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

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Усовершенствованные алгоритмы сортировки массивов
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 усовершенствованных алгоритмов сортировки массивов.

2 Задачи

Написать программу, реализующую шейкерный алгоритм сортировки массива и алгоритм сортировки массива Шелла.

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).

3 Описание реализованного алгоритма

Реализованы алгоритмы шейкерный алгоритм сортировки массива и алгоритм сортировки массива Шелла.

4 Описание программы

Для решения задачи была написана программа на языке С. Ниже приведен листинг кода.

Листинг 1 – Код программы

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

enum Constants
{
    INPUT_SIZE = 100
};

void SwapInt(int* a, int* b)
{
    int box;
    box = *b;
    *b = *a;
    *a = box;
}

int CycleInputInt(char* stringToOutput, bool(* pChecker)(int))
{
    int number;
    int position;
    char input[INPUT_SIZE];

    while (true)
    {
        printf("%s\n", stringToOutput);
        fflush(stdout);
        char* fgetsRet = fgets(input, INPUT_SIZE, stdin);
        if (fgetsRet == NULL)
        {
            printf("Wrong format!\n");
            continue;
        }
        int inputLength = strlen(input) - 1;
        input[inputLength] = '\0';
        int sscanfRet = sscanf(input, "%d%n", &number, &position);
        if (position != inputLength)
        {
            printf("Wrong format!\n");
            continue;
```

Продолжение листинга 1

```
        }
        if (pChecker && !pChecker(number))
        {
            printf("Wrong format!\n");
            continue;
        }
        if (sscanfRet == 1) break;
        printf("Wrong format!\n");
    }
    return number;
}

bool ArraySizeInputChecker(int arraySize)
{
    return arraySize > 0;
}

bool AnyIntInputChecker(int _)
{
    return true;
}

typedef struct
{
    int* content;
    int size;
} DynArray;

void PrintArray(DynArray* object)
{
    for (int i = 0; i < object->size; i++)
    {
        printf("%d ", object->content[i]);
    }
    printf("\n");
}

void ArrayCopy(DynArray* origin, DynArray* object)
{
    if (object->size != 0)
    {
```

Продолжение листинга 1

```
        free(object->content);
    }
    object->size = origin->size;
    object->content = (int*) malloc(object->size * sizeof(int));
    for (int i = 0; i < object->size; i++)
    {
        object->content[i] = origin->content[i];
    }
}

void ShakerSort(DynArray* object)
{
    int left;
    left = 0;
    int right;
    right = object->size - 1;
    while(left <= right)
    {
        for (int i = left; i < right; i++)
        {
            if(object->content[i] > object->content[i+1])
            {
                SwapInt(&object->content[i], &object->content[i+1]);
            }
        }
        right--;
        for (int i = right; i > left; i--)
        {
            if(object->content[i-1] > object->content[i])
            {
                SwapInt(&object->content[i], &object->content[i-1]);
            }
        }
        left++;
    }
}

void ShellSort(DynArray* object)
{
    for (int h = object->size/3 ? object->size/3 : 1; h > 0; h /= 2)
        for (int n = 0; n < h; n++)
```

Продолжение листинга 1

```
        {
            for (int i = n + h;
                 i < object->size; i += h)
            {
                int box;
                box = object->content[i];
                int j = n;
                while (j < i && box > object->content[j])
                {
                    j += h;
                }
                for (int k = i; k > j; k -= h)
                {
                    object->content[k] = object->content[k - h];
                }
                object->content[j] = box;

            }
        }
}

int main()
{
    DynArray object;
    object.size = CycleInputInt("Enter size of array",ArraySizeInputChecker);
    object.content = (int*) malloc(object.size * sizeof(int));
    printf("Enter elements, one by one\n");
    for (int i = 0; i < object.size; i++)
    {
        object.content[i] = CycleInputInt("Enter next element",
                                          AnyIntInputChecker);
    }

    printf("Origin array:\n");
    PrintArray(&object);
    DynArray copied;

    printf("Shell sorted:\n");
    ArrayCopy(&object, &copied);
    ShellSort(&copied);
    PrintArray(&copied);
```

Окончание листинга 1

```
    printf("Shaker sorted:\n");
    ArrayCopy(&object, &copied);
    ShakerSort(&copied);
    PrintArray(&copied);

    free(object.content);
    free(copied.content);
}
```

5 Результаты работы программы

На следующем рисунке приведен скриншот с результатами работы программы.

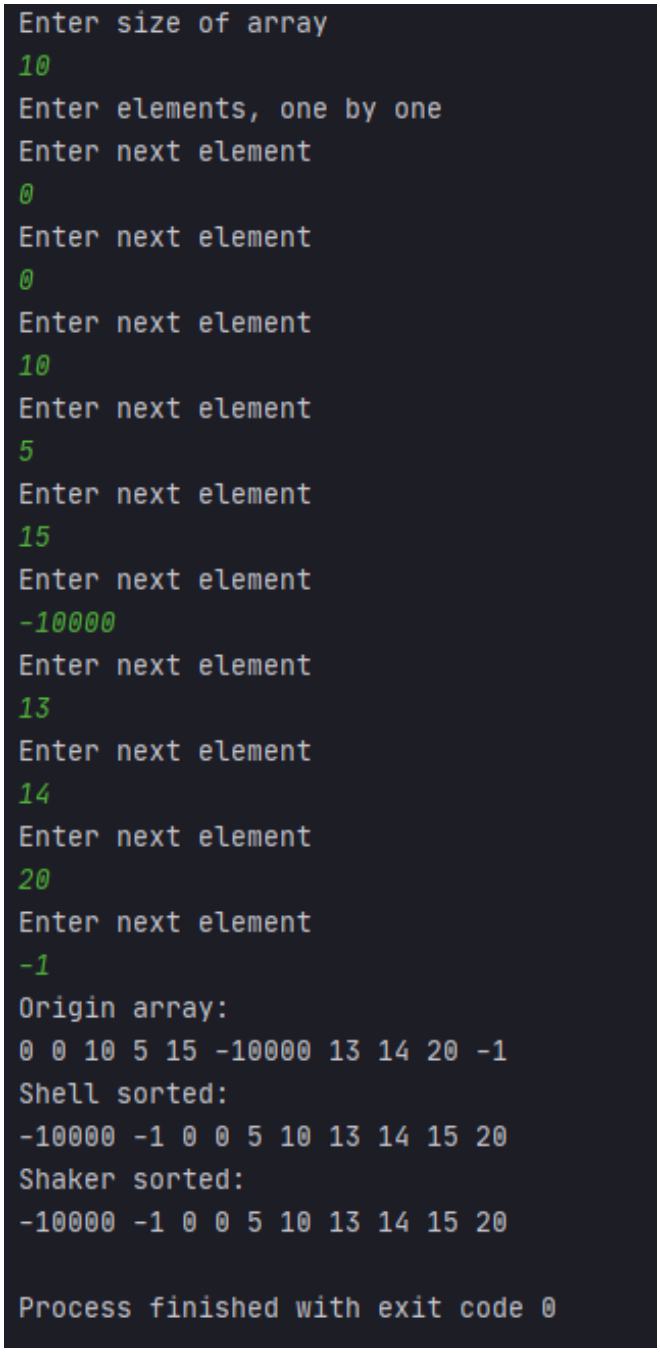

Рисунок 1 – Результаты работы программы