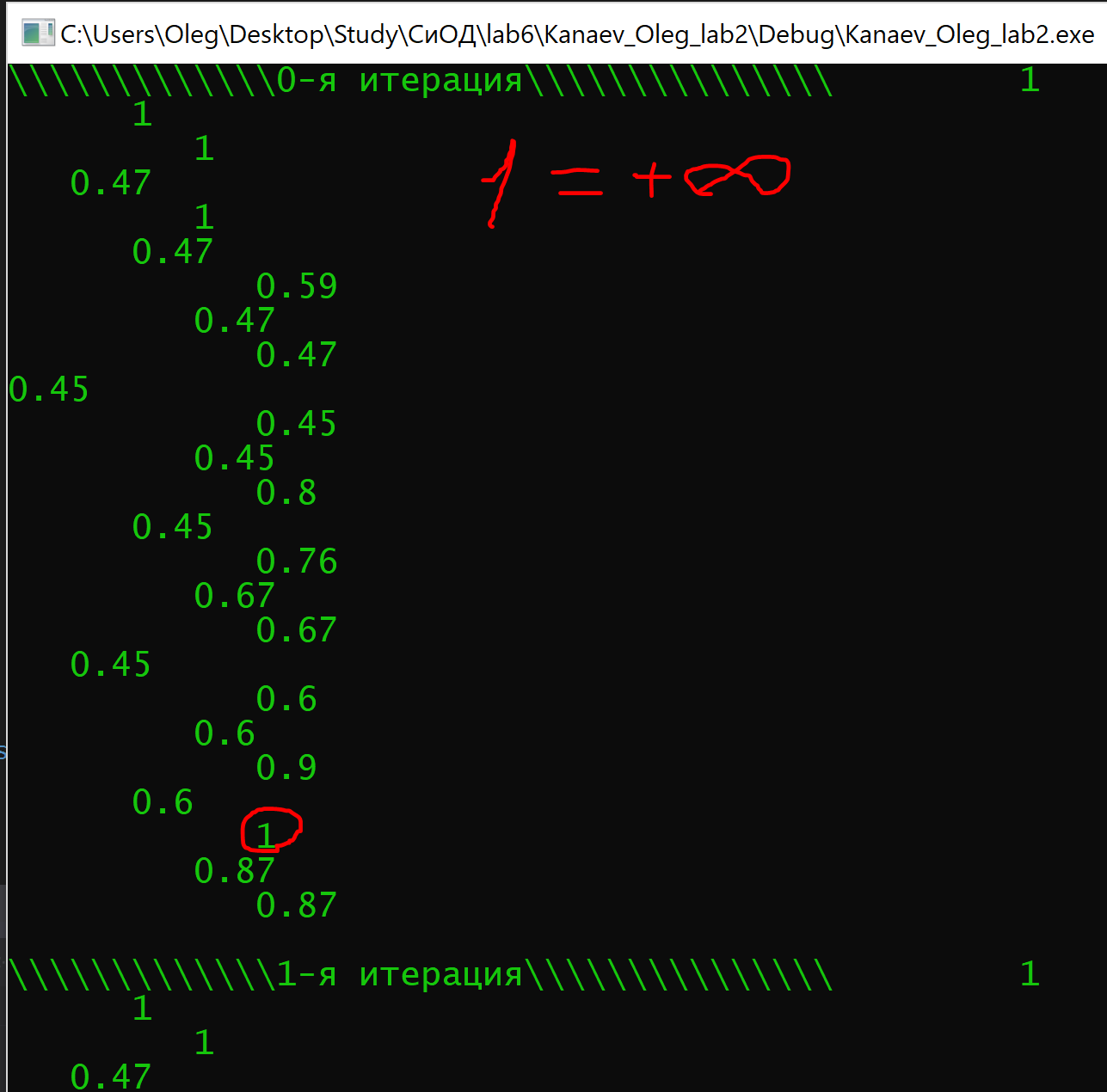


Рисунок №8 — 1-я итерация: запись корня в выходную последовательность и проход по наименьшему путю, записать 1 (+∞).

**Вывод:**Научился использовать алгоритм сортировки: турнирная сортировка. Вычислительная эффективность. Эффективность турнирной сортировки составляет O(n log2n). В массиве, содержащем n = 2k элементов, для выявления наименьшего элемента требуется n-1 сравнений.