



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет 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»"**

**Выполнил: студент группы 211-721**

**Дерендяев Дмитрий Сергеевич**  
(Фамилия И.О.)

**Дата, подпись 27.10.2021** \_\_\_\_\_  
(Дата) (Подпись)

**Проверил: Новичков Иван Константинович** \_\_\_\_\_  
(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись** \_\_\_\_\_  
(Дата) (Подпись)

**Замечания:**

---

---

---

**Москва**  
**2021**

# Лабораторная работа №6-7

## "Алгоритм сортировки «расческа»"

(продолжительность 4 часа)

**Цель:** Получить практические навыки разработке алгоритмов и их программной реализации.

### Идея алгоритма:

**Сортировка расчёской** (англ. *comb sort*) — это довольно упрощённый алгоритм сортировки, изначально спроектированный в 1980 г.

Сортировка расчёской улучшает сортировку пузырьком, и конкурирует с алгоритмами, подобными быстрой сортировке. Основная идея — устранить *черепашки*, или маленькие значения в конце списка, которые крайне замедляют сортировку пузырьком.

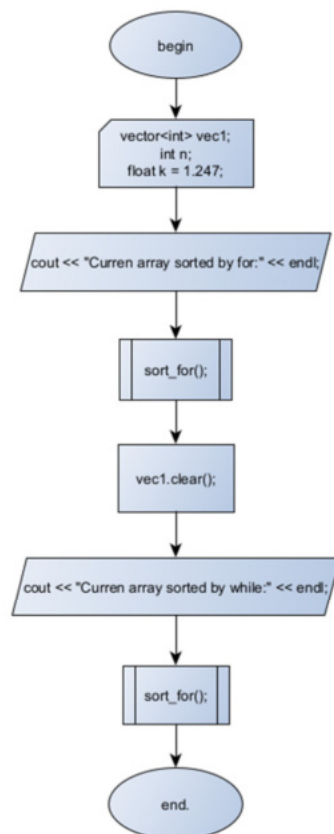
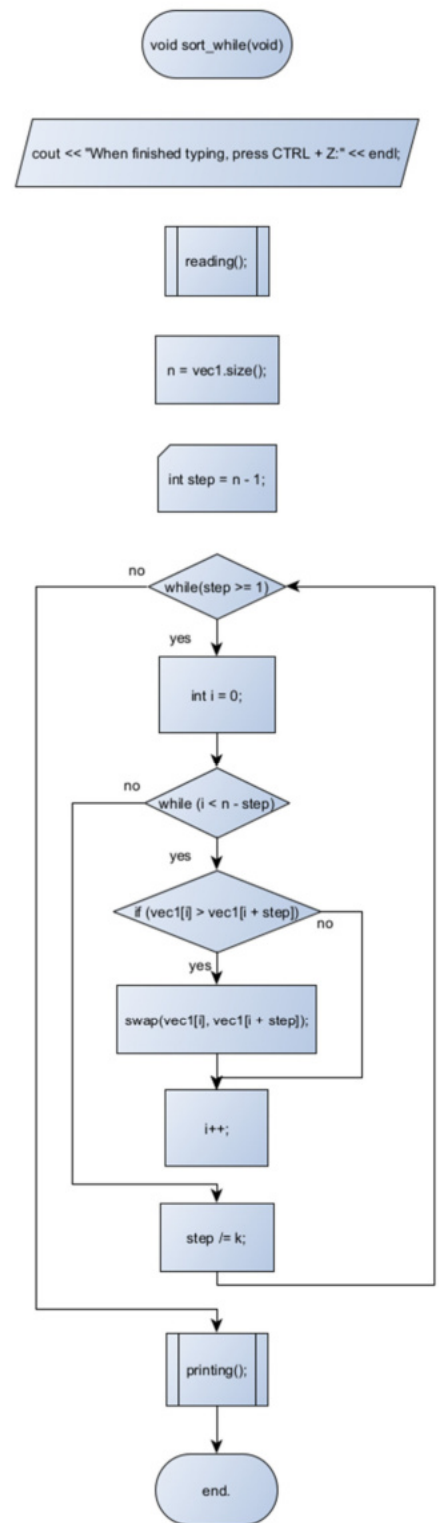
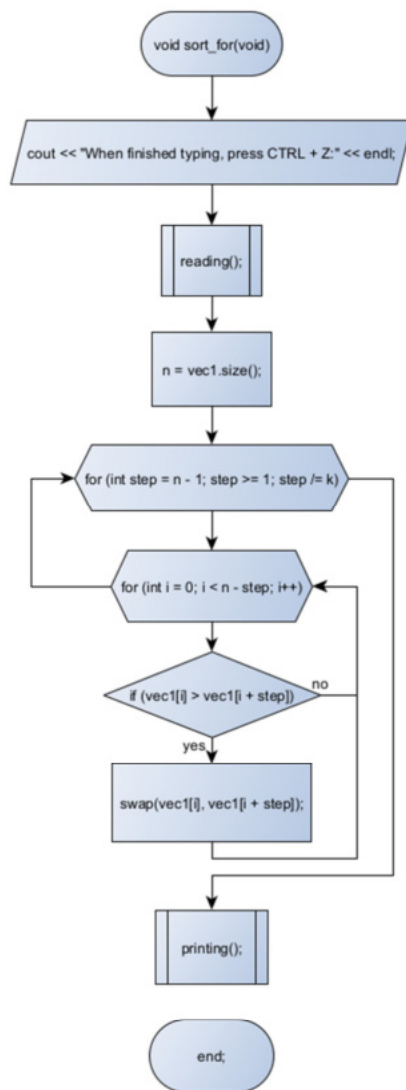
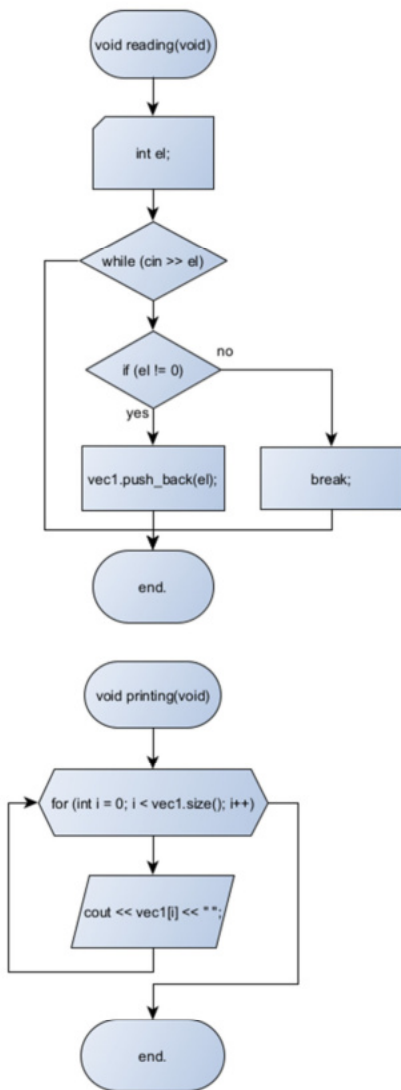
В сортировке пузырьком, когда сравниваются два элемента, промежуток (расстояние друг от друга) равен 1. Основная идея сортировки расчёской в том, что этот промежуток может быть гораздо больше, чем единица.

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).

### Задачи:

Необходимо выполнить и оформить описание следующих пунктов:

1. Сформулировать идею алгоритма
2. Выполнить словесное представление алгоритма
3. Выполнить полнить 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.



### Словесное описание алгоритма:

vec1 – массив(вектор), n = vec1.size(); - размер массива; k – коэффициент уменьшения step; step - шаг;

1. Расчет шага step = n – 1
2. Если step >= 1: to п.3; else: п.10
3. Параметр внутреннего цикла i = 0
4. Если step >= 1, то п.3, иначе переход к п.10
5. Если i < n – step, то п.6, иначе п.9
6. Если vec1[i] > vec1[i + step], то п.7, иначе п.8
7. Перестановка vec1[i] и vec1[i + step]
8. i++, п.4
9. step/=k, п.2
10. конец алгоритма

### Листинг программы:

```
#include <iostream> //подключение необходимых библиотек
#include <vector>
#include <algorithm>

using namespace std; //объявление стандартного пространства имен

vector<int> vec1; //объявление переменных
int n;
float k = 1.247;

void reading(void) //инициализации функции заполнения вектора(с клавиатуры)
{
    int el; //объявление промежуточной переменной, куда будет буферно записываться
    переменная для очередного элемента вектора
    while (cin >> el) //начало цикла while и работа его до тех пор, пока вводятся
    значения с клавиатуры(Stop: Ctrl+Z) - НЕ РЕКОМЕНДУЕТСЯ
    {
        if (el != 0) //конец строки для определенной задачи по символу 0
            vec1.push_back(el); //внесение очередного элемента в конец вектора
        else
            break; //если был нажат 0, то выход из подпрограммы записи вектора
    }
}

void printing(void) //инициализация функции печати/вывода элементов на экран
{
    for (int i = 0; i < vec1.size(); i++) //цикл вывода элементов на экран с индексом от
    начала до конца вектора(по размеру массива)
    {
        cout << vec1[i] << " "; //печать значения
    }
    cout << endl; //перенос строки
}

void sort_for(void) //инициализация функции сортировки массива с помощью цикла for
{
    cout << "When finished typing, press CTRL + Z or enter 0 to contine next task:" <<
    endl; //предупреждение
    reading(); //вызов функции записи вектора
    n = vec1.size(); //определение размера массива
    for (int step = n - 1; step >= 1; step /= k) //инициализация цикла, который
    выполняется пока шаг больше 1, при уменьшении шага на коэфф K
    {
        for (int i = 0; i < n - step; i++)
        {
```

```

        if (vec1[i] > vec1[i + step])//если первый элемент больше элемента
+step произвести обмен
            swap(vec1[i], vec1[i + step]);//выполнение перемещения
элементов вектора по индексам, заданным в условии
    }
    printing();//вывод вектор на экран, использую подпрограмму
}

void sort_while(void)//инициализация функции сортировки массива с помощью цикла while
{
    cout << "When finished typing, press CTRL + Z:" << endl;//предупреждение
    reading();//вызов функции записи вектора
    n = vec1.size();//определение размера массива
    int step = n - 1;//инициализация первого шага(крайняя точка массива)
    while(step >= 1)//инициализация внешнего цикла while до момента, пока step больше 1
    {
        int i = 0;//инициализация счетчика внутреннего цикла
        while (i < n - step)//внутренний цикл выполнит функцию прохода от начала
вектора до конца(в последствии размер массива - шаг)
        {
            if (vec1[i] > vec1[i + step])//если первый элемент больше элемента
+step произвести обмен
                swap(vec1[i], vec1[i + step]);//выполнение перемещения
элементов вектора по индексам, заданным в условии
                i++;
        }
        step /= 2;//уменьшение шага
    }
    printing();//вывод вектор на экран, использую подпрограмму
}

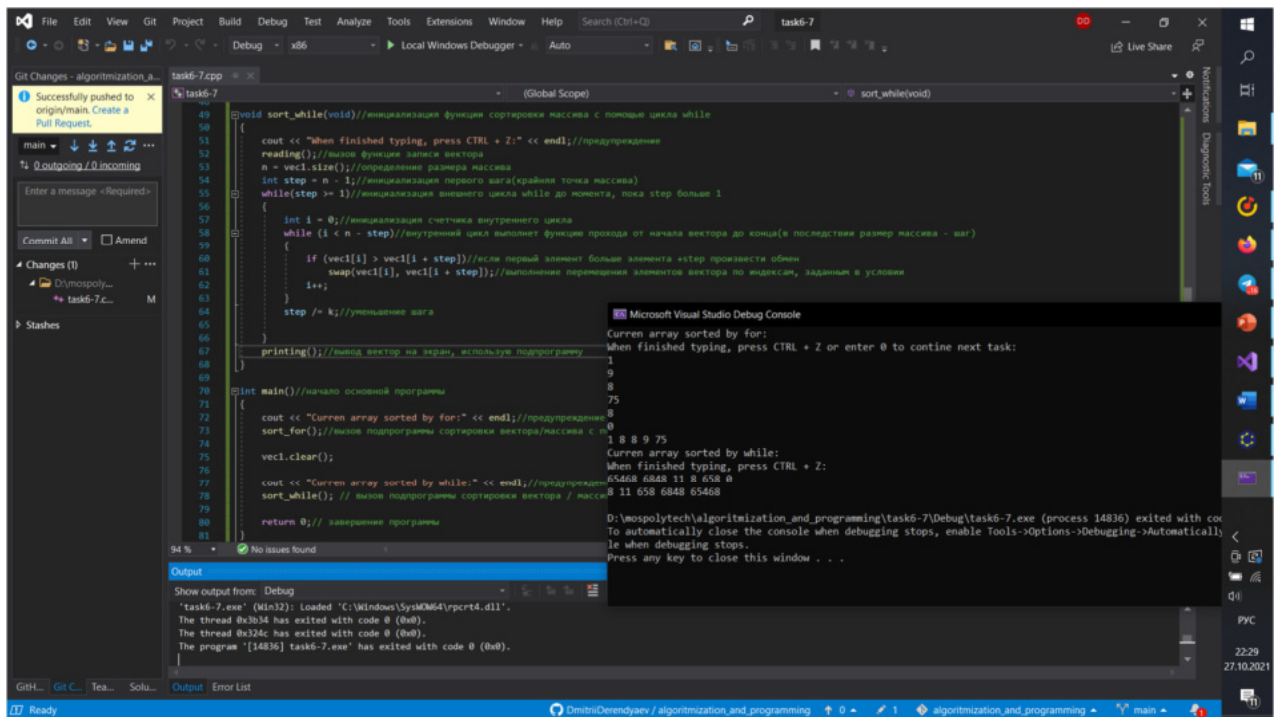
int main();//начало основной программы
{
    cout << "Current array sorted by for:" << endl;//предупреждение
    sort_for();//вызов подпрограммы сортировки вектора/массива с помощью цикла for

    vec1.clear();

    cout << "Current array sorted by while:" << endl;//предупреждение
    sort_while(); // вызов подпрограммы сортировки вектора / массива с помощью цикла
while

    return 0;// завершение программы
}

```



При необходимости, вы можете найти всю историю разработки программы на моем GitHub:

[https://github.com/DmitriDerendyaev/algoritimization\\_and\\_programming](https://github.com/DmitriDerendyaev/algoritimization_and_programming)