

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

*Институт Принтмедиа и информационных технологий
Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 6-7

Дисциплина: Основы алгоритмизации и программирования.

Тема: Алгоритм сортировки «расческа»

Выполнила:

студентка группы 201-723

Круглова А.М.

02.11.20

(Дата)

(Подпись)

Проверил: преп. Хуснулина Д.Р.

(Дата)

(Подпись)

Замечания: _____

Москва

2020

Оглавление

Цель	3
Постановка задачи.....	3
Идея алгоритма	3
Словесное представление алгоритма	4
Блок-схема с использованием элемента модификации.....	5
Листинг программы с использованием параметрического цикла	6
Результат работы программы.....	7
Блок-схема без использования элемента модификации.....	8
Листинг программы с использованием цикла с предусловием	9
Результат работы программы.....	10

Цель:

Получить практические навыки разработке алгоритмов и их программной реализации.

Постановка задачи:

Необходимо выполнить и оформить описание следующих пунктов:

1. Сформулировать идею алгоритма
2. Выполнить словесное представление алгоритма
3. Выполнить представление алгоритма с помощью блок схем с использованием элемента модификации и без него.
4. Выполнить программную реализацию алгоритмов на языке C с использованием параметрического цикла и цикла с предусловием.

Идея алгоритма:

Алгоритм является модификацией «пузырька». Отличие алгоритмов состоит в том, что сравниваются не соседние элементы, а отстоящие друг от друга на определённую величину, или шаг (назовём его $step$). Алгоритм реализован с помощью двух циклов. Окончание внешнего цикла (и алгоритма) происходит тогда, когда $step$ станет меньше 1. На первой итерации расстояние ($step$) максимально возможное (размер массива – 1), а на последующих итерациях оно изменяется по формуле $step /= k$ (дробная часть отбрасывается). k – это фактор уменьшения, константа, равная 1.2473309 (при написании программы можно использовать примерное значение, равное 1.247). Во внутреннем цикле движение происходит от начала к концу, перемещаясь на $step$. Если значение текущего элемента больше, чем значение элемента через $step$ шагов от текущего, то сравниваемые элементы меняются местами. Условием продолжения цикла является условие $i < n - step$ (где i – номер текущего элемента).

Словесное представление алгоритма:

array – массив, n – длина массива, k – фактор уменьшения, равный 1.247, step – шаг

1. расчет шага ($\text{step} = n - 1$)
2. если $\text{step} \geq 1$, то п.3, иначе п.10
3. параметр внутреннего цикла $i = 0$
4. если $\text{step} \geq 1$, то п.3, иначе переход к п.10
5. если $i < n - \text{step}$, то п.6, иначе п.9
6. если $\text{array}[i] > \text{array}[i + \text{step}]$, то п.7, иначе п.8
7. перестановка $\text{array}[i]$ и $\text{array}[i + \text{step}]$
8. $i++$, п.4
9. $\text{step} /= k$, п.2
10. конец алгоритма

Блок-схема с использованием элемента модификации:

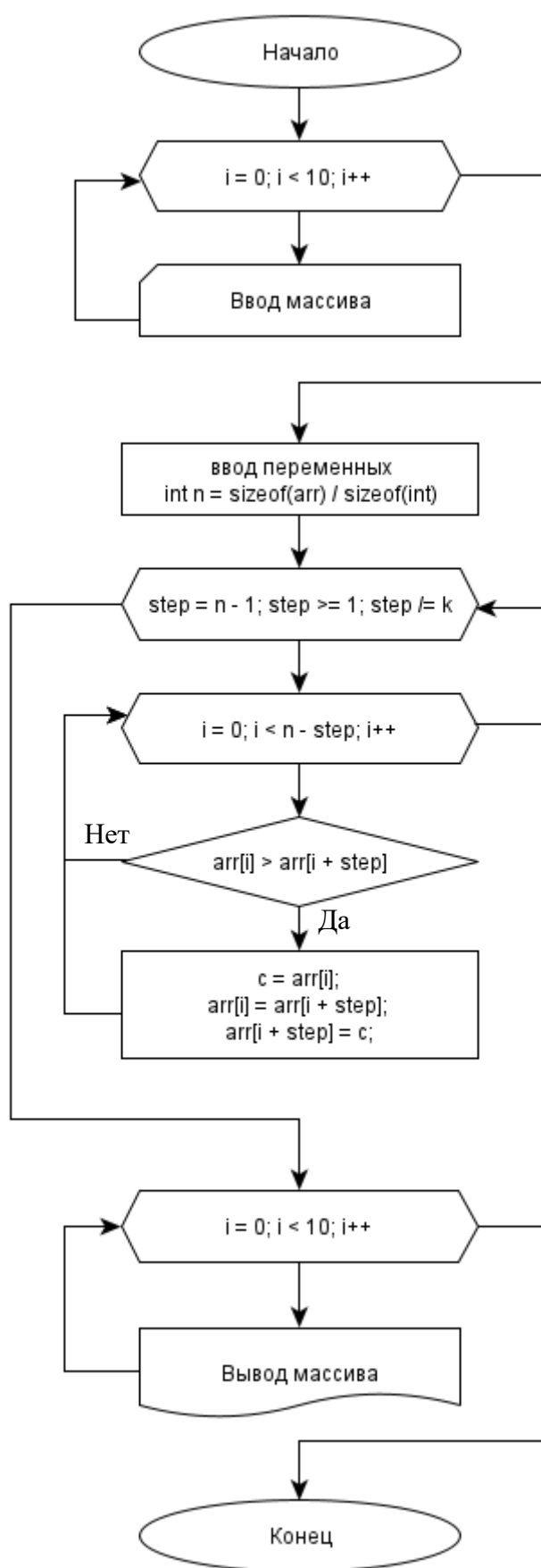


Рисунок 1 — Блок-схема 1 программы

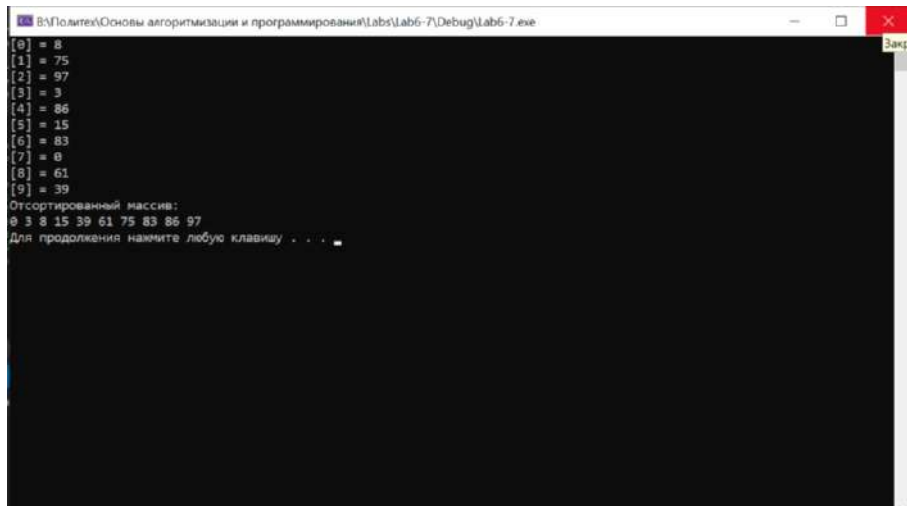
Листинг программы с использованием параметрического цикла:

Листинг 1 — Исходный код 1 программы

```
#include <iostream>
#include <stdlib.h>

int main()
{
    setlocale(LC_ALL, "Russian"); //установка русского языка
    int i, step, c;
    float k = 1.247; //фактор уменьшения
    int arr[10]; // Объявляем массив из 10 элементов
    for (i = 0; i < 10; i++) // Вводим значения элементов массива через
клавиатуру
    {
        printf("[%d] = ", i);
        scanf_s("%d", &arr[i]);
    }
    int n = sizeof(arr) / sizeof(int); //размер массива
    for (step = n - 1; step >= 1; step /= k) //внешний цикл
    {
        for (i = 0; i < n - step; i++) //внутренний цикл
        {
            if (arr[i] > arr[i + step]) //условие, если текущий элемент больше
другого
            {
                c = arr[i];
                arr[i] = arr[i + step]; //меняем их местами
                arr[i + step] = c;
            }
        }
    }
    // Выводим отсортированные элементы массива
    printf("Отсортированный массив:\n");
    for (i = 0; i < 10; i++)
        printf("%d ", arr[i]);
    printf("\n");
    system("pause");
    return 0;
}
```

Результат работы программы:



```
8:\Уолитех\Основы алгоритмизации и программирования\Лabs\Lab6-7\Debug\Lab6-7.exe
[0] = 8
[1] = 75
[2] = 97
[3] = 3
[4] = 86
[5] = 15
[6] = 83
[7] = 0
[8] = 61
[9] = 39
Отсортированный массив:
0 3 8 15 39 61 75 83 86 97
Для продолжения нажмите любую клавишу . . .
```

Рисунок 2 — Результат работы 1 программы

Блок-схема без использования элемента модификации:

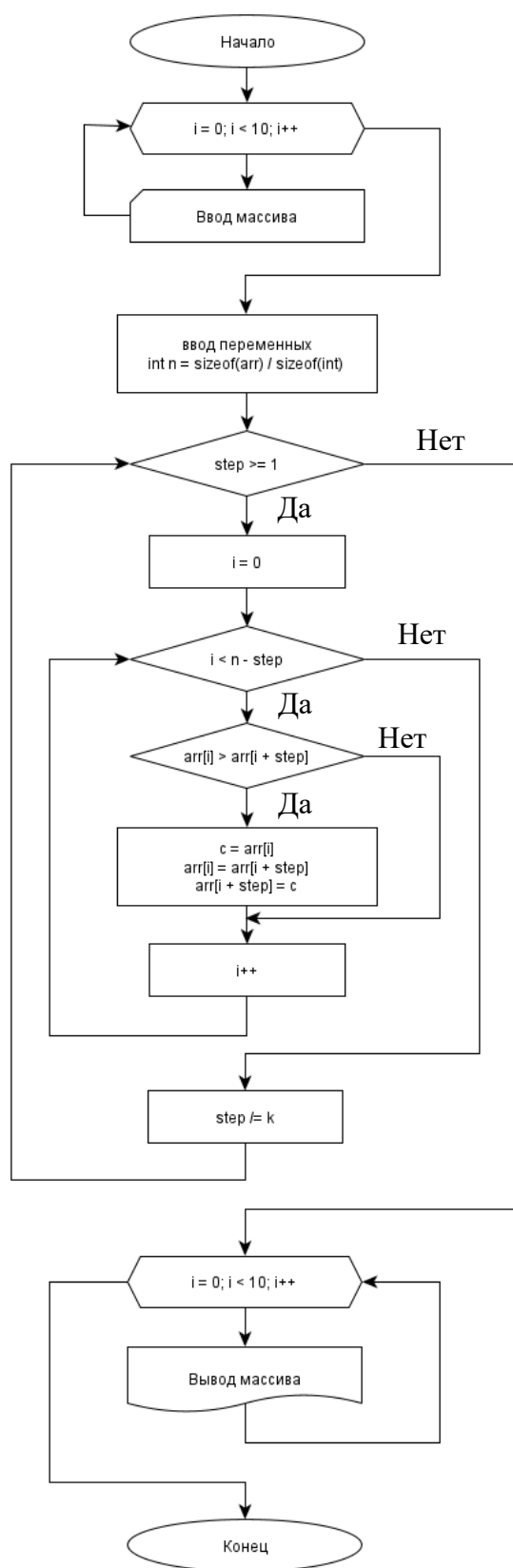


Рисунок 3 — Блок-схема 2 программы

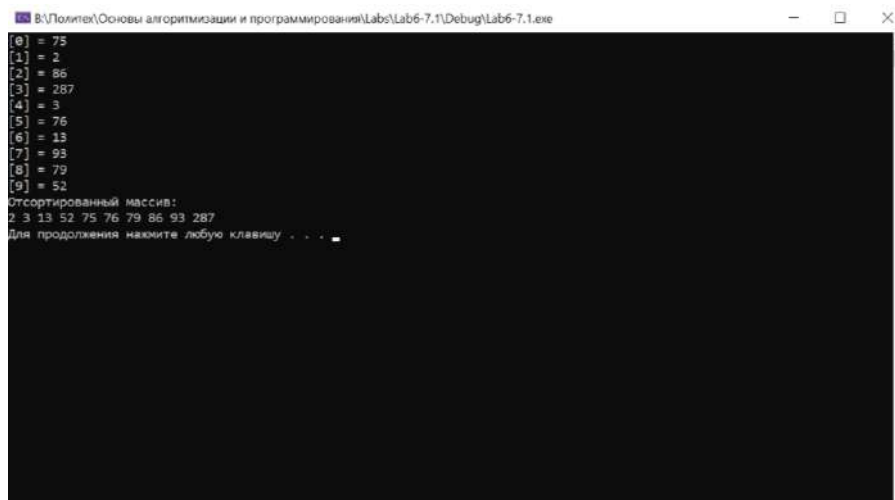
Листинг программы с использованием цикла с предусловием:

Листинг 2 — Исходный код 2 программы

```
#include <iostream>
#include <stdlib.h>

int main()
{
    setlocale(LC_ALL, "Russian"); //установка русского языка
    int i, c;
    float k = 1.247; //фактор уменьшения
    int arr[10]; // Объявляем массив из 10 элементов
    for (i = 0; i < 10; i++) // Вводим значения элементов массива через
клавиатуру
    {
        printf("[%d] = ", i);
        scanf_s("%d", &arr[i]);
    }
    int n = sizeof(arr) / sizeof(int); //размер массива
    int step = n - 1;
    while (step >= 1) //начало внешнего цикла
    {
        i = 0;
        while (i < n - step) //начало внутреннего цикла
        {
            if (arr[i] > arr[i + step]) //условие, если текущий элемент больше
другого
            {
                c = arr[i];
                arr[i] = arr[i + step]; //меняем местами
                arr[i + step] = c;
            }
            i++;
        }
        step /= k;
    }
    // Выводим отсортированные элементы массива
    printf("Отсортированный массив:\n");
    for (i = 0; i < 10; i++)
        printf("%d ", arr[i]);
    printf("\n");
    system("pause");
    return 0;
}
```

Результат работы программы:



```
8:\Политех\Основы алгоритмизации и программирования\Lab6-7.1\Debug\Lab6-7.1.exe
[0] = 75
[1] = 2
[2] = 86
[3] = 287
[4] = 3
[5] = 76
[6] = 13
[7] = 93
[8] = 79
[9] = 52
Отсортированный массив:
2 3 13 52 75 76 79 86 93 287
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 — Результат работы 2 программы