**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий**

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 10-11**

**Дисциплина:** Основы алгоритмизации и программирования

# Тема: "Алгоритм сортировки «Шелла»"

# Выполнил: студент группы 211-721

Дерендяев Дмитрий Сергеевич

(Фамилия И.О.)

**Дата, подпись** 1.12.2021  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: Новичков Иван Константинович *\_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2021**

# Лабораторная работа №10-11 "Алгоритм сортировки «вставками»"

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

**Понятие алгоритма:  
Сортировка Шелла** (англ. *Shell sort*) — алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками. Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. Иными словами — это сортировка вставками с предварительными «грубыми» проходами. Аналогичный метод усовершенствования пузырьковой сортировки называется сортировка расчёской.

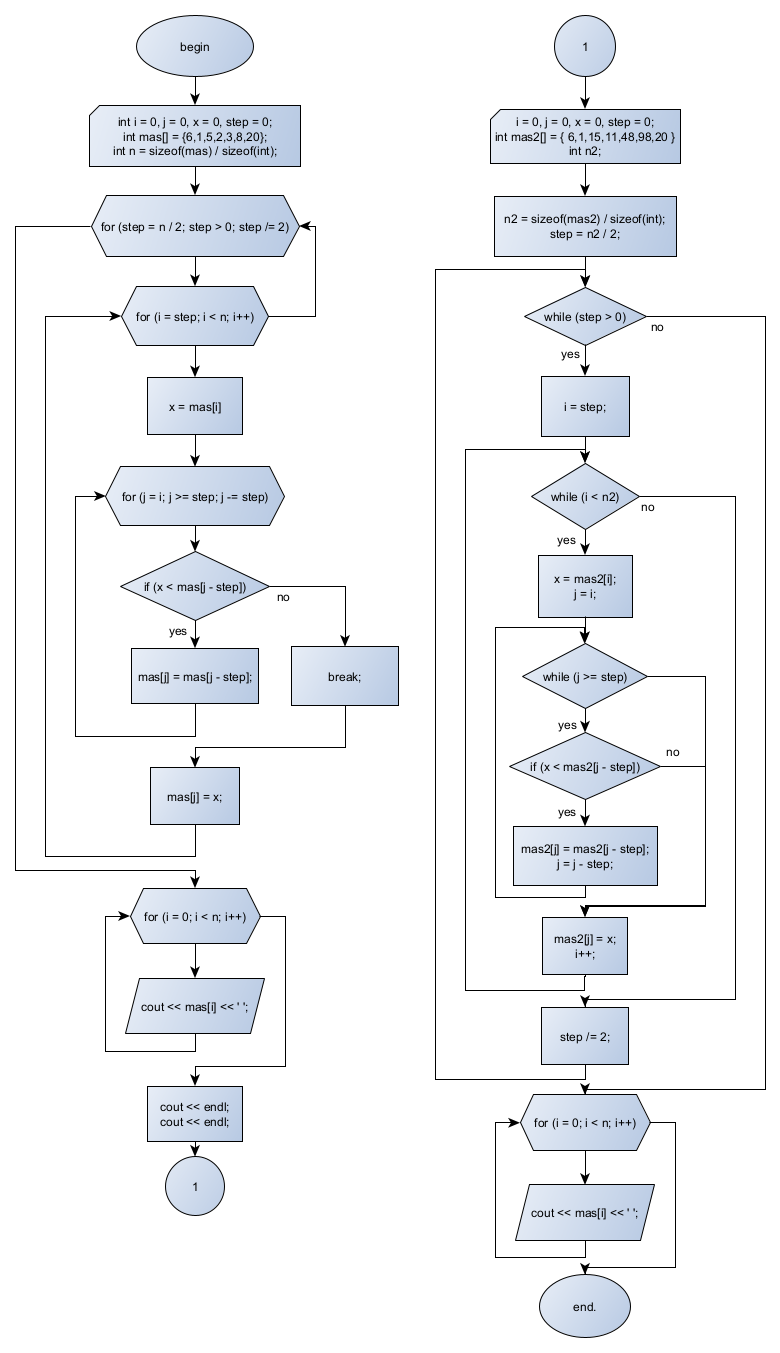
**Идея алгоритма:**

Алгоритм сортирует  элементы отстоящие друг от друга на некотором расстоянии. Затем сортировка повторяется при меньших значениях шага, и в конце процесс сортировки Шелла завершается при шаге, равном 1 (а именно обычной сортировкой вставками). Шелл предложил такую последовательность размера шага: N/2, N/4, N/8 …, где N – количество элементов в сортируемом массиве.

**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 С с использованием параметрического цикла и цикла с предусловием.

****

**Словесное описание алгоритма:  
n,n2 – длина массива, step – шаг**

1. **Рассчитываем начальное значение шага: step = n / 2**
2. **Если step > 0, то п.3, иначе п. 14**
3. **Номер анализ. эл-та = step**
4. **Если номер анализ. элемента < n, то п.5, иначе п.13**
5. **Запоминаем значение анализ. элемента**
6. **Номер текущего элемента = номеру анализ. элемента**
7. **Если номер текущего элемента >= step, то п.8, иначе п.11**
8. **Если значение текущего элемента < значение элемента с номером (текущего элемента – step), то п.9, иначе п.11**
9. **Значение т.э. = значение эл-та с номером (т.э – step)**
10. **Номер т.э = номер (т.э – step), п. 7**
11. **Значение т.э = значение анализ. элемента**
12. **i++, п. 4**
13. **step /= 2, п. 2**
14. **Конец алгоритма**

**Листинг программы:**

#include <iostream>//подключение необходимых библиотек

using namespace std;//определение пространства имен

int main()

{

int i = 0, j = 0, x = 0, step = 0;//объявление переменных, отвечающих за параметры внутри циклов

int mas[] = {6,1,5,2,3,8,20};//обьъявление радочего массива

int n = sizeof(mas) / sizeof(int);//получаем размер массива mas

for (step = n / 2; step > 0; step /= 2)//рассчитываем величину шага, изначально определяем его, как половину размера исходгого массива

{//выполняем массив до тех пор, пока шаг больше 0

for (i = step; i < n; i++)//начинаем

{

x = mas[i];//используем буферную переменную, куда помещаем значение анализируемого элемента

for (j = i; j >= step; j -= step)//номеру текущего элемента присваивается номер анализируемого элемента

{

if (x < mas[j - step])//сравниваем значение анализируемого элемента с элементом, отличающимся на шаг(по индексу)

{

mas[j] = mas[j - step];//ставим на место j-го элемента элемент с индексом j-шаг

}

else

break;//выходим из текущей итерации, если значение анализируемого элемента больше, чем значение элемента отличающегося на step

}

mas[j] = x;//возвращаем значение буфера

}

}

for (i = 0; i < n; i++)//организуем вывод массива на экран

{

cout << mas[i] << ' ';

}

cout << endl;

cout << endl;

//--------------------------------------------------------------------------

i = 0, j = 0, x = 0, step = 0;//подготовка переменных к обработке массива mas2 с помощью цикла while

int mas2[] = { 6,1,15,11,48,98,20 };

int n2 = sizeof(mas2) / sizeof(int);//получаем размер массива mas2

step = n2 / 2;//вычисляем размер шага

while (step > 0)//цикл выполянется, пока значение шага болльше нуля

{

i = step;// переходим к обработке элемента с номером шага

while (i < n2)//выполняем цикл, пока исследуемый элемент меньше размера массива

{

x = mas2[i];//в буферную переменную помещаем значение исслудуемого элемента

j = i;//присвоение номеру текущего элемента номер исследуемого элемента

while (j >= step)//выполнение цикла, пока номер исследуемого элемента больше шага step

{

if (x < mas2[j - step])//сравниваем значение анализируемого элемента с элементом, отличающимся на шаг(по индексу)

{

mas2[j] = mas2[j - step];////ставим на место j-го элемента элемент с индексом j-шаг

j = j - step;//ументшаем значение индекса j на величину шага

}

else

break;////выходим из текущей итерации, если значение анализируемого элемента больше, чем значение элемента отличающегося на step

}

mas2[j] = x;//возвращаем значение буфера

i++;//переходим к следующем элементу

}

step /= 2;//уменьшаем шаг

}

for (i = 0; i < n2; i++)//выаодим массив на экран

{

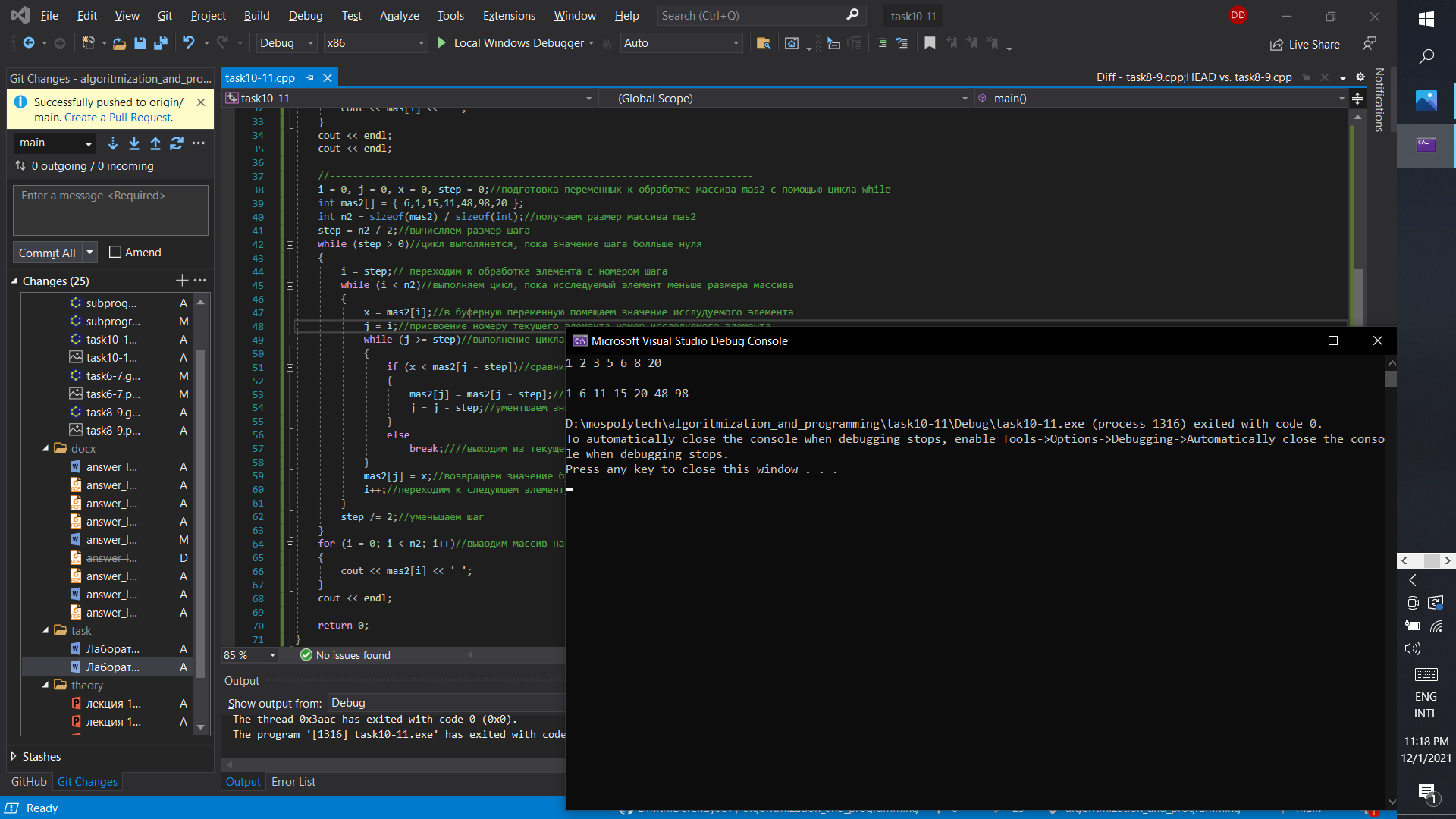
cout << mas2[i] << ' ';

}

cout << endl;

return 0;

}



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

[*https://github.com/DmitriiDerendyaev/algoritmization\_and\_programming*](https://github.com/DmitriiDerendyaev/algoritmization_and_programming)