Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЁТ

по лабораторной работе №2

Тема работы: Исследование алгоритмов сортировки массивов

Выполнил

студент: гр. 151003 Барановский Р.А.

Проверил: Фадеева Е.П.

Минск 2022

содержание

[1 Постановка задачи 3](#_Toc113032287)

[2 Методика решения задачи 4](#_Toc113032288)

[2.1 Подготовка массивов 4](#_Toc113032289)

[2.2 Подсчёт сравнений 4](#_Toc113032290)

[2.3 Сортировки 4](#_Toc113032291)

[3 Описание алгоритмов решения задачи 6](#_Toc113032292)

[4 Структура данных 8](#_Toc113032293)

[4.1 Структура данных программы 8](#_Toc113032294)

[4.2 Структура данных алгоритма Swap 8](#_Toc113032295)

[4.3 Структура данных алгоритма RandomArr 8](#_Toc113032296)

[4.4 Структура данных алгоритма StraightArr 8](#_Toc113032297)

[4.5 Структура данных алгоритма RevArr 9](#_Toc113032298)

[4.6 Структура данных алгоритма ShakerSortSwapsExp 9](#_Toc113032299)

[4.7 Структура данных алгоритма ShakerSortSwapsTheory 9](#_Toc113032300)

[4.8 InsertBarSortSwapsExp 10](#_Toc113032301)

[4.9 InsertBarSortSwapsTheory 10](#_Toc113032302)

[4.10 ExperimentPrint 11](#_Toc113032303)

[4.11 PrintResults 11](#_Toc113032304)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 12](#_Toc113032305)

[5.1 Схема алгоритма решения задачи 12](#_Toc113032306)

[5.2 Схема алгоритма Swap 13](#_Toc113032307)

[5.3 Схема алгоритма RandomArr 14](#_Toc113032308)

[5.4 Схема алгоритма StraightArr 15](#_Toc113032309)

[5.5 Схема алгоритма RevArr 16](#_Toc113032310)

[5.6 Схема алгоритма ShakerSortSwapsExp 17](#_Toc113032311)

[5.7 Схема алгоритма ShakerSortSwapsTheory 20](#_Toc113032312)

[5.8 Схема алгоритма InsertBarSortSwapsExp 21](#_Toc113032313)

[5.9 Схема алгоритма InsertBarSortSwapsTheory 23](#_Toc113032314)

[5.10 Схема алгоритма PrintExperiment 24](#_Toc113032315)

[5.11 Схема алгоритма PrintResults 27](#_Toc113032316)

[6 Результаты расчетов 28](#_Toc113032317)

[Приложение А 29](#_Toc113032318)

[Приложение Б 35](#_Toc113032319)

# Постановка задачи

Провести сравнительный анализ по числу сравнений следующих видов сортировки:

1. Шейкерная сортировка;
2. Метод простых вставок с барьером.

Размерности массивов соответственно: 100, 250, 500, 1000, 2000, 3000.

Типы массивов:

1. Случайный;
2. Отсортированный;
3. Перевернутый.

# Методика решения задачи

## Подготовка массивов

Для каждой размерности N генерируется массив по определённым правилам.

Сразу отсортированный масив создаётся посредством присваивания его элементам значений, равных индексам элементов:

A[I]:=I

Отсортированный в другую сторону масив создаётся ровно наоборот:

A[I]:=N-I

Случайный массив генерируется посредством присваивания его элементам случайных значений в диапазоне от 1 до верхней границы массива:

[I]:=Random(N) + 1

## Подсчёт сравнений

При перестановке элементов в массиве количество перестановок увеличивается на 1.

## Сортировки

Шейкерная сортировка является улучшенной версией пузырьковой сортировки. Ее отличие состоит в том, что в пузырьковой сортировке проход выполняется в одну сторону, после каждой очередной итерации уменьшая количество проходимых элементов на 1. В шейкерной сортировке проходы выполняются слева направо, а затем справа налево, после каждой итерации уменьшая количество проходимых элементов на 1.

Вставки с барьерным элементом являются улучшенной версией простых вставок. В отличие от простых вставок, в вставках с барьерным элементом перед сортируемым массивом помещается вставляемый на данный момент элемент, который выполняет роль “барьера”, не давая индексу j выйти за границу массива.

В шейкерной сортировке для перестановок справедливы следующие выражения:

Mmin = 0;

Mavg = 3\*(n2 - n)/12

Mmax = 3\*(n2 - n)/6

, где Mmin – минимальное число перестановок, Mavg – среднее число перестановок, Mmax – максимальное число перестановок.

В методе простых вставок с барьером справедливы следующие выражения:

Mmin = 0;

Mavg = (n2 + 9\*n - 10)/4

Mmax = (n2 + 3\*n - 4)/2

, где Mmin – минимальное число перестановок, Mavg – среднее число перестановок, Mmax – максимальное число перестановок.

# Описание алгоритмов решения задачи

Таблица – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предполагаемый тип реализации |
| 1. | Основной алгоритм | Вызывает подпрограмму PrintResults |  |  |
| 3. | Swap  (A, B) | Обменивает элементы A и B местами | A, B – получает адрес от фактического параметра | Процедура |
| 4. | RandomArr  (Arr1) | Заполняет массив Arr1 рандомными числами в диапазоне [1..N], где N – размер массива | Arr1 – получает адрес от фактического  параметра | Процедура |
| 5. | StraightArr  (Arr2) | Заполняет массив Arr2 числами от 1 до N, где N – размер массива | Arr2 – получает адрес от фактического параметра | Процедура |
| 6. | RevArr  (Arr3) | Заполняет массив Arr3 числами от N до 1, где N – размер массива | Arr3 – получает адрес от фактического параметра | Процедура |
| 7. | ShakerSortSwasExp(Arr4, Res) | Шейкерно сортирует массив Arr4 и подсчитывает количество перестановок, результат формируется в переменной Res | Arr4 – получает адрес c защитой от фактического параметра  Res – возвращаемый параметр | Функция  Res – возвращаемый параметр |
| 8. | ShakerSortSwapsTheory(N, ArrayType, Res) | Подсчитывает теоретическое количество перестановок при шейкерной сортировке массива размером N, вида ArrayType, результат формируется в переменной Res | N– получает адрес с защитой от фактического параметра  ArrayType – получает адрес с защитой от фактического параметра  Res – возвращаемый параметр | Функция  Res – возвращаемый параметр |

Продолжение таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9. | InsertBarSortswapsExp  (Arr5, Res) | Методом вставок с барьером сортирует массив Arr5 и подсчитывает количество перестановок, результат формируется в переменной Res | Arr5 – получает адрес c защитой от фактического параметра  Res – возвращаемый параметр | Функция  Res – возвращаемый параметр |
| 10. | InsertBarSortSwapsTheory(N, ArrayType, Res) | Подсчитывает теоретическое количество перестановок при сортировке методом простых вставок с барьером массива размером N, вида ArrayType, результат формируется в переменной Res | N– получает адрес с защитой от фактического параметра  ArrayType – получает адрес с защитой от фактического параметра  Res – возвращаемый параметр | Функция  Res – возвращаемый параметр |
| 11. | ExperimentPrint(N) | Распечатывает количество практических и теоретичских перестановок при сортировке случайного, сортированного, обратного массивов методами Шейкера и простых вставок с барьером, N – размер массива | N – получает адрес с защитой от фактического параметра | Процедура |
| 12. | PrintResults | Распечатывает целиком таблицу со всеми экспериментами |  | Процедура |

# Структура данных

## Структура данных программы

Таблица – Структура данных

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
|  |  |  |

## Структура данных алгоритма Swap

Таблица – Структура данных алгоритма Swap(A, B)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | Integer | Обмениваемое число | Формальный |
| B | Integer | Обмениваемое число | Формальный |
| Temp | Integer | Сохранённое число | Локальный |

## Структура данных алгоритма RandomArr

Таблица – Структура данных алгоритма RandomArr(Arr1)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr1 | Array of integer | Массив | Формальный |
| I | Integer | Параметр цикла | Локальный |

## Структура данных алгоритма StraightArr

Таблица – Структура данных алгоритма StraightArr(Arr2)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr2 | Array of integer | Массив | Формальный |
| I | Integer | Параметр цикла | Локальный |

## Структура данных алгоритма RevArr

Таблица – Структура данных алгоритма RevArr(Arr3)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr3 | Array of integer | Массив | Формальный |
| I | Integer | Параметр цикла | Локальный |

## Структура данных алгоритма ShakerSortSwapsExp

Таблица – Структура данных алгоритма ShakerSortSwapsExp(Arr4, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr4 | Array of integer | Массив | Формальный |
| I | Integer | Текущий индекс | Локальный |
| Count | Integer | Количество перестановок | Локальный |
| LeftBorder | Integer | Левая граница сортировки | Локальный |
| RightBorder | Integer | Правая граница сортировки | Локальный |
| Arr | Array of integer | Копия массива | Локальный |
| Res | Integer | Возвращаемый параметр | Формальный |

## Структура данных алгоритма ShakerSortSwapsTheory

Таблица – Структура данных алгоритма ShakerSortSwapsTheory(N, ArrayType, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Тип параметра |
| N | Integer | Количество элементов массива | Формальный |

Продолжение таблицы 8

|  |  |  |  |
| --- | --- | --- | --- |
| ArrayType | Integer | Тип массива: случайный(1)| сортированный(2)| обратный(3) | Формальный |
| Res | Integer | Возвращаемый  параметр | Формальный |

## InsertBarSortSwapsExp

Таблица – Структура данных алгоритма InsertBarSortSwapsExp(Arr5, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr5 | Array of integer | Массив | Формальный |
| I | Integer | 1 параметр цикла | Локальный |
| J | Integer | 2 параметр цикла | Локальный |
| Count | Integer | Количество  перестановок | Локальный |
| Arr | Array of integer | Копия массива | Локальный |
| Res | Integer | Возвращаемый  параметр | Формальный |

## InsertBarSortSwapsTheory

Таблица – Структура данных алгоритма InsertBarSortSwapsTheory(N, ArrayType, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Тип параметра |
| N | Integer | Количество элементов массива | Формальный |
| ArrayType | Integer | Тип массива: случайный(1)| сортированный(2)| обратный(3) | Формальный |
| Res | Integer | Возвращаемый  параметр | Формальный |

## ExperimentPrint

Таблица – Структура данных алгоритма ExperimentPrint(N)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Тип параметра |
| N | Integer | Количество элементов массива | Формальный |
| I | Integer | 1 параметр цикла | Локальный |
| J | Integer | 2 параметр цикла | Локальный |
| Arr6 | Array of integer | Массив, который будет изменять свое первоначальное состояние(случайный|сортированный|обратный) и будет сортироваться разными способами | Локальный |
| ArrResults | Array[1..3, 1..4] of integer | Массив, содержащий выводимые в таблицу значения | Локальный |

## PrintResults

Таблица – Структура данных алгоритма PrintResults

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение | Тип параметра |
|  |  |  |  |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

## Схема алгоритма решения задачи

|  |
| --- |
|  |
| Рисунок – Схема алгоритма решения задачи |

## Схема алгоритма Swap

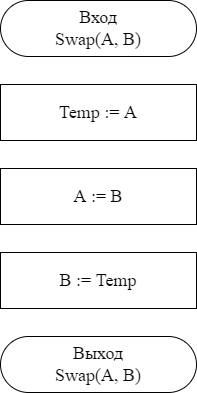


Рисунок – Схема алгоритма Swap

## Схема алгоритма RandomArr

Рисунок – Схема алгоритма RandomArr

## Схема алгоритма StraightArr

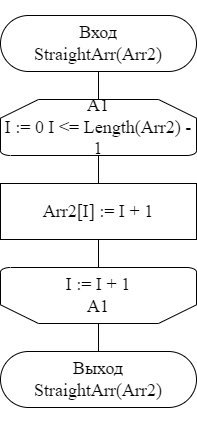


Рисунок – Схема алгоритма StraightArr

## Схема алгоритма RevArr

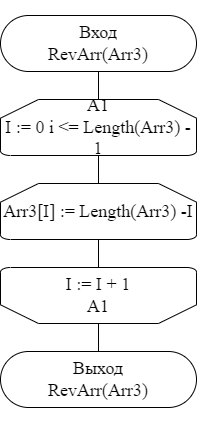


Рисунок – Схема алгоритма RevArr

## Схема алгоритма ShakerSortSwapsExp

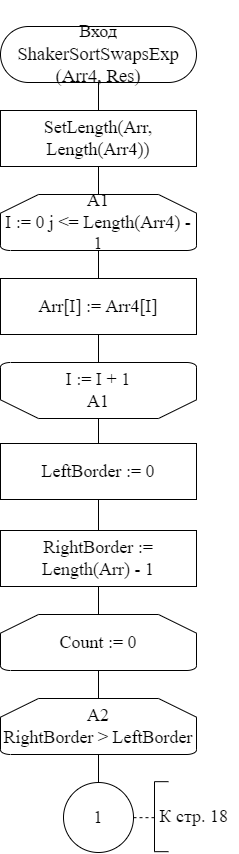


Рисунок – Схема алгоритма ShakerSortSwapsExp (часть 1)

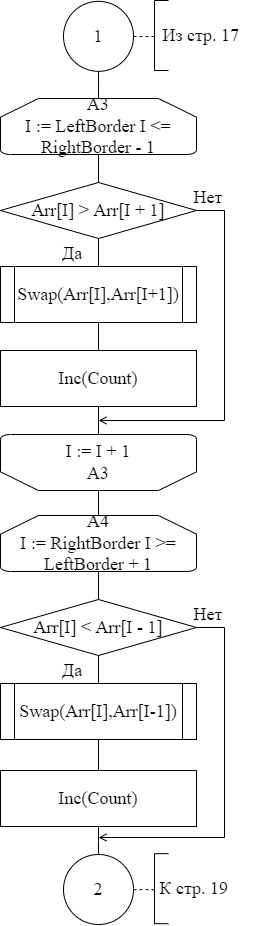


Рисунок – Схема алгоритма ShakerSortSwapsExp (часть 2)

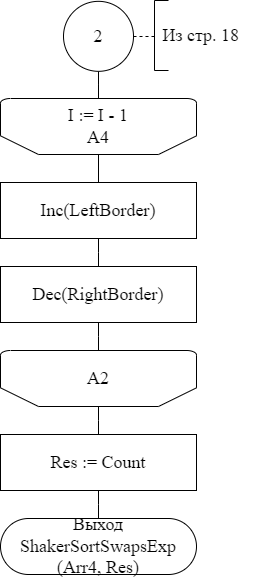


Рисунок – Схема алгоритма ShakerSortSwapsExp (часть 3)

## Схема алгоритма ShakerSortSwapsTheory

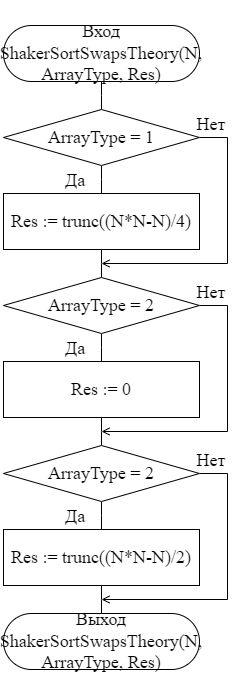


Рисунок – Схема алгоритма ShakerSortSwapsTheory

## Схема алгоритма InsertBarSortSwapsExp

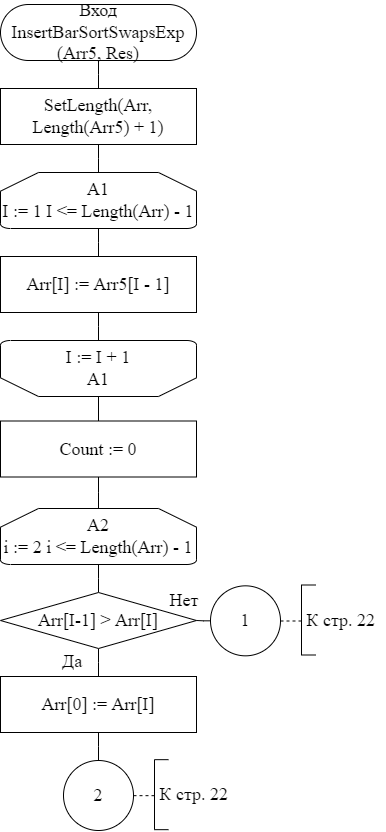


Рисунок – Схема алгоритма InsertBarSortSwapsExp (часть 1)

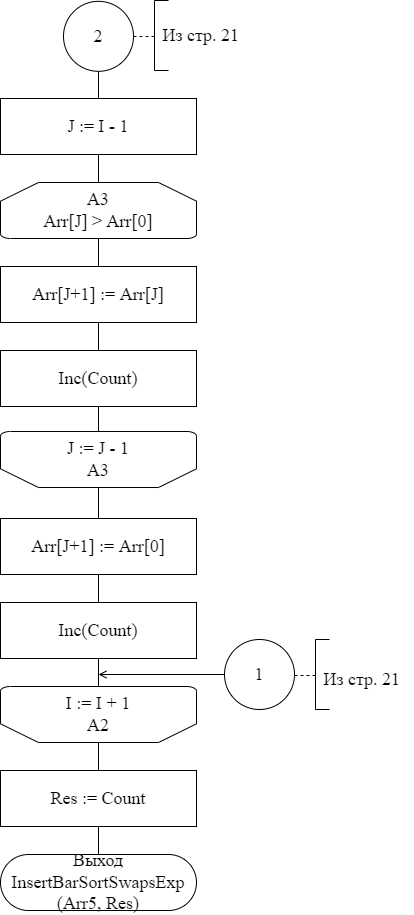


Рисунок – Схема алгоритма InsertBarSortSwapsExp (часть 2)

## Схема алгоритма InsertBarSortSwapsTheory

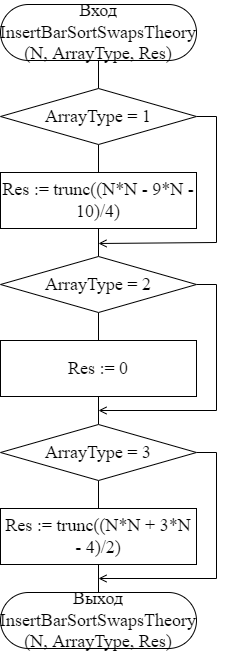


Рисунок – Схема алгоритма InsertBarSortSwapsTheory

## Схема алгоритма PrintExperiment

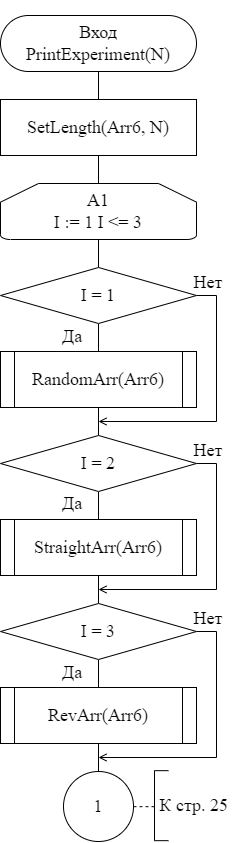


Рисунок – Схема алгоритма PrintExperiment (часть 1)

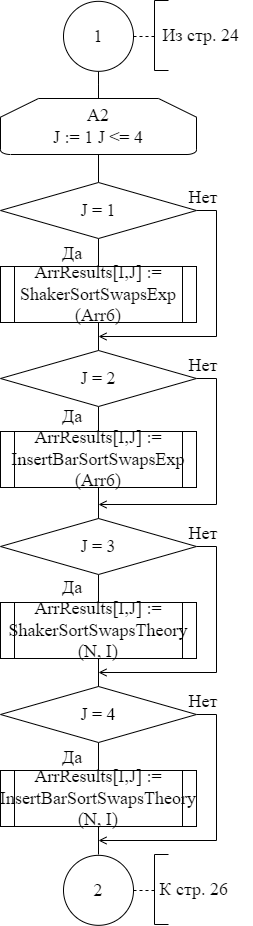


Рисунок – Схема алгоритма PrintExperiment (часть 2)

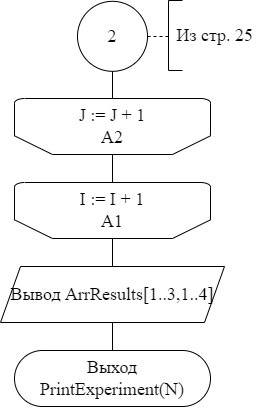


Рисунок – Схема алгоритма PrintExperiment (часть 3)

## Схема алгоритма PrintResults

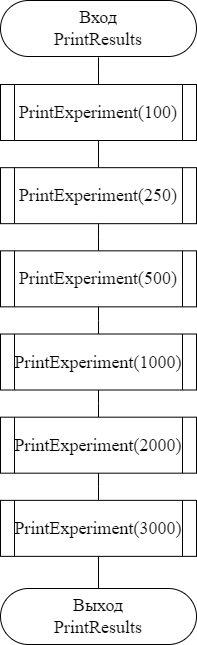
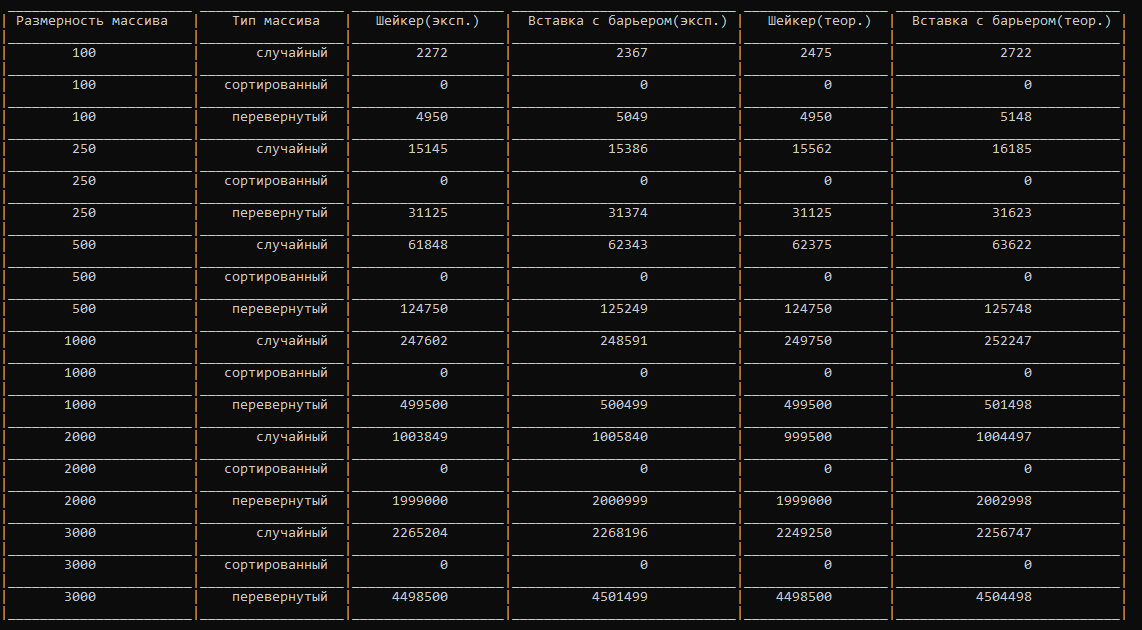


Рисунок – Схема алгоритма PrintResults

# Результаты расчетов

Вследствие результатов программы мы получаем следующие результаты:



Приложение А

(обязательное)

Исходный код программы

program Lab2;

//Program compare sorts

//Consnole app

{$APPTYPE CONSOLE}

//Modules declaration

uses

System.SysUtils;

//Array type

type

mas = array of integer;

//Fills array with random integer numbers in range [1..length(arr1)]

procedure RandomArr(var arr1: mas);

var

i: integer;

{ arr1 - array

i - counter }

begin

for i := 0 to length(arr1) - 1 do

arr1[i] := random(length(arr1)) + 1;

end;

//Fills array with integer numbers from 1 to N

//where N - number of array elements

procedure StraightArr(var arr2: mas);

var

i: integer;

{ arr2 - array

i - counter }

begin

for i := 0 to length(arr2) - 1 do

arr2[i] := i + 1;

end;

//Fills array with integer numbers from N to 1

//where N - number of array elements

procedure RevArr(var arr3: mas);

var

i: integer;

begin

for i := 0 to length(arr3) - 1 do

arr3[i] := length(arr3) - i;

end;

//Swap elements

procedure Swap(var a,b: integer);

var

temp: integer;

{ a - 1-st number

b - 2-nd number

temp - auxiliary variable}

begin

temp := a;

a := b;

b := temp;

end;

//Calculate number of swaps in shaker sort

function ShakerSortSwapsExp(const arr4: mas): integer;

var

i, count: integer;

leftBorder, rightBorder: integer;

arr: mas;

{ arr4 - array

i - counter

count - number of swaps

leftBorder - left border in sort

rightBorder - rightBorder in sort }

begin

//Copy elements of Arr4 to Arr

setLength(arr, length(arr4));

for i := 0 to length(arr4) - 1 do

arr[i] := arr4[i];

//Set borders

leftBorder := 0;

rightBorder := length(arr) - 1;

//Initialize count

count := 0;

//Shaker Sort

while rightBorder > leftBorder do

begin

//From left to right

for i := leftBorder to rightBorder - 1 do

if arr[i] > arr[i+1] then

begin

swap(arr[i], arr[i+1]);

inc(count);

end;

//From right to left

for i := rightBorder downto leftBorder + 1 do

if arr[i] < arr[i-1] then

begin

swap(arr[i], arr[i-1]);

inc(count);

end;

//Move borders

inc(leftBorder);

dec(rightBorder);

end;

//Return number of swaps

ShakerSortSwapsExp := count;

end;

//Calculate theoretical number of swaps in shaker sort

function ShakerSortSwapsTheory(const N: integer;

const arrayType: integer): integer;

begin

if arrayType = 1 then

ShakerSortSwapsTheory := trunc((N\*N - N)/4);

if arrayType = 2 then

ShakerSortSwapsTheory := 0;

if arrayType = 3 then

ShakerSortSwapsTheory := trunc((N\*N - N)/2);

end;

//Calculate number of swaps in insert barrier sort

function InsertBarSortSwapsExp(const arr5: mas): integer;

var

i, j, count: integer;

arr: mas;

{ arr5 - array

i - counter 1

j - counter 2

count - number of swaps}

begin

//Copy elements of Arr5 to Arr reserving free space for //barrier

setLength(arr, length(arr5) + 1);

for i := 1 to length(arr) - 1 do

arr[i] := arr5[i - 1];

//Initialize count

count := 0;

//Barrier sort

for i:= 2 to length(arr) - 1 do

if arr[i-1] > arr[i] then

begin

arr[0]:= arr[i];

j:= i-1;

while arr[j]>arr[0] do

begin

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

inc(count);

j:= j-1;

end;

arr[j+1]:= arr[0];

inc(count);

end;

//Return number of swaps

InsertBarSortSwapsExp := count;

end;

//Calculate theoretical number of swaps in insert barrier //sort

function InsertBarSortSwapsTheory(const N: integer;

const arrayType: integer): integer;

begin

if arrayType = 1 then

InsertBarSortSwapsTheory := trunc((N\*N + 9\*N - 10)/4);

if arrayType = 2 then

InsertBarSortSwapsTheory := 0;

if arrayType = 3 then

InsertBarSortSwapsTheory := trunc((N\*N + 3\*N - 4)/2);

end;

//Prints result of theoretical and practical swaps in both //sorts

procedure PrintExperiment(const N: integer);

var

i, j: integer;

arr6: mas;

arrResults: array[1..3, 1..4] of integer;

{ i - counter 1

j - counter 2

arr6 - experimental array to calculate swaps

arrResults - array which contains results }

begin

//Calculate values and fill array with them

setLength(arr6, N);

for i := 1 to 3 do

begin

if i = 1 then

RandomArr(arr6);

if i = 2 then

StraightArr(arr6);

if i = 3 then

RevArr(arr6);

for j := 1 to 4 do

begin

if j = 1 then

arrResults[i,j] := ShakerSortSwapsExp(arr6);

if j = 2 then

arrResults[i,j] := InsertBarSortSwapsExp(arr6);

if j = 3 then

arrResults[i,j] := ShakerSortSwapsTheory(N,i);

if j = 4 then

arrResults[i,j] := InsertBarSortSwapsTheory(N,i);

end;

end;

//Print results

for i := 1 to 3 do

begin

if i = 1 then

write('|',N:11, ' | случайный |');

if i = 2 then

write('|',N:11, ' | сортированный |');

if i = 3 then

write('|',N:11, ' | перевернутый |');

//Print values of arrResults

for j := 1 to 4 do

begin

if j = 1 then

write(arrResults[i,j]:12,' |');

if j = 2 then

write(arrResults[i,j]:17,' |');

if j = 3 then

write(arrResults[i,j]:11,' |');

if j = 4 then

write(arrResults[i,j]:17,' |');

end;

writeln;

writeln('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|');

end;

end;

//Print results for all sizes of arrays

procedure PrintResults;

begin

writeln(' \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

writeln('| Размерность массива | Тип массива |

Шейкер(эксп.) | Вставка с барьером(эксп.) |

Шейкер(теор.) | Вставка с барьером(теор.)

|');

writeln('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|');

PrintExperiment(100);

PrintExperiment(250);

PrintExperiment(500);

PrintExperiment(1000);

PrintExperiment(2000);

PrintExperiment(3000);

end;

//Start the program

begin

randomize;

//Print table

PrintResults;

//Final operations

readln;

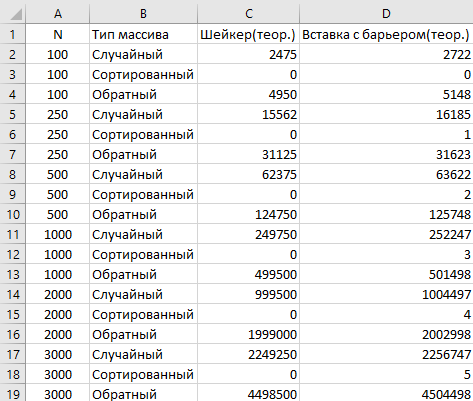
end.

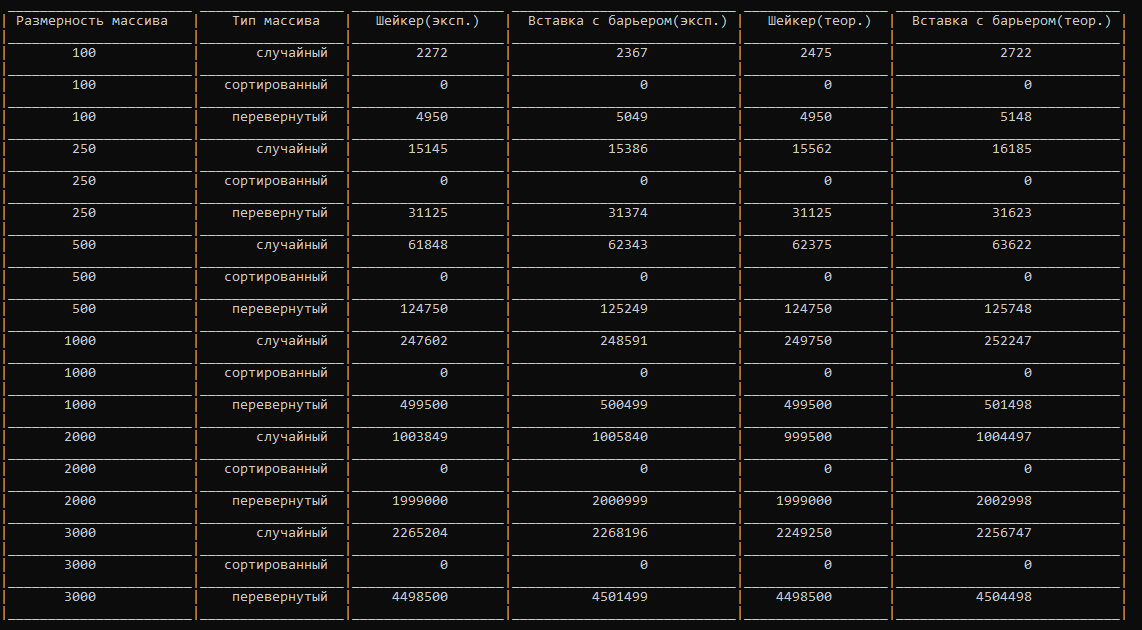
Приложение Б

(обязательное)

Тестовые наборы

Результаты на практике подтверждают теоретические результаты, рассчитанные в Microsoft Excel.





|  |  |
| --- | --- |
|  |  |