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 №7

Тема работы: Сортировки

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2022

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc122295026)

[2 Методика решения 4](#_Toc122295027)

[2.1 Заполнение матрицы 4](#_Toc122295028)

[2.2 Сортирвка пузырьком с быстрой границей 4](#_Toc122295029)

[2.3 Сортирвка вставками 4](#_Toc122295030)

[2.4 Краткое описание алгоритма решения задачи 4](#_Toc122295031)

[3 Текстовый алгоритм решения задачи 5](#_Toc122295032)

[4 Структура данных 7](#_Toc122295033)

[5 Схема алгоритма решения задачи по Дамке 8](#_Toc122295034)

[Приложение А 11](#_Toc122295035)

[Приложение Б 15](#_Toc122295036)

# Постановка задачи

Дана матрица ***X***[9,7]. Упорядочить элементы столбцов матрицы по неубыванию, а сами столбцы по возрастанию произведения модулей нечетных элементов столбцов.

# [Методика решения](#_Toc83996305)

## Заполнение матрицы

Заполнение исходной матрицы Matrix просходит с помощью функции Random. Чтобы данная функция генерировала числа из диапазона чисел для ShortInt, сделаем следующее: Random(256) - 128.

## Сортирвка пузырьком с быстрой границей

Предполагаем, что отсортированная часть равна длине массива. Далее обходим массив с конца до отсортированная части, сравнивая соседей. Если текущее значение меньше значения соседа, меняем их местами и отсортированной части присваиваем номер текущего элемента. Продолжаем делать данные действия до тех пор, пока отсортированная часть не станет равна длине массива.

## Сортирвка вставками

Данный алгоритм проходится, начиная со второго элемента, до конца массива и делает следующие действия. Запомнить во временную переменную значение текущего элемента массива. Пока элементы слева от запомненного значения больше чем запомненное – перемещаем их на позицию вправо. Движение элементов заканчивается, если очередной элемент, который нужно сдвинуть, оказывается по значению меньше, чем тот, что запомнили во временную переменную в начале, далее текущее значение элемента вставляем в предыдущий элемент.

## Краткое описание алгоритма решения задачи

Чтобы упорядочить столбцы по возрастанию произведения модулей нечетных элементов столбцов, посчитаем для каждого столбца произведение и запишем в двумерный массив ProdColumns[1..2, 1..7]. В первой строке будем записывать произведение столбца, а во второй – номер столбца из исходней матрицы. Далее с помощью сортировки пузырьком с быстрой границей упорядов упорядочиваем массив ProdColumns. Далее формируем от отсортированную матрицу SortedMatrix. Столбцы записываем из исходной матрицы Matrix по индексу второй строки ProdColumns. Далее элементы столбцов матрицы SortedMatrix сортируем по неубыванию с помощью сортировки вставками.

# Текстовый алгоритм решения задачи

Таблица 1 – Алгоритм решения

|  |  |
| --- | --- |
| Номер  шага | Назначение шага |
|  | Matrix[1..MaxI,1..MaxJ]:= Random(256) - 128 |
|  | Вывод Matrix[1..MaxI,1..MaxJ] |
|  | j := 1 |
|  | Начало цикла А1. Проверка выполнения условия (j <= MaxJ). Если условие истинно, перейти к шагу 5, иначе – к шагу 14 |
|  | ProdColumns[1,j]:= 1 |
|  | ProdColumns[2,j]:= j |
|  | i:= 1 |
|  | Начало цикла А1.1. Проверка выполнения условия (i <= MaxI). Если условие истинно, перейти к шагу 9, иначе – к шагу 12 |
|  | ProdColumns[1,j]:= ProdColumns[1,j] \* Abs(Matrix[i,j]) |
|  | Inc(i, 2) |
|  | Конец цикла А1.1. Вернуться к шагу 8 |
|  | j:= j + 1 |
|  | Конец цикла А1. Вернуться к шагу 4 |
|  | j:= 2 |
|  | Начало цикла А2. Проверка выполнения условия (j <= MaxJ). Если условие истинно, перейти к шагу 16, иначе – к шагу 27 |
|  | SortedPart:= MaxJ |
|  | k:= MaxJ |
|  | Начало цикла А2.1. Проверка выполнения условия (k >= j). Если условие истинно, перейти к шагу 19, иначе – к шагу 25 |
|  | Если условие (ProdColumns[1,k] < ProdColumns[1,k-1]) истинно, перейти к шагу 20. Иначе перейти к шагу 26 |
|  | Swap(ProdColumns[1,k], ProdColumns[1,k-1]) |
|  | Swap(ProdColumns[2,k], ProdColumns[2,k-1]) |
|  | SortedPart:= k |
|  | k:= k-1 |
|  | Конец цикла А2.1. Вернуться к шагу 18 |
|  | j:= SortedPart + 1 |
|  | Конец цикла А2. Вернуться к шагу 15 |
|  | j:= 1 |
|  | Начало цикла А3. Проверка выполнения условия (j <= MaxJ). Если условие истинно, перейти к шагу 29, иначе – к шагу 36 |
|  | i:= 1 |
|  | Начало цикла А3.1. Проверка выполнения условия (i <= MaxI). Если условие истинно, перейти к шагу 31, иначе – к шагу 34 |

Продолжение таблицы 1

|  |  |
| --- | --- |
|  | SortedMatrix[i,j]:= Matrix[i,ProdColumns[2,j]] |
|  | i:= i + 1 |
|  | Конец цикла А3.1. Вернуться к шагу 30 |
|  | j:= j + 1 |
|  | Конец цикла А3. Вернуться к шагу 28 |
|  | j:= 1 |
|  | Начало цикла А4. Проверка выполнения условия (j <= MaxJ). Если условие истинно, перейти к шагу 38, иначе – к шагу 51 |
|  | i:= 2 |
|  | Начало цикла А4.1. Проверка выполнения условия (i <= MaxI). Если условие истинно, перейти к шагу 40, иначе – к шагу 49 |
|  | Temp:= SortedMatrix[i,j] |
|  | k:=i-1 |
|  | Начало цикла А4.2. Проверка выполнения условий while (k >= 1) and (SortedMatrix[k,j] > Temp). Если условия истинны, перейти к шагу 43, иначе – к шагу 46 |
|  | SortedMatrix[k+1,j]:=SortedMatrix[k,j] |
|  | k:=k-1 |
|  | Конец цикла А4.2. Вернуться к шагу 42 |
|  | SortedMatrix[k+1,j]:= Temp |
|  | i:= i + 1 |
|  | Конец цикла А4.1. Вернуться к шагу 39 |
|  | j:= j + 1 |
|  | Конец цикла А4. Вернуться к шагу 37 |
|  | Вывод SortedMatrix[1..MaxI,1..MaxJ] |
|  | Останов. |

# Структура данных

Таблица 2 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| MaxI | Byte | Максимальное количество строк |
| MaxJ | Byte | Максимальное количество столбцов |
| Matrix | Array[1..MaxI, 1..MaxJ] of ShortInt | Исходная матрица |
| SortedMatrix | Array[1..MaxI, 1..MaxJ] of ShortInt | Отсортированная матрица |
| i | Byte | Счетчик цикла |
| j | Byte | Счетчик цикла |
| k | Byte | Счетчик цикла |
| SortedPart | Byte | Отсортированная часть в сортировке пузырьком |
| Temp | Integer | Буферная переменная |

# Схема алгоритма решения задачи по Дамке



Рисунок 1 – Схема алгоритма решения задачи по Дамке (часть 1)



Рисунок 2 – Схема алгоритма решения задачи по Дамке (часть 2)



Рисунок 3 – Схема алгоритма решения задачи по Дамке (часть 3)

Приложение А

(обязательное)

Исходный код программы

Program Lab7;

{

Sort the elements of the matrix columns in non-decreasing

order, and the columns

in ascending order by the products of the moduli of the odd

column elements

}

{$APPTYPE CONSOLE}

uses

System.SysUtils;

Const

MaxI = 9;

MaxJ = 7;

// MaxI - the amount of lines in the matrix

// MaxJ - the amount of columns in the matrix

Var

Matrix, SortedMatrix : Array[1..MaxI, 1..MaxJ] of ShortInt;

ProdColumns : array [1..2, 1..MaxJ] of LongWord;

i, j, k, SortedPart : Byte;

Temp: Integer;

// Martix - initial matrix

// SortedMatrix - sorted matrix

// ProdColumns - an array that, in the first line, stores

// the products of the moduli of the odd column elements.

// To the second column number corresponding to the product

// i, j, l - cycle counter

// SortedPart - sorted part in bubble sort

// Temp - Temp

Begin

Writeln('Initial matrix:');

Writeln;

// Fill the matrix with possible numbers and output it

Randomize;

for i := 1 to MaxI do

begin

for j := 1 to MaxJ do

begin

Matrix[i,j]:= Random(256) - 128;

Write(Matrix[i,j]:5,' ');

end;

Writeln;

Writeln;

end;

// Count the products of the moduli of the odd column

// elements

for j := 1 to MaxJ do

begin

// Initializing variables

ProdColumns[1,j]:= 1;

ProdColumns[2,j]:= j;

// Finding the product of odd elements

i:= 1;

while i <= MaxI do

begin

ProdColumns[1,j]:= ProdColumns[1,j] \* Abs(

Matrix[i,j]);

Inc(i, 2);

end;

Write('The product of the ',j,' column = ',

ProdColumns[1,j]);

Writeln;

Writeln;

end;

Writeln;

// Sort ProdColumns in ascending order using bubble sort

// with fast border

j:= 2;

while j <= MaxJ do

begin

// Assume that the matrix is completely sorted

SortedPart:= MaxJ;

// Checking all neighbors to the border. If find

// inconsistencies, swap them

for k:= MaxJ downto j do

if ProdColumns[1,k] < ProdColumns[1,k-1] then

begin

// Swap the neighbors from the first line using the

// Temp

Temp:= ProdColumns[1,k];

ProdColumns[1,k]:= ProdColumns[1,k-1];

ProdColumns[1,k-1]:= Temp;

// Swap the neighbors from the second line using the

// Temp

Temp:= ProdColumns[2,k];

ProdColumns[2,k]:= ProdColumns[2,k-1];

ProdColumns[2,k-1]:= Temp;

// Сhange the sorted part

SortedPart:= k;

end;

// Modernize j given the sorted part

j:= SortedPart + 1;

end;

// Form a sorted array, given the already sorted order of

// the columns

for j := 1 to MaxJ do

for i := 1 to MaxI do

SortedMatrix[i,j]:= Matrix[i,ProdColumns[2,j]];

// Sort SortedMatrix columns in non-decreasing order using

// insertion sort

for j := 1 to MaxJ do

for i:= 2 to MaxI do

begin

// Adding the current checked number to the Temp

Temp:= SortedMatrix[i,j];

// Initialize j for the cycle

k:=i-1;

// Change numbers until all elements are sorted up to

// the current number

while (k >= 1) and (SortedMatrix[k,j] > Temp) do

begin

SortedMatrix[k+1,j]:=SortedMatrix[k,j];

k:=k-1;

end;

// Assign the number reached in the cycle to the Temp

SortedMatrix[k+1,j]:= Temp;

end;

Writeln;

Writeln('Sorted matrix');

Writeln;

for i := 1 to MaxI do

begin

for j := 1 to MaxJ do

Write(SortedMatrix[i,j]:5,' ');

Writeln;

Writeln;

end;

for j := 1 to MaxJ do

begin

Write('The product of the ',j,' column = ',

ProdColumns[1,j]);

Writeln;

Writeln;

end;

Readln;

End.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

**Тест 1**

Полученный результат:

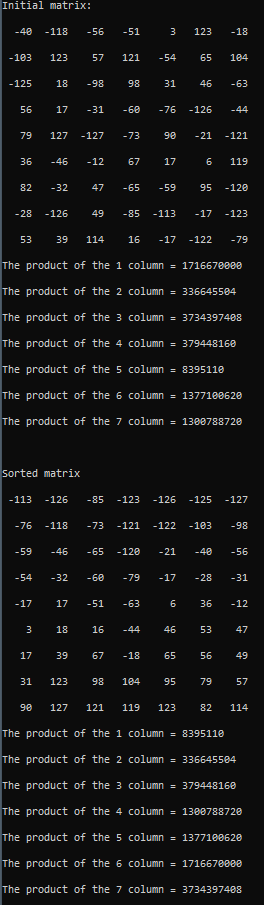


Рисунок 4 – Результаты расчетов

**Тест 2**

Полученный результат:

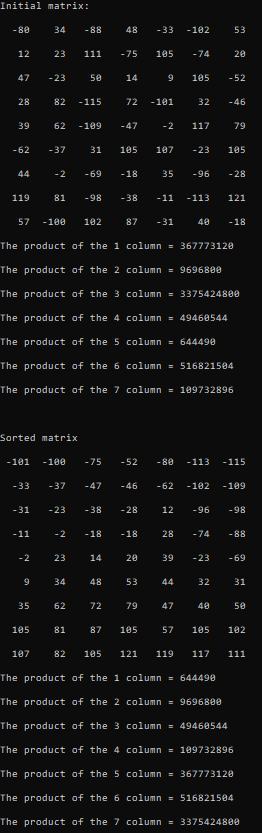


Рисунок 5 – Результаты расчетов

**Тест 3**

Полученный результат:

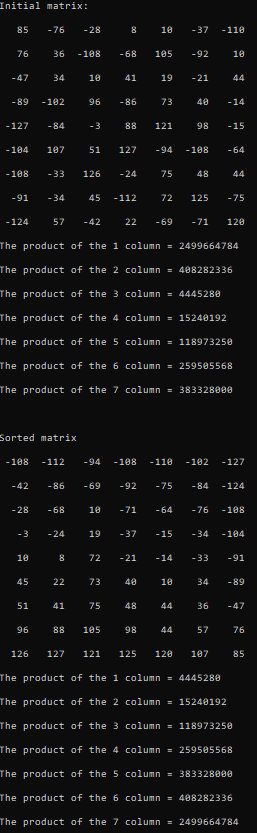


Рисунок 6 – Результаты расчетов

**Тест 4**

Полученный результат:

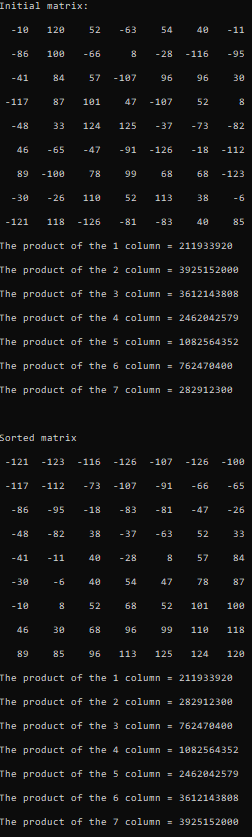


Рисунок 7 – Результаты расчетов

**Тест 5**

Полученный результат:

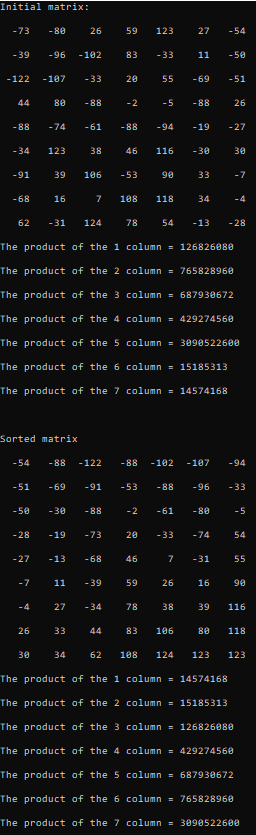


Рисунок 8 – Результаты расчетов