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)

ОТЧЕТ

по лабораторной работе №2

Тема работы: Исследование алгоритмов сортировки массивов

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2023

Содержание

[1 Постановка задачи 3](#_Toc135124159)

[2 Дополненная постановка задачи 4](#_Toc135124160)

[3 Метод решения 5](#_Toc135124161)

[3.1 Сортировка методом выбора 5](#_Toc135124162)

[3.2 Сортировка методом бинарных вставок 5](#_Toc135124163)

[4 Описание алгоритмов решения задачи 7](#_Toc135124164)

[5 Структура данных 9](#_Toc135124165)

[6 Схема алгоритма решения задачи по ГОСТ 19.701-90 11](#_Toc135124166)

[6.1 Схема алгоритма GetArrays 13](#_Toc135124167)

[6.2 Схема алгоритма CalcBinaryInsertionTheor 14](#_Toc135124168)

[6.3 Схема алгоритма CalcSelectionTheor 14](#_Toc135124169)

[6.4 Схема алгоритма CalcBinaryInsertionPract 15](#_Toc135124170)

[6.5 Cхема алгоритма CalcSelectionPract 17](#_Toc135124171)

[7 Результаты расчетов 19](#_Toc135124172)

[Приложение А 20](#_Toc135124173)

[Приложение Б 27](#_Toc135124174)

# Постановка задачи

Провести сравнительный анализ сортировок методом выбора и методом бинарных вставок. Критерием является количество сравнений при сортировке.

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

Типы массивов: случайный, сортированный, перевернутый.

Результаты вывести в следующем виде:

Таблица 1 – Таблица результатов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размерность  массива | Тип массива | 1-й метод сортировки | | 2-й метод сортировки | |
| Количество эксперимента  льное | Количество теоретичес  кое | Количество эксперименталь  ное | Количество теоретичес  кое |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Дополненная постановка задачи

Провести сравнительный анализ сортировок методом выбора и методом бинарных вставок. Критерием является количество сравнений при сортировке.

Размерности массива вводит пользователь. Размер должен быть не менее 3 и не более 10000.

Типы массивов: случайный, сортированный, перевернутый.

Результаты вывести в следующем виде:

Таблица 2 – Таблица результатов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размерность массива | Тип массива | 1-й метод сортировки | | 2-й метод сортировки | |
| Количество экспериментальное | Количество теоретическое | Количество экспериментальное | Количество теоретическое |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Метод решения

## Сортировка методом выбора

Рассмотрим последовательность чисел от 1 до N-1, где N - длина сортируемого массива. На первом шаге необходимо сравнить первый элемент со всеми остальными элементами в массиве, чтобы найти наименьший элемент и поместить его на первую позицию. Это означает, что первый элемент был сравнен с (N-1) оставшимися элементами. На втором шаге необходимо сравнить второй элемент с оставшимися (N-2) элементами, чтобы найти следующий наименьший элемент и поместить его на вторую позицию. И так далее, пока массив не будет полностью отсортирован.

Таким образом, общее количество сравнений в методе сортировки выбором можно рассчитать, используя сумму арифметической прогрессии:

Чтобы решить эту сумму, можно воспользоваться формулой суммы арифметической прогрессии:

Применяя эту формулу к нашей последовательности, получим:

Таким образом, общее количество сравнений в методе сортировки выбором является результатом упрощения формулы для суммы арифметической прогрессии.

## Сортировка методом бинарных вставок

Метод сортировки вставками с использованием бинарного поиска основан на следующей идее: выбираются элементы из неотсортированной части массива и вставляются в отсортированную часть массива. Таким образом, чтобы результат был отсортированным массивом.

Чтобы найти место вставки для каждого нового элемента, используем бинарный поиск. Начинается с середины отсортированной части массива и сравнивается этот элемент с новым элементом. Если новый элемент меньше текущего, продолжается поиск в левой половине отсортированной части, иначе - в правой. Делать это до тех пор, пока не найдется место для вставки нового элемента.

На каждом шаге делится отсортированная часть массива пополам, что позволяет быстро находить позицию для вставки нового элемента. Следовательно, общее количество сравнений, которое необходимо для сортировки массива методом сортировки вставками с использованием бинарного поиска, составляет .

# Описание алгоритмов решения задачи

Таблица 3 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Название алгоритма | Формальные  параметры | Предпола-гаемый тип реализации |
| 1 | Основной алгоритм | Запрашивает ввод у пользователя для получения количества элементов. Вызов следующих подпрограмм:  GetArrays, CalcSelectionPract, CalcBinaryInsertionPract, CalcSelectionTheor, CalcBinaryInsertionTheor |  |  |
| 2 | GetArrays  (  AAmOfElements,  Result  ) | Добавление AAmOfElements элементов в массив Result | AAmOfElements – получает от фактического параметра значение;  Result – получает от фактического параметра адрес, возвращаемый параметр | Функция,  Result – возвращаемый параметр |
| 3 | CalcBinaryInsertionTheor  (  AAmOfElements,  Result  ) | Подсчитывает теоретическое количество сравнений в алгоритме сортировке бинарными вставками для AAmOfElements элементов и записывает в Result | AAmOfElements – получает от фактического параметра адрес с защитой;  Result – получает от фактического параметра адрес, возвращаемый параметр | Функция,  Result – возвращаемый параметр |

Продолжение Таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | CalcSelectionTheor  (  AAmOfElements,  Result  ) | Подсчитывает теоретическое количество сравнений в алгоритме сортировке выборкой для AAmOfElements элементов и записывает в Result | AAmOfElements – получает от фактического параметра адрес с защитой;  Result – получает от фактического параметра адрес, возвращаемый параметр | Функция,  Result – возвращаемый параметр |
| 5 | CalcBinaryInsertionPract  (  AArray,  AAmOfElements,  Result  ) | Сортирует массив AArray с длинной AAmOfElements элементов методом бинарных вставок. Количество сравнений записывает в Result | AArray – получает от фактического параметра значение;  AAmOfElements – получает от фактического параметра адрес с защитой;  Result – получает от фактического параметра адрес, возвращаемый параметр | Функция,  Result – возвращаемый параметр |
| 5 | CalcSelectionPract (  AArray,  AAmOfElements,  Result  ) | Сортирует массив AArray с длинной AAmOfElements элементов методом выборки. Количество сравнений записывает в Result | AArray – получает от фактического параметра значение;  AAmOfElements – получает от фактического параметра адрес с защитой;  Result – получает от фактического параметра адрес, возвращаемый параметр | Функция,  Result – возвращаемый параметр |

# Структура данных

Таблица 4 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TArray | array [0..9999] of Integer | Cодержит массив целых числа |
| TNumOfComp | Record  BinaryInsertionPract: Integer;  SelectionPract: Integer;  end; | Cодержит количество сравнений элементов при выполнении алгоритма сортировки бинарной вставкой и сортировки выбором. |
| TRes | Record  Rand, Sorted, Reversed: TNumOfComp;  BinaryInsertionTheor, SelectionTheor : Integer;  end; | Cодержит количество сравнений элементов при выполнении теоретических и практических анализов алгоритмов сортировки |
| TArrays | Record  RandArr, SortedArr,  ReversedArr : TArray;  end; | Cодержит массивы, с рандомным, отсортированным и перевернутым порядком элементов |

Таблица 5 – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Arrays | TArrays | Массивы, которые содержат рандомным, отсортированным и перевернутым порядком элементов |
| AmOfElements | Integer | Количество элементов у массива |
| IsCorrect | Boolean | Индикатор правильного ввода |
| Сhoice | String | Индикатор продолжения программы |
| Res | TRes | Результаты вычислений |

Таблица 6 – Структура данных алгоритма GetArrays(AAmOfElements,Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| AAmOfElements | Integer | Количество элементов | Формальный |
| Result | TRes | Сформированные массивы | Формальный |
| I | Integer | Счетчик цикла | Локальный |

Таблица 7 – Структура данных алгоритма CalcBinaryInsertionTheor (AAmOfElements,Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекоменду  емый тип | Назначение | Тип параметра |
| AAmOfElements | Integer | Количество элементов | Формальный |
| Result | Integer | Полученное теоретическое количество сравнений | Формальный |

Таблица 8 – Структура данных алгоритма CalcSelectionTheor (AAmOfElements,Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекоменду  емый тип | Назначение | Тип параметра |
| AAmOfElements | Integer | Количество элементов | Формальный |
| Result | Integer | Полученное теоретическое количество сравнений | Формальный |

Таблица 9 – Структура данных алгоритма CalcBinaryInsertionPract (AArray, AAmOfElements, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекоменду  емый тип | Назначение | Тип параметра |
| AArray | TArray | Исходный массив | Формальный |
| AAmOfElements | Integer | Количество элементов | Формальный |
| Result | Integer | Количество сравнений | Формальный |
| i, j | Integer | Счетчики циклов | Локальный |
| L, R | Integer | Границы для бинарного поиска | Локальный |
| M | Integer | Средний элемент в границе L и R | Локальный |

Таблица 9 – Структура данных алгоритма CalcBinaryInsertionPract (AArray, AAmOfElements, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекоменду  емый тип | Назначение | Тип параметра |
| AArray | TArray | Исходный массив | Формальный |
| AAmOfElements | Integer | Количество элементов | Формальный |
| Result | Integer | Количество сравнений | Формальный |
| MinIndex | Integer | Индекс минимального элемента | Локальный |
| Temp | Integer | Буферная переменная | Локальный |

# Схема алгоритма решения задачи по ГОСТ 19.701-90



Рисунок 1 – Схема алгоритма в соответствии с ГОСТ 19.701-90 (часть 1)



Рисунок 2 – Схема алгоритма в соответствии с ГОСТ 19.701-90 (часть 2)

## Схема алгоритма GetArrays



Рисунок 3 – Схема алгоритма GetArrays

## Схема алгоритма CalcBinaryInsertionTheor



Рисунок 4 – Схема алгоритма CalcBinaryInsertionTheor

## Схема алгоритма CalcSelectionTheor



Рисунок 5 – Схема алгоритма CalcSelectionTheor

## Схема алгоритма CalcBinaryInsertionPract



Рисунок 6 – Схема алгоритма CalcBinaryInsertionPract (часть 1)



Рисунок 7 – Схема алгоритма CalcBinaryInsertionPract (часть 2)

## Cхема алгоритма CalcSelectionPract



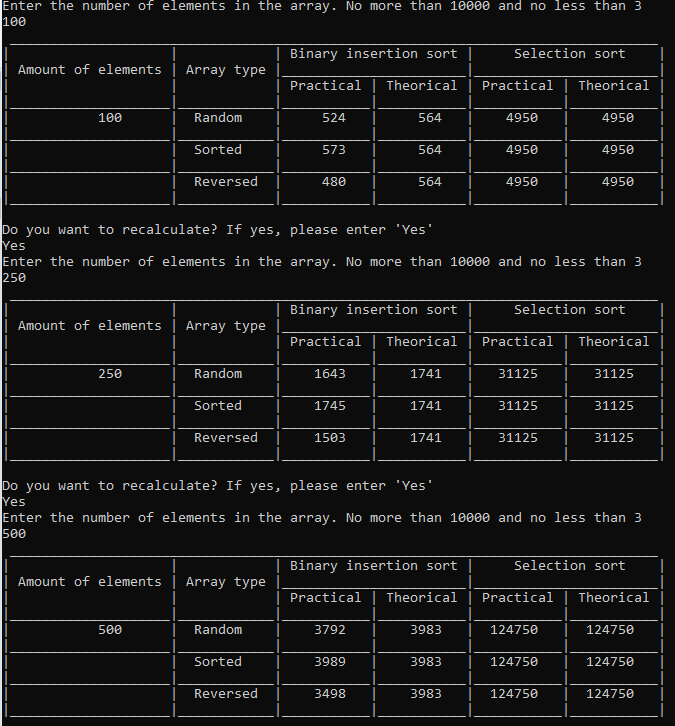
Рисунок 7 – Схема алгоритма CalcSelectionPract (часть 1)



Рисунок 8 – Схема алгоритма CalcSelectionPract (часть 2)

# Результаты расчетов

В ходе работы были получены следующие результаты:



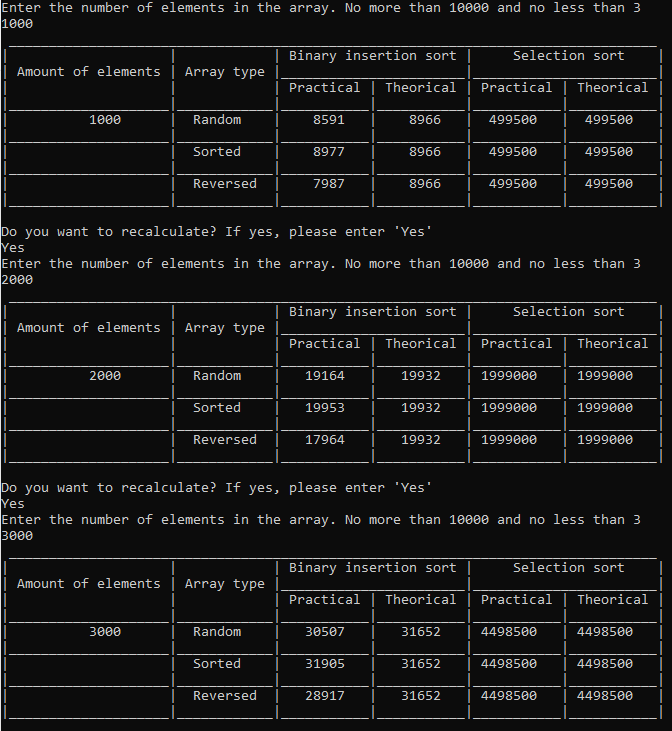


Рисунок 9 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

program Lab2;

{

This program performs an analysis of the time complexity of

binary insertion sort and selection sort algorithms. It

generates arrays of random, sorted, and reverse-ordered

elements and calculates the theoretical and practical number

of comparisons for each sorting algorithm on each array

type. The results are displayed in a table. The user can

choose to recalculate

the program.

}

{$APPTYPE CONSOLE}

uses

System.SysUtils,

uSortAnalyze in 'uSortAnalyze.pas';

const

MinAmOfElements = 3;

СontinuationStr = 'Yes';

// MinAmOfElements - minimum amount of elements in the array

// СontinuationStr - string for continue cycle

type

TNumOfComp = record

BinaryInsertionPract: Integer;

SelectionPract: Integer;

end;

TRes = record

Rand, Sorted, Reversed: TNumOfComp;

BinaryInsertionTheor, SelectionTheor : Integer;

end;

TArrays = record

RandArr, SortedArr, ReversedArr : TArray;

end;

// TNumOfComp - record with a number of comparisons

// TRes - record with results for each type of array

// TArrays - record with arrays

var

Arrays: TArrays;

AmOfElements : Integer;

IsCorrect : Boolean;

Сhoice : string;

Res: TRes;

// Arrays - record with arrays

// AmOfElements - number of elements in the array

// IsCorrect - flag for validation of input data

// Сhoice - string for continue cycle

// Res - record with results

// Fills all the required arrays up to AAmOfElements - 1 element

function GetArrays(AAmOfElements : Integer): TArrays;

var

I: Integer;

begin

Randomize;

Dec(AAmOfElements);

for I := 0 to AAmOfElements do

begin

Result.RandArr[I] := Random(AAmOfElements);

Result.SortedArr[I] := I;

Result.ReversedArr[AAmOfElements - I] := I;

end;

end;

begin

repeat

Write('Enter the number of elements in the array. No more

than ');

Writeln(High(TArray) + 1,' and no less than ',

MinAmOfElements);

//Cycle with postcondition for entering correct data.

repeat

//Initialize the flag

IsCorrect:= True;

//Validating the correct input data type

try

Readln(AmOfElements);

except

Writeln('Wrong input of amount of numbers! It must be

an integer');

IsCorrect:= False;

end;

//Validate Range

if ((AmOfElements > High(TArray) + 1) or (AmOfElements

< MinAmOfElements)) and IsCorrect then

begin

Write('Wrong input of amount of numbers! It must be

No more than ');

Writeln(High(TArray) + 1,' and no less than ',

MinAmOfElements);

IsCorrect:= False;

end;

until IsCorrect;

Arrays := GetArrays(AmOfElements);

// Calculations

Res.BinaryInsertionTheor :=

CalcBinaryInsertionTheor(AmOfElements);

Res.SelectionTheor := CalcSelectionTheor(AmOfElements);

Res.Rand.BinaryInsertionPract :=

CalcBinaryInsertionPract(Arrays.RandArr,

AmOfElements);

Res.Rand.SelectionPract :=

CalcSelectionPract(Arrays.RandArr, AmOfElements);

Res.Sorted.BinaryInsertionPract :=

CalcBinaryInsertionPract(Arrays.SortedArr,

AmOfElements);

Res.Sorted.SelectionPract :=

CalcSelectionPract(Arrays.SortedArr,

AmOfElements);

Res.Reversed.BinaryInsertionPract :=

CalcBinaryInsertionPract(Arrays.ReversedArr,

AmOfElements);

Res.Reversed.SelectionPract :=

CalcSelectionPract(Arrays.ReversedArr,

AmOfElements);

//Outputting results in table form

Writeln('

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

Writeln('| | | Binary

insertion sort | Selection sort |');

Writeln('| Amount of elements | Array type

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|');

Writeln('| | | Practical |

Theorical | Practical | Theorical |');

//Part with random

Writeln('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|

\_\_\_

\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|');

Writeln('|', AmOfElements:14, ' | Random |',

Res.Rand.BinaryInsertionPract:8, ' |',

Res.BinaryInsertionTheor:8, ' |',

Res.Rand.SelectionPract:8, ' |',

Res.SelectionTheor:8, ' |');

//Part with sorted

Writeln('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|

\_\_\_

\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|');

Writeln('| | Sorted |',

Res.Sorted.BinaryInsertionPract:8, ' |',

Res.BinaryInsertionTheor:8, ' |',

Res.Sorted.SelectionPract:8, ' |',

Res.SelectionTheor:8, ' |');

//Part with reverse

Writeln('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|');

Writeln('| | Reversed |',

Res.Reversed.BinaryInsertionPract:8, ' |',

Res.BinaryInsertionTheor:8, ' |',

Res.Reversed.SelectionPract:8, ' |',

Res.SelectionTheor:8, ' |');

Writeln('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|

\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|');

// Give the opportunity to continue the calculations

Writeln;

Writeln('Do you want to recalculate? If yes, please enter

''',СontinuationStr,'''');

Readln(Сhoice);

until Сhoice <> СontinuationStr;

end.

unit uSortAnalyze;

interface

type

TArray = array [0..9999] of Integer;

function CalcSelectionPract(AArray: TArray; const

AAmOfElements: Integer): Integer;

function CalcBinaryInsertionPract(AArray: TArray; const

AAmOfElements: Integer): Integer;

function CalcSelectionTheor(const AAmOfElements: Integer):

Integer;

function CalcBinaryInsertionTheor(const AAmOfElements:

Integer): Integer;

implementation

// This function calculates the practical complexity of

// sorting an array of

// integers using binary insertion sort. The function takes

// an array of integers

// and the number of elements in the array as input

// parameters. The function

// returns the number of comparisons performed during

// sorting.

function CalcBinaryInsertionPract(AArray: TArray; const

AAmOfElements: Integer): Integer;

var

I, J, L, R, M, Temp: Integer;

// I, J - loop counter

// L - left index of the unsorted portion of the array

// R - right index of the unsorted portion of the array

// M - middle index used for binary search

// Temp - temporary variable used for swapping elements

// during sorting

begin

// Initialize the number of comparisons to zero

Result:= 0;

// Iterate over each element in the array

for I := 1 to AAmOfElements - 1 do

begin

Temp := AArray[I];

L := 0;

R := I - 1;

// Perform binary search until the left and right indices overlap

while L <= R do

begin

Inc(Result);

M := (L + R) div 2;

if Temp < AArray[M] then

R := M - 1

else

L := M + 1;

end;

// Shift the elements to the right to make room for the current

for J := I - 1 downto L do

AArray[J + 1] := AArray[J];

AArray[L] := Temp;

end;

end;

// This function calculates the practical complexity of

// sorting an array of

// integers using selection sort. The function takes an array

// of integers

// and the number of elements in the array as input

// parameters. The function

// returns the number of comparisons performed during

// sorting.

function CalcSelectionPract(AArray: TArray; const

AAmOfElements: Integer): Integer;

var

I, J, MinIndex, Temp: Integer;

// I, J - loop counter

// MinIndex - index of the current minimum element

// Temp - temporary variable used for swapping elements

// during sorting

begin

// Initialize the number of comparisons to zero

Result := 0;

// For each element in the array except the last one

for I := 0 to AAmOfElements - 2 do

begin

MinIndex := I;

// Find the minimum element in the unsorted portion of

// the array

for J := I + 1 to AAmOfElements - 1 do

begin

Inc(Result);

if AArray[J] < AArray[MinIndex] then

MinIndex := J;

end;

// Swap the minimum element with the current element

Temp := AArray[I];

AArray[I] := AArray[MinIndex];

AArray[MinIndex] := Temp;

end;

end;

// This function calculates the theoretical number of

// comparisons needed

// for selection sort algorithm to sort an array of

// AAmOfElements elements.

// It uses the formula (n^2 - n) / 2 to calculate the number

// of comparisons

function CalcSelectionTheor(const AAmOfElements: Integer):

Integer;

begin

Result := Round((sqr(AAmOfElements) - AAmOfElements) / 2);

end;

// This function calculates the theoretical number of

// comparisons needed

// for binary insertion sort algorithm to sort an array of

// AAmOfElements elements.

// It uses the formula n\*(log2(n)) / 2 to calculate the number

// of comparisons

function CalcBinaryInsertionTheor(const AAmOfElements:

Integer): Integer;

begin

Result := Round((AamOfElements \* (ln(AAmOfElements) /

ln(2)))) - AAmOfElements;

end;

end.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

Исходные данные: Некорректный ввод

Ожидаемый результат: Повторная попытка

Полученный результат:

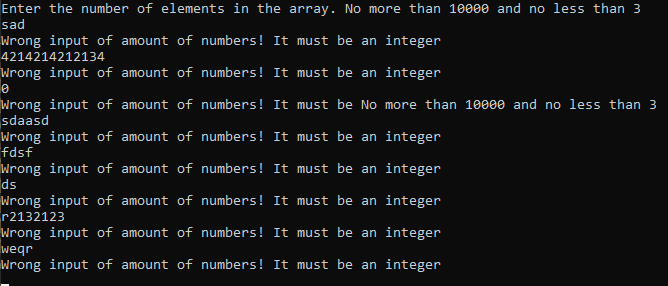


Рисунок 10 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Таблица 10 – Тестовые наборы

|  |  |
| --- | --- |
| Тест | Результат |
| 1 |  |
| 2 |  |

Продолжение таблицы 10

|  |  |
| --- | --- |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |