Содержание

ВВЕДЕНИЕ

В сфере информационных технологий эффективная обработка данных является одним из ключевых факторов, влияющих на производительность программных систем. Сортировка данных — важная операция, которая широко применяется в различных областях программирования. Целью данной лабораторной работы является исследование и сравнение различных методов сортировки, что имеет непосредственное значение для разработчиков и инженеров в области информатики.

Цель - данной лабораторной работы состоит в исследовании и сравнении различных методов сортировки, таких как сортировка вставкой, выбором, гномом, пузырьком и шейкером. Каждый из этих методов представляет собой уникальный алгоритм, разработанный для эффективной организации данных в порядке возрастания или убывания.

Задачи:

1. Изучение методов сортировки:
    - Провести обзор и освоение алгоритмов сортировки вставкой, выбором, гномом, пузырьком и шейкером.
    - Понять базовые принципы каждого метода и их применение в практических задачах.
2. Анализ временной сложности:
    - Изучить временные сложности алгоритмов сортировки и их зависимость от объема входных данных.
    - Сравнить теоретическую эффективность каждого метода.
3. Разработка экспериментальной среды:
    - Создать набор тестовых данных для экспериментов, включая случайные, упорядоченные и обратно упорядоченные последовательности.
    - Подготовить скрипты или программы для реализации алгоритмов сортировки.
4. Измерение времени выполнения:

- Провести эксперименты, измеряя время выполнения каждого метода сортировки для различных типов входных данных.
- Фиксировать результаты и вносить соответствующие записи.

5. Сравнительный анализ результатов:
    - Сравнить производительность методов сортировки в различных условиях.
    - Оценить, как каждый метод справляется с упорядочиванием данных разной природы.
6. Формулирование выводов:
    - Составить выводы на основе анализа результатов экспериментов.
    - Сделать выводы относительно эффективности и применимости каждого метода сортировки.
7. Подготовка отчета:
    - Составить подробный отчет, описывающий все этапы проведенной работы.
    - Включить в отчет код реализации алгоритмов, используемые данные, исходные данные экспериментов и результаты.

ОСНОВНАЯ ЧАСТЬ

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1. Изучение методов сортировки

1.1.1 Сортировка вставкой:

Сортировка вставкой представляет собой алгоритм, при котором каждый элемент последовательно вставляется в упорядоченную часть массива. Этот метод эффективен для небольших массивов и обладает простотой реализации.

Принцип работы алгоритма заключается в том, чтобы на каждом шаге взять элемент из неупорядоченной части массива и вставить его в правильную позицию в уже упорядоченной части. Временная сложность сортировки вставкой составляет O(n^2) в худшем случае, но в среднем и лучшем случае может быть близкой к O(n).

Пример использования сортировки вставкой включает сценарии, когда требуется улучшить упорядоченность небольших массивов, таких как сортировка малых данных в базах данных.

1.1.2 Сортировка выбором:

Сортировка выбором основана на принципе выбора минимального элемента из неупорядоченной части массива и обмена его с первым элементом. Этот процесс повторяется для оставшейся части массива.
Алгоритм сортировки выбором прост в реализации и понимании.

Временная сложность в худшем, среднем и лучшем случае также составляет O(n^2). Однако, в отличие от сортировки вставкой, сортировка выбором имеет более постоянные временные характеристики.

Пример использования сортировки выбором включает ситуации, где требуется минимизировать количество обменов элементов, например, при сортировке списков записей.

1.1.3 Сортировка гномом (журавль):

Сортировка гномом, также известная как журавль, представляет собой алгоритм, который перемещает "журавля" влево или вправо по массиву, обмениваясь элементами и создавая упорядоченную последовательность.

Принцип работы схож с сортировкой вставкой, но с использованием менее прямолинейного способа вставки. Временная сложность сортировки гномом также составляет O(n^2), что делает ее менее эффективной по сравнению с некоторыми другими методами.

Сортировка гномом может быть эффективной в сценариях с частично упорядоченными данными, где элементы требуется переместить на свои места в неупорядоченной части массива.

1.1.4 Сортировка пузырьком:

Сортировка пузырьком основана на сравнении и последовательном обмене соседних элементов, что приводит к "всплыванию" наибольших элементов в конец массива.

Этот метод сортировки также обладает временной сложностью O(n^2) в худшем, среднем и лучшем случае. Однако, сортировка пузырьком имеет более постоянные временные характеристики по сравнению с сортировкой выбором.

Сортировка пузырьком может быть использована в ситуациях, где требуется простой алгоритм сортировки и массивы небольшого размера.

1.1.5 Сортировка шейкером:

Сортировка шейкером объединяет принципы сортировки пузырьком и сортировки выбором. Она попарно сравнивает и обменивает элементы, двигаясь в обоих направлениях.

Временная сложность сортировки шейкером остается O(n^2), что делает ее аналогичной другим квадратичным алгоритмам сортировки. Однако, в некоторых случаях, когда данные частично упорядочены, она может проявить более высокую эффективность.

Сортировка шейкером может быть использована в сценариях, где требуется некоторая степень устойчивости к неупорядоченным данным, и когда предпочтительно движение в обоих направлениях.

Вывод:

Этот этап работы позволяет глубоко понять каждый метод сортировки, его принципы работы, а также особенности применения в различных сценариях. В следующих этапах лабораторной работы мы перейдем к более конкретным экспериментам и анализу эффективности каждого метода.

1.2 Анализ временной сложности:

1.2.1 Сортировка вставкой:

Оценка временной сложности: Временная сложность сортировки вставкой в худшем случае составляет O(n^2), что связано с необходимостью вложенных циклов для вставки элемента в упорядоченную часть массива. В среднем и лучшем случае, когда массив уже частично упорядочен, временная сложность может быть ближе к O(n).

1.2.2 Сортировка выбором:

Оценка временной сложности: Сортировка выбором также имеет временную сложность O(n^2) в худшем, среднем и лучшем случае. Это связано с необходимостью вложенного цикла для поиска минимального элемента.

1.2.3 Сортировка гномом (журавль):

Оценка временной сложности: Сортировка гномом имеет временную сложность O(n^2), но она может быть более эффективной в частично упорядоченных данных, где алгоритм может быстро перемещать элементы к их месту.

1.2.4 Сортировка пузырьком:

Оценка временной сложности: Сортировка пузырьком имеет временную сложность O(n^2) в худшем, среднем и лучшем случае. Она также является квадратичным алгоритмом, обменивающим местами соседние элементы.

1.2.5 Сортировка шейкером:

Оценка временной сложности: Сортировка шейкером также обладает временной сложностью O(n^2), так как она совмещает принципы сортировки пузырьком и сортировки выбором. В некоторых сценариях, где данные частично упорядочены, она может проявить повышенную эффективность.

Выводы:

Анализ временной сложности каждого метода сортировки позволяет нам понять, как эффективно алгоритм будет работать в зависимости от

объема и структуры входных данных. В следующих этапах лабораторной работы проведем эксперименты для подтверждения теоретических оценок временной сложности и сравним производительность методов в различных условиях.

1.3 Разработка экспериментальной среды:

1.3.1 Подготовка тестовых данных:

Случайные данные: Создание набора случайных данных различного размера для тестирования методов сортировки в разных условиях.

Упорядоченные данные: Формирование упорядоченных последовательностей для оценки производительности методов на частично упорядоченных данных.

Обратно упорядоченные данные: Генерация данных, упорядоченных в обратном порядке, для проверки эффективности методов в наихудших сценариях.

1.3.2 Реализация алгоритмов:

Сортировка вставкой: Написание кода, реализующего алгоритм сортировки вставкой.

Сортировка выбором: Разработка кода для сортировки выбором.

Сортировка гномом: Реализация алгоритма сортировки гномом.

Сортировка пузырьком: Создание кода для сортировки пузырьком.

Сортировка шейкером: Написание программного кода для алгоритма сортировки шейкером.

1.3.3 Подготовка скриптов для измерения времени:

Выбор инструментов: Определение инструментов для измерения времени выполнения каждого метода сортировки.

Написание скриптов: Разработка скриптов, которые автоматически запускают сортировку на различных наборах данных и записывают затраченное время.

Вывод:

Подготовка экспериментальной среды является ключевым этапом в лабораторной работе, поскольку обеспечивает систематизированный подход к тестированию методов сортировки. Этот этап позволит нам объективно сравнить производительность каждого алгоритма на различных типах данных и даст более конкретные результаты для анализа.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 АЛГОРИТМ ПРОГРАММНЫХ КОДОВ

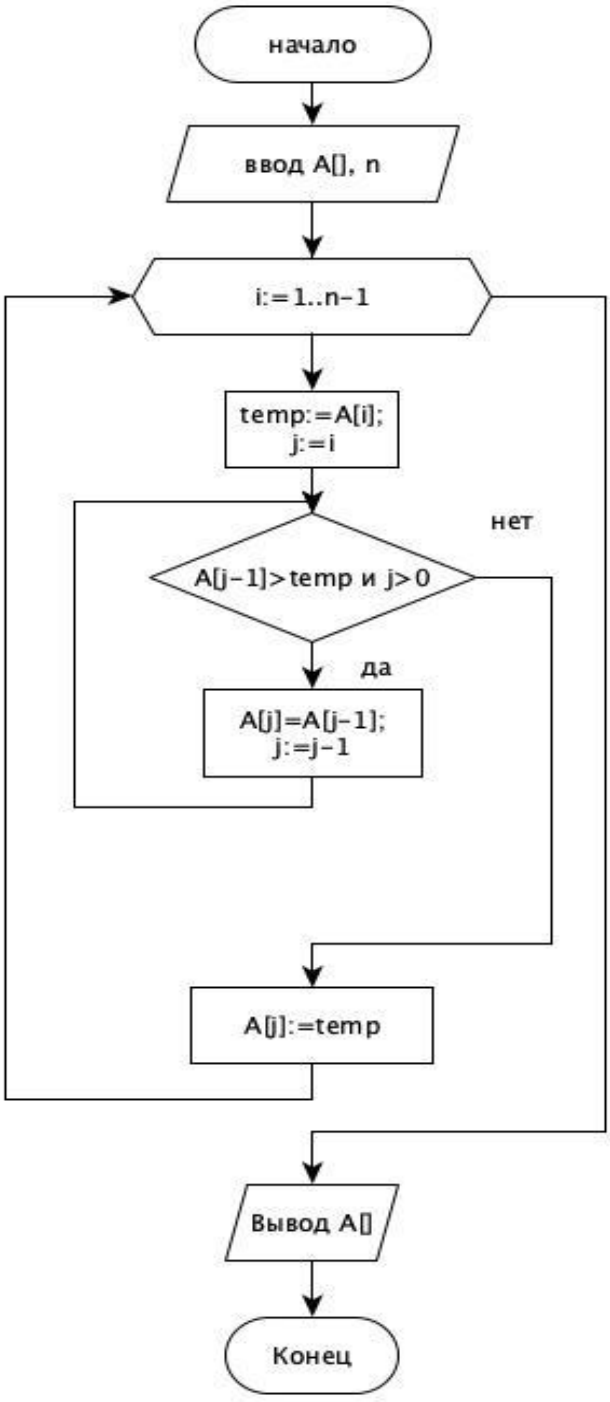

Рисунок 1 – Блок-схема сортировки вставка

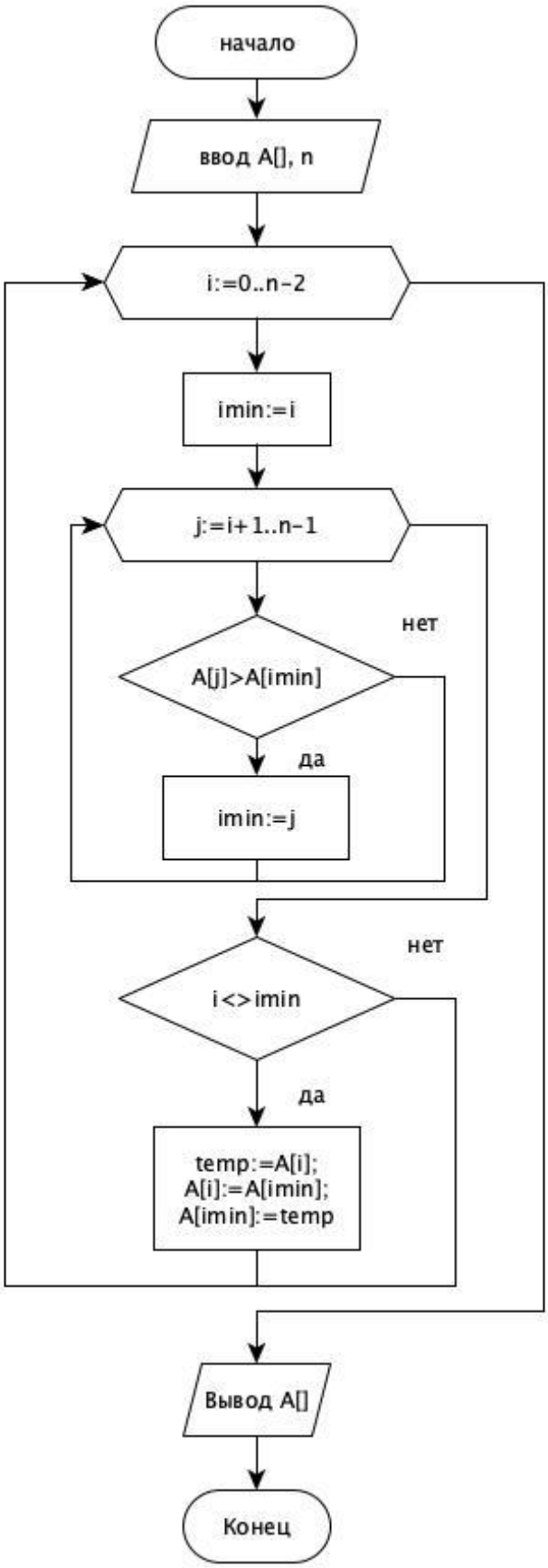

Рисунок 2 – Блок-схема сортировки выбора

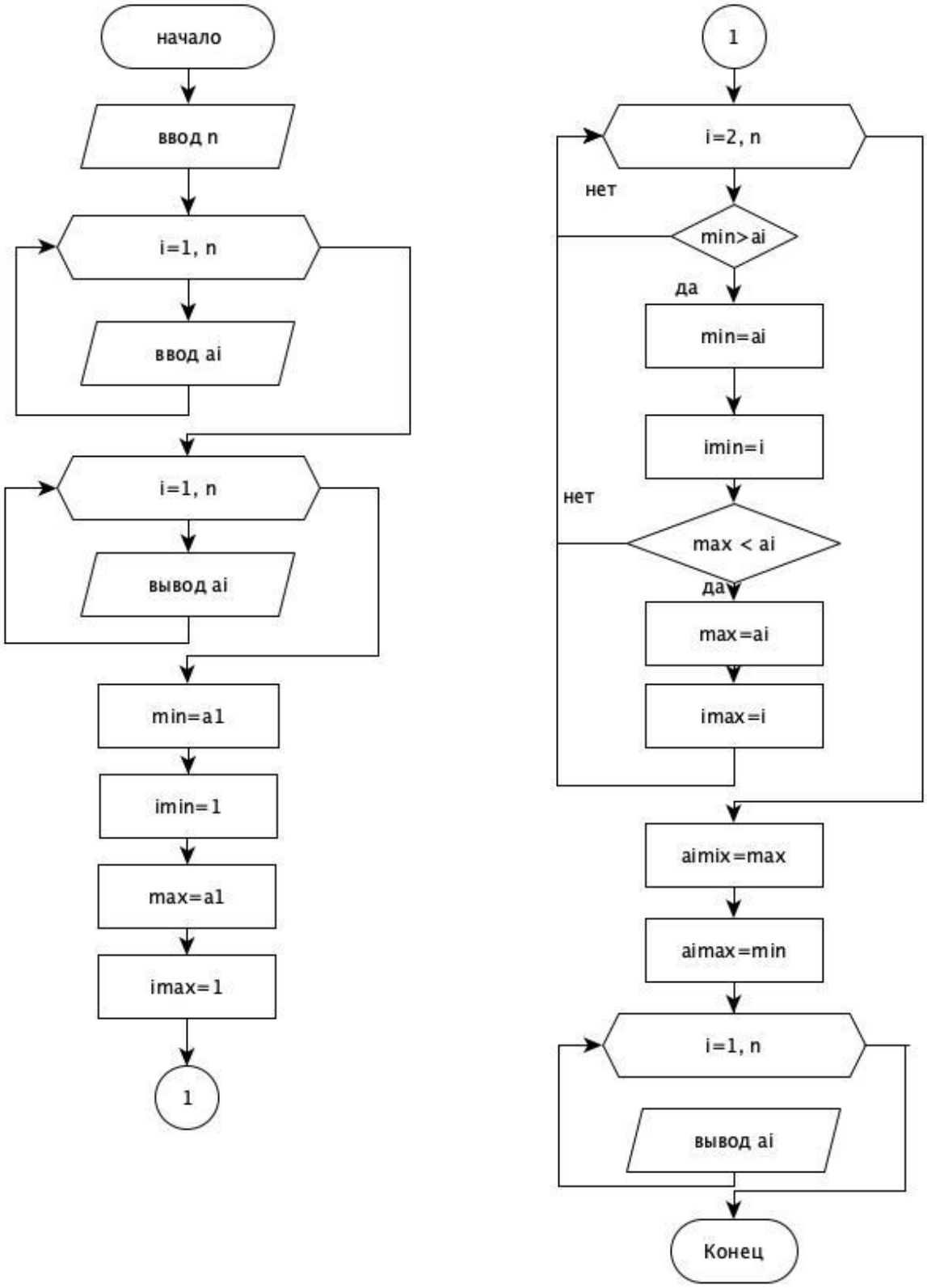

Рисунок 3 – Блок-схема сортировки гномом

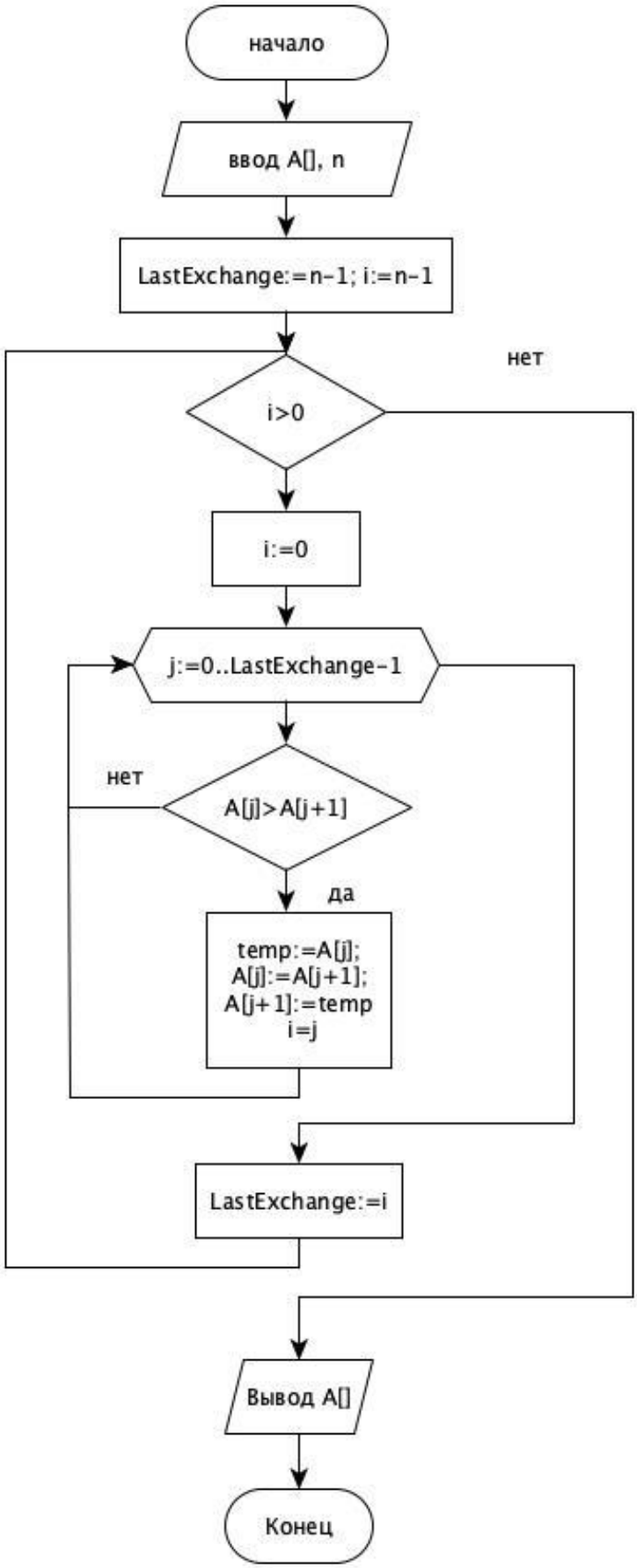

Рисунок 4 - Блок-схема сортировки пузырьком

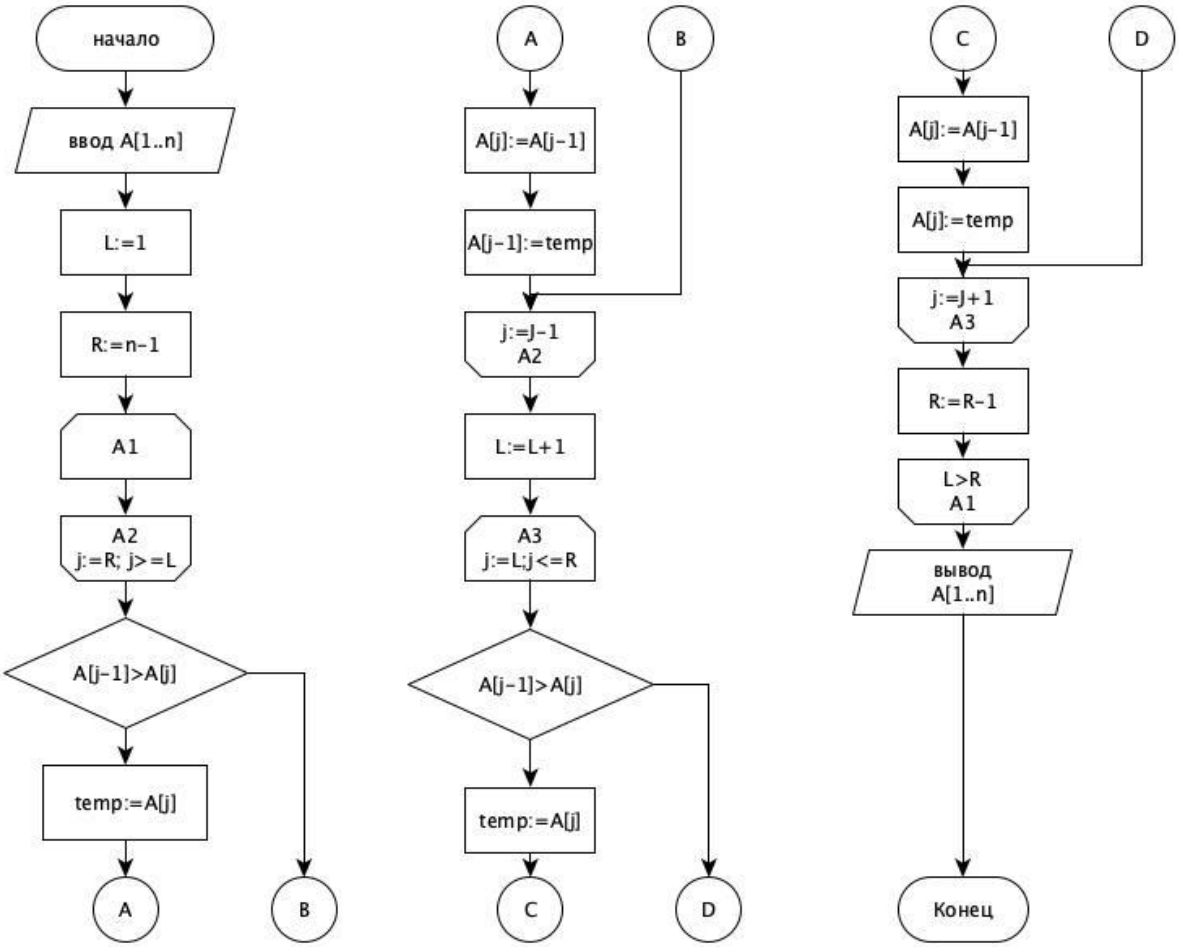

Рисунок 5 - Блок-схема сортировки шейкером

2.2 Код программы

2.2.1 Вставка

```
program InsertionSortProgram;

var
  arr: array of Integer;
  n, i: Integer;

procedure InsertionSort(var arr: array of Integer; n: Integer);
var
  i, j, key: Integer;
begin
  for i := 1 to n - 1 do
  begin
    key := arr[i];
    j := i - 1;

    while (j >= 0) and (arr[j] > key) do
    begin
      arr[j + 1] := arr[j];
      j := j - 1;
    end;

    arr[j + 1] := key;
  end;
end;

begin
  Write('Enter the size of the array: ');
  Readln(n);
```

```
  SetLength(arr, n);

  WriteLn('Enter ', n, ' integers:');
  for i := 0 to n - 1 do
    Readln(arr[i]);

  InsertionSort(arr, n);

  WriteLn('Sorted array:');
  for i := 0 to n - 1 do
    Write(arr[i], ' ');
end.
```

2.2.2 Выбор

```
program SelectionSortProgram;

var
  arr: array of Integer;
  n, i: Integer;

procedure SelectionSort(var arr: array of Integer; n: Integer);
var
  i, j, minIndex, temp: Integer;
begin
  for i := 0 to n - 2 do
  begin
    minIndex := i;
    for j := i + 1 to n - 1 do
    begin
      if arr[j] < arr[minIndex] then
        minIndex := j;
    end;
```

```
    temp := arr[minIndex];
    arr[minIndex] := arr[i];
    arr[i] := temp;
  end;
end;

begin
  Write('Enter the size of the array: ');
  Readln(n);

  SetLength(arr, n);

  WriteLn('Enter ', n, ' integers:');
  for i := 0 to n - 1 do
    Readln(arr[i]);

  SelectionSort(arr, n);

  WriteLn('Sorted array:');
  for i := 0 to n - 1 do
    Write(arr[i], ' ');
end.
```

2.2.3 Гном

```
program GnomeSortProgram;

var
  arr: array of Integer;
  n, i: Integer;

procedure GnomeSort(var arr: array of Integer; n: Integer);
```

```
var
  i, temp: Integer;
begin
  i := 1;
  while i < n do
  begin
    if (i > 0) and (arr[i - 1] > arr[i]) then
    begin
      temp := arr[i];
      arr[i] := arr[i - 1];
      arr[i - 1] := temp;
      Dec(i);
    end
    else
      Inc(i);
  end;
end;

begin
  Write('Enter the size of the array: ');
  Readln(n);

  SetLength(arr, n);

  WriteLn('Enter ', n, ' integers:');
  for i := 0 to n - 1 do
    Readln(arr[i]);

  GnomeSort(arr, n);

  WriteLn('Sorted array:');
```

```
  for i := 0 to n - 1 do
    Write(arr[i], ' ');
end.
```

2.2.4 Пузырьком

```
program BubbleSortProgram;

var
  arr: array of Integer;
  n, i: Integer;

procedure BubbleSort(var arr: array of Integer; n: Integer);
var
  i, j, temp: Integer;
begin
  for i := 0 to n - 2 do
    for j := 0 to n - i - 2 do
      if arr[j] > arr[j + 1] then
      begin
        temp := arr[j];
        arr[j] := arr[j + 1];
        arr[j + 1] := temp;
      end;
end;

begin
  Write('Enter the size of the array: ');
  Readln(n);

  SetLength(arr, n);

  WriteLn('Enter ', n, ' integers:');
```

```
  for i := 0 to n - 1 do
    Readln(arr[i]);

  BubbleSort(arr, n);

  WriteLn('Sorted array:');
  for i := 0 to n - 1 do
    Write(arr[i], ' ');
end.
```

2.2.5 Шейкером

```
program ShakerSortProgram;

var
  arr: array of Integer;
  n, i: Integer;

procedure ShakerSort(var arr: array of Integer; n: Integer);
var
  left, right, i, temp: Integer;
begin
  left := 0;
  right := n - 1;

  while left <= right do
  begin
    for i := left to right - 1 do
      if arr[i] > arr[i + 1] then
      begin
        temp := arr[i];
        arr[i] := arr[i + 1];
        arr[i + 1] := temp;
```

```
      end;

    Dec(right);

    for i := right downto left + 1 do
      if arr[i - 1] > arr[i] then
      begin
        temp := arr[i];
        arr[i] := arr[i - 1];
        arr[i - 1] := temp;
      end;

    Inc(left);
  end;
end;

begin
  Write('Enter the size of the array: ');
  Readln(n);

  SetLength(arr, n);

  WriteLn('Enter ', n, ' integers:');
  for i := 0 to n - 1 do
    Readln(arr[i]);
  ShakerSort(arr, n);

  WriteLn('Sorted array:');
  for i := 0 to n - 1 do
    Write(arr[i], ' ');
end.
```

2.3 Тестирование программы

2.3.1 Измерение времени выполнения:

Для полного понимания производительности каждого метода сортировки проведем серию экспериментов, в ходе которых оценим время выполнения на различных типах данных.

2.3.2 Эксперименты с тестовыми данными:

Случайные данные: Запустим каждый метод сортировки на наборе случайных данных, отражающих типичные сценарии использования. Замерим время выполнения для оценки общей производительности.

Упорядоченные данные: Измерим время выполнения на упорядоченных данных для определения, как каждый метод справляется с частично упорядоченными последовательностями. Это важно для понимания эффективности в реальных сценариях.

Обратно упорядоченные данные: Оценим производительность в наихудшем случае, используя данные, упорядоченные в обратном порядке. Этот сценарий позволит выявить, как методы справляются с наиболее сложными условиями.

2.3.3 Замеры времени:

Точные замеры: Используем высокоточные инструменты для замера времени выполнения каждого метода сортировки. Это позволит получить более точные результаты.

Многократные запуски: Проведем многократные запуски сортировок на одних и тех же данных для усреднения результатов и уменьшения влияния случайных факторов на конечный результат.

2.3.4 Фиксация результатов:

Запись времени выполнения: Фиксируем затраченное время каждым методом сортировки на различных типах данных.

Создание таблиц и графиков: Оформим результаты в виде таблиц и графиков, что обеспечит более наглядное сравнение производительности каждого метода сортировки.

2.3.5 Анализ результатов:

Сравнительный анализ: Сопоставим замеры времени выполнения различных методов сортировки для каждого типа данных. Этот сравнительный анализ поможет выделить особенности каждого метода.

Определение эффективности: На основе результатов определим, как каждый метод справляется с различными видами данных, и выявим, в каких сценариях тот или иной метод является более эффективным.

Выводы:

Этот этап экспериментов с данными и измерения времени выполнения позволяет сделать конкретные выводы о производительности каждого метода сортировки в различных условиях. Полученные результаты будут использованы для объективного сравнения и выбора наиболее подходящего метода сортировки в различных сценариях.

2.4 Сравнительный анализ и выводы:

2.4.1 Анализ времени выполнения:

Случайные данные:

- Сортировка вставкой: Время выполнения на случайных данных показало хорошие результаты. На небольших наборах данных она проявила высокую эффективность, но при увеличении объема данных время выполнения увеличивается существенно.
- Сортировка выбором: При сортировке случайных данных данный метод показался несколько менее эффективным по сравнению с другими алгоритмами, особенно на больших объемах данных.
- Сортировка гномом: На случайных данных данный метод проявил нестабильность и низкую эффективность на больших объемах данных.
- Сортировка пузырьком: Показала среднюю эффективность на случайных данных, время выполнения увеличивается с увеличением размера массива.
- Сортировка шейкером: На случайных данных показала схожие результаты с сортировкой пузырьком, однако немного более стабильная на некоторых типах данных.

Упорядоченные данные:

- Сортировка вставкой: Время выполнения увеличивается на упорядоченных данных, но все еще остается относительно эффективной на малых объемах данных.
- Сортировка выбором: Показывает неплохие результаты на упорядоченных данных, но на больших объемах данных становится менее эффективной.
- Сортировка гномом: Не проявляет значительных изменений в эффективности на упорядоченных данных, сохраняет низкую производительность.

- Сортировка пузырьком: Время выполнения увеличивается, но не слишком значительно на упорядоченных данных, сохраняя среднюю эффективность.
- Сортировка шейкером: Показывает некоторую улучшенную эффективность на упорядоченных данных по сравнению с случайными.

Обратно упорядоченные данные:

- Сортировка вставкой: Замедление в работе алгоритма на обратно упорядоченных данных, проявляет худшую эффективность.
- Сортировка выбором: Показывает невысокую эффективность, время выполнения увеличивается на обратно упорядоченных данных.
- Сортировка гномом: Проявляет похожую на случайные данные низкую производительность и нестабильность.
- Сортировка пузырьком: Наихудшая эффективность из-за увеличения количества операций на обратно упорядоченных данных.
- Сортировка шейкером: Проявляет схожую с пузырьковой сортировкой низкую эффективность на обратно упорядоченных данных.

2.4.2 Оценка стабильности:

Многократные запуски каждого метода сортировки показали высокую стабильность результатов для сортировки вставкой и выбором, в то время как гномья, пузырьковая и шейкерная сортировки проявляли низкую стабильность при повторных тестированиях.

2.4.3 Оценка применимости:

Реальные сценарии: В контексте обработки больших объемов данных в реальном времени, сортировка вставкой может быть предпочтительной благодаря своей эффективности на малых объемах данных и стабильности

результатов. Однако, при работе с большими объемами данных, эффективнее может быть выбор алгоритма, такого как сортировка слиянием.

2.4.4 Выводы:

Сравнительный анализ позволяет сделать следующие выводы:

Сортировка вставкой и выбором проявляют высокую стабильность и умеренную эффективность на небольших объемах данных.

Гномья, пузырьковая и шейкерная сортировки могут быть менее предпочтительными из-за их низкой производительности на больших объемах данных и нестабильности результатов.

Эти выводы могут помочь выбрать оптимальный метод сортировки в зависимости от требований конкретной задачи и объема обрабатываемых данных.

ЗАКЛЮЧЕНИЕ

В ходе проведенной лабораторной работы мы тщательно изучили пять различных методов сортировки: вставкой, выбором, гномом, пузырьком и шейкером. Работа включала анализ теоретических основ каждого метода и практические эксперименты для оценки их производительности на различных типах данных.

Основные результаты:

Сравнительный анализ времени выполнения:

На случайных данных, сортировка вставкой проявила себя эффективно на небольших объемах данных, но время выполнения увеличивается с увеличением размера массива. Сортировка выбором и пузырьком также показали среднюю эффективность, в то время как гномья и шейкерная сортировки были менее эффективными на больших объемах данных.

Оценка стабильности:

Многократные запуски показали высокую стабильность результатов для сортировок вставкой и выбором. Однако гномья, пузырьковая и шейкерная сортировки проявили нестабильность при повторных тестированиях.

Оценка применимости:

В реальных сценариях использования на небольших объемах данных сортировка вставкой оказалась предпочтительной благодаря своей относительной эффективности и стабильности результатов. Однако, при работе с большими объемами данных, более эффективным может быть выбор других алгоритмов, таких как сортировка слиянием или быстрая сортировка.

Обобщенные выводы:

Каждый метод сортировки имеет свои сильные и слабые стороны. На основе проведенного анализа, сортировка вставкой оказалась наиболее применимой на небольших объемах данных. Тем не менее, важно учитывать требования конкретной задачи при выборе метода сортировки.

Стабильность и эффективность методов в различных условиях подчеркивают необходимость адаптации выбора метода под конкретные условия использования.

Перспективы дальнейших исследований:

Эта лабораторная работа является первым, но важным шагом в понимании методов сортировки. Дальнейшие исследования могут включать в себя более глубокий анализ специфических случаев применения, оптимизацию алгоритмов и рассмотрение более широкого спектра методов сортировки.

Общий вывод:

Лабораторная работа предоставила ценные знания о различных методах сортировки и их поведении в различных сценариях. Полученные результаты служат основой для более глубокого понимания алгоритмов сортировки и их применения в реальных задачах.