**Лабораторная работа №1-2**

**Анализ алгоритмов и их сложности**

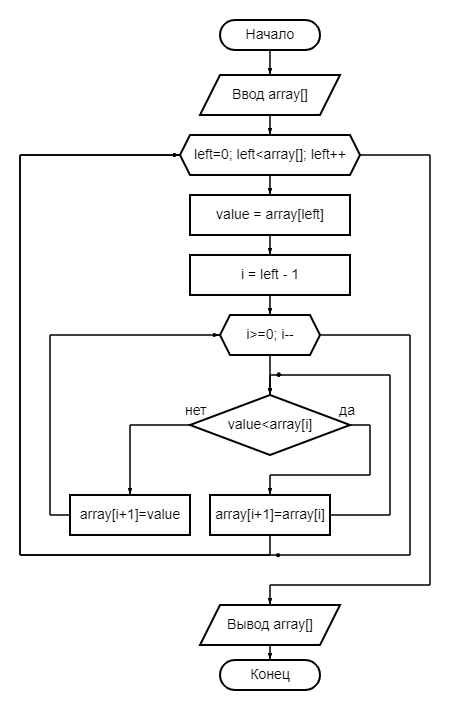
**Цель работы:** провести анализ алгоритма и оценить его сложность. Приобретение навыков исследования временной сложности алгоритмов и определения ее асимптотических оценок.

**Вариант 19:** Алгоритм сортировки вставками.

**Отчёт**

Постановка задачи: требуется провести анализ и оценку временной сложности алгоритма сортировки вставками.

Блок-схема алгоритма:



Программная реализация алгоритма на языке Java:

import java.util.Arrays;

public class InsertionSort {

public static void main(String[] args) {

int array[] = {};

for (int left = 0; left < array.length; left++){

// Вытаскиваем значение элемента

int value = array[left];

// Перемещаемся по элементам, которые перед вытащенным элементом

int i = left - 1;

for (; i >= 0; i--) {

// Если вытащили значение меньшее — передвигаем больший элемент дальше

if (value < array[i]) {

array[i + 1] = array[i];

} else {

// Если вытащенный элемент больше — останавливаемся

break;

}

}

// В освободившееся место вставляем вытащенное значение

array[i + 1] = value;

}

System.out.println(Arrays.toString(array));

}

}

Алгоритм имеет O(n2) сложность – это означает, что при увеличении данных в 10 раз количество операций и время выполнения увеличится примерно в 100 раз.

Данная сортировка наиболее эффективна, когда массив уже частично отсортирован/элементов массива не много. Он является лучшим, если элементов меньше 10.

Контрольные вопросы:

1. **Временная сложность** – сложность алгоритма, выражаемая в зависимости времени выполнения алгоритма от количества обрабатываемых входных данных, которые необходимо выполнить алгоритму для достижения нужного результата.
2. **Асимптотическая оценка временной сложности** – использование асимптотического анализа для оценки вычислительной сложности алгоритмов и вычислительных задач (скорости роста количества операций при бо́льших входных значениях).
3. **Основные принципы получения асимптотических оценок**:

* За функцию берётся количество операций, возрастающее быстрее всего.
* O() константы не учитываются.
* Основание логарифма внутри символа O() не пишется.

1. **Правила анализа алгоритмов:**

* Выбор параметра оценки, по которому идёт оценка размера входных данных.
* Определение основной операции.
* Проверка на наличие зависимости не только от размера входных данных.
* Составить соотношение, выражающее количество выполняемых операций алгоритма и его условий.
* Найти решение или определить хотя бы порядок роста.

Выводы: анализ и оценка сложности алгоритма нужны для определения затрачиваемого времени или ресурсов в зависимости от размеров входных данных.