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: Сравнение быстрой сортировки и шейкерной сортировки по количеству перестановок.

Выполнил

студент: гр. 551003 Дементей В.С.

Проверила: Фадеева Е.П.

Минск 2016

Содержание

[1 Постановка задачи 3](#_Toc449278441)

[2 Описание алгоритмов 4](#_Toc449278442)

[4 Структура данных 6](#_Toc449278443)

[4.1 Структура данных основной программы 6](#_Toc449278444)

[4.1.1 Структура данных констант главной программы 6](#_Toc449278445)

[4.1.2 Структура данных переменных главной программы 6](#_Toc449278446)

[4.1 Структура данных подпрограмм 2](#_Toc449278447)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 4](#_Toc449278448)

[5.1 Схема основного алгоритма 4](#_Toc449278449)

[5.2 Схема алгоритма Cocktailsort (SortMass, SizeArray, transposition) 5](#_Toc449278450)

[5.3 Схема алгоритма QuickSort (SortMass, SizeArray, transposition) 6](#_Toc449278451)

[5.4 Схема алгоритма FillingAnArray (mass, SizeArray, TypeFillng) 7](#_Toc449278452)

[5.5 Схема алгоритма Theoretical\_calculation (SizeArray, TypeFilling, TypeSort, Theoretical) 8](#_Toc449278453)

[5.6 Схема алгоритма push (stack, sender) 9](#_Toc449278454)

[5.7 Схема алгоритма pop (stack, reseiver) 10](#_Toc449278455)

[6 Результаты расчетов и тестирование программы 11](#_Toc449278456)

[Приложение А 12](#_Toc449278457)

# Постановка задачи

Провести сравнительный анализ быстрой сортировки и шейкерной сортировки по числу перестановок элементов.

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

Рекомендуется использовать динамический массив.

Типы массивов: случайный, сортированный, перевернутый.

Результат вывести в таблице.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размерность массива | Быстрая сортировка | | Сортировка вставками | |
| Количество экспериментальное | Количество теоретическое | Количество экспериментальное | Количество теоретическое |
| 100 |  | |  | |
| 250 |  | |  | |
| 500 |  | |  | |
| 1000 |  | |  | |
| 2000 |  | |  | |
| 3000 |  | |  | |

# Описание алгоритмов

Таблица 1 – Описание алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование  Алгоритма | Назначение  алгоритма | Формальные  параметры | Рекомендуемый  тип |
| 1 | Основной  Алгоритм | Используя алгоритмы:  Cocktailsort  FillingAnArray  QuickSort  Theoretical\_calculation  проинициализировать массив total.  Вывести таблицу с результатом. |  |  |
| 2 | Cocktailsort  (SortMass,  SizeArray, transposition) | Алгоритм производит шейкерную сортировку массива SortMass, размерности SizeArray, сохраняет количество произведенный перестановок в  transposition | SortMass, SizeArray,  transposition  Возвращаемый параметр:  transposition. | Процедура |
| 3 | QuickSort (SortMass,  SizeArray, transposition) | Алгоритм производит быструю сортировку массива SortMass, размерности SizeArray, сохраняет количество произведенный перестановок в  Transposition  Вызывает алгоритмы:  Push,pop. | SortMass, SizeArray,  transposition  Возвращаемый параметр:  transposition. | Процедура |
| 4 | FillingAnArray  (mass,SizeArray,TypeFillng) | Алгоритм заполняет массив mass, размерности SizeArray.  В зависимости от  TypeFillng массив заполнится случайными, отсортированными или перевернутыми данными. | mass,SizeArray  TypeFillng  Возвращаемый параметр:  mass. | Процедура |
| 5 | Theoretical  \_calculation  (SizeArray,  TypeFilling,  TypeSort,  Theoretical); | Алгоритм производит теоретический расчет перестановок элементов в матрице размерностью SizeArray в зависимости от типа сортировки TypeSort и типа заполнения TypeFillng. | SizeArray, TypeFilling,  TypeSort,  Theoretical.  Возвращаемый параметр:  Theoretical. | Процедура |
| 6 | push (stack,sender) | Помещает данные sender в стек stack | stack,sender.  Возвращаемый параметр:  Stack. | Процедура |
| 7 | pop  (stack, Receiver) | Помещает данные с верхушки стека stack в Receiver | Receiver,stack  Возвращаемый параметр:  stack,receiver. | Процедура |

# Структура данных

## 4.1 Структура данных основной программы

## 4.1.1 Структура данных констант главной программы

Таблица 2 – Описание констант главной программы

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| ACSize | Array of integer | Массив с размерностями массивов |
| ACTypeStr | Array of string | Массив названий типов данных |

## 4.1.2 Структура данных переменных главной программы

Таблица 3 – Описание переменных главной программы

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый  тип | Назначение |
| MassForTest | Array of Integer | Массив данных |
| total | array[1..18, 1..4] of Integer | Массив с результатом |
| column | Integer | Счетчик |
| str | Integer | Счетчик |
| SizeArray | Integer | Счетчик |
| TypeArray | Integer | Счетчик |

## Структура данных подпрограмм

Таблица 4 – Описание переменных подпрограммы Cocktailsort

(SortMass, SizeArray, transposition)

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| SortMass | Array of Integer | Матрица с данными |
| SizeArray | Integer | Размерность матрицы |
| transposition | Integer | Количество перестановок |
| low | Integer | Левая граница сортировки |
| hight | Integer | Правая граница сортировки |
| i | Integer | Счётчик |
| TempValue | Integer | Временные данные |

Таблица 5 – Описание переменных подпрограммы QuickSort

(SortMass, SizeArray, transposition)

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| SortMass | Array of Integer | Матрица с данными |
| SizeArray | Integer | Размерность матрицы |
| transposition | Integer | Количество перестановок |
| LeftBorder | Integer | Левая граница сортировки |
| RightBorder | Integer | Правая граница сортировки |
| mid | Integer | Значение центрального элемента |
| Stack | record  data:integer;  next:TStak;  end; | Стек для хранения данных |
| TempValue | Integer | Временные данные |

Таблица 6 – Описание переменных подпрограммы FillingAnArray (MatA,MatB)

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| mass | Array of Integer | Матрица с данными |
| SizeArray | Integer | Размерность матрицы |
| TypeFillng | Integer | Тип заполнения |
| i | Integer | Счётчик |
| j | Integer | Счётчик |

Таблица 7 – Описание переменных подпрограммы Theoretical\_calculation (SizeArray, TypeFilling, TypeSort, Theoretical)

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| SizeArray | Integer | Размерность матрицы |
| TypeFillng | Integer | Тип заполнения |
| TypeSort | Integer | Тип сортировки |
| Theoretical | Integer | Количество теоретическое перестановок |

Таблица 8 – Описание переменных подпрограммы push (stack, sender)

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| TempValue | record  data:integer;  next:TStak;  end; | Размерность матрицы |
| sender | Integer | Источник данных |
| stack | record  data:integer;  next:TStak;  end; | Тип сортировки |

Таблица 9– Описание переменных подпрограммы pop (stack, sender)

|  |  |  |
| --- | --- | --- |
| Элементы  данных | Рекомендуемый тип | Назначение |
| TempValue | record  data:integer;  next:TStak;  end; | Размерность матрицы |
| Receiver | Integer | Приемник данных |
| stack | record  data:integer;  next:TStak;  end; | Тип сортировки |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

## Схема основного алгоритма

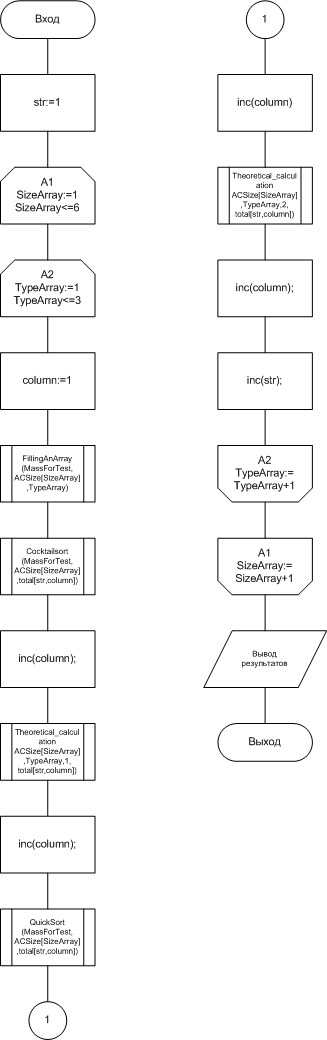


Рисунок 1 – Схема основного алгоритма

## Схема алгоритма Cocktailsort (SortMass, SizeArray, transposition)

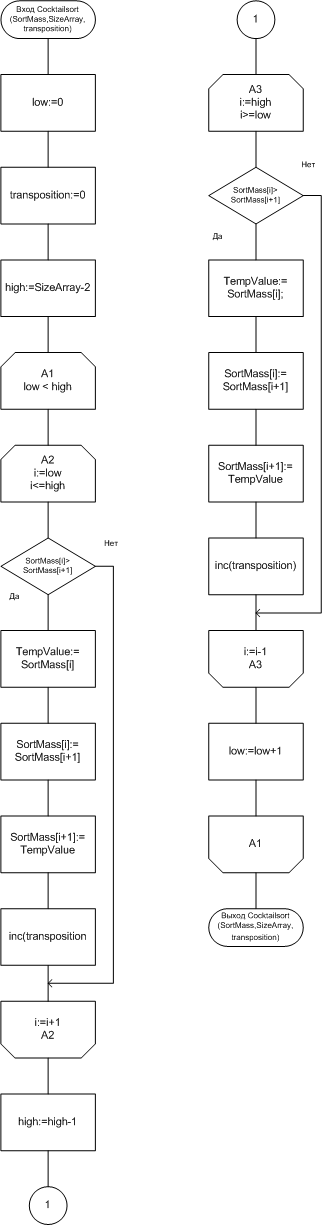
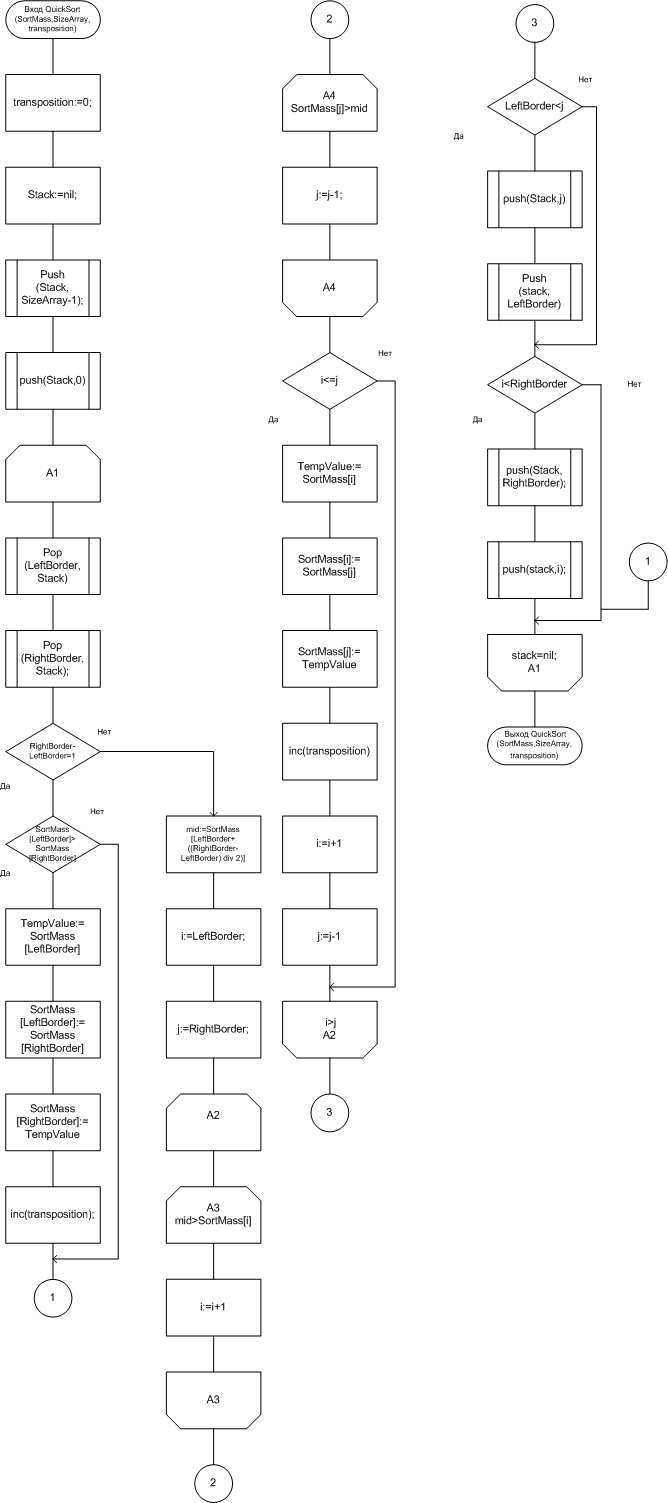


Рисунок 2 - Схема алгоритма Cocktailsort

## Схема алгоритма QuickSort (SortMass, SizeArray, transposition)

Рисунок 3 - Схема алгоритма QuickSort

## Схема алгоритма FillingAnArray (mass, SizeArray, TypeFillng)

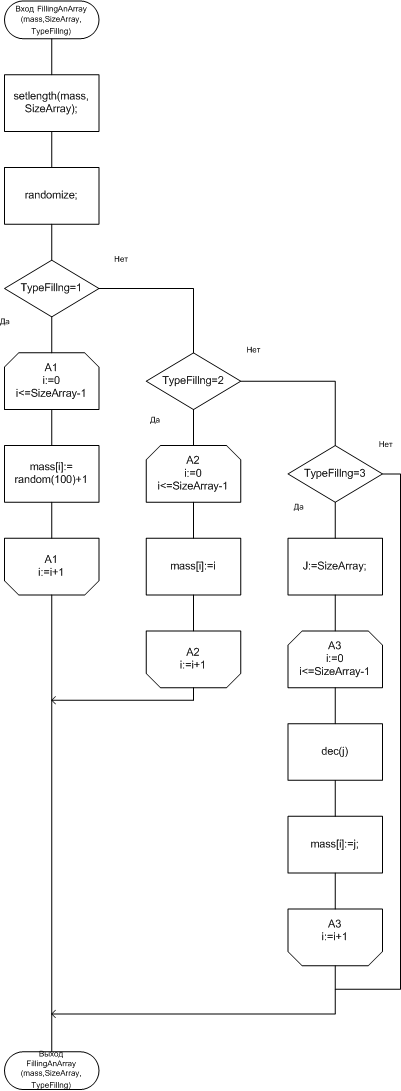


Рисунок 4 - Схема алгоритма FillingAnArray

## Схема алгоритма Theoretical\_calculation (SizeArray, TypeFilling, TypeSort, Theoretical)

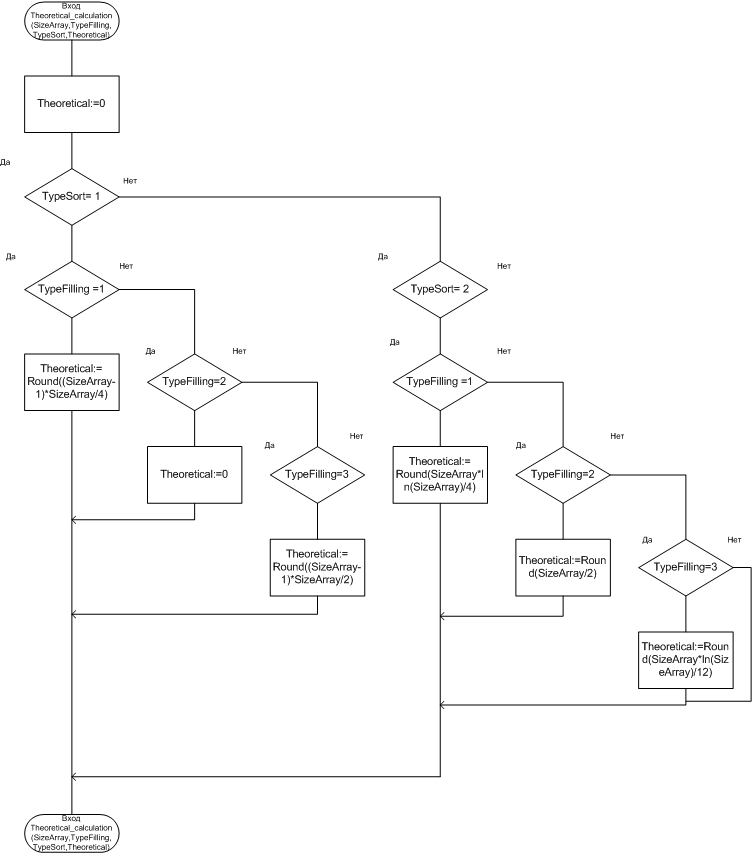


Рисунок 5 - Схема алгоритма Theoretical\_calculation

## Схема алгоритма push (stack, sender)

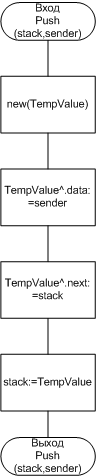


Рисунок 6 - Схема алгоритма push

## Схема алгоритма pop (stack, reseiver)

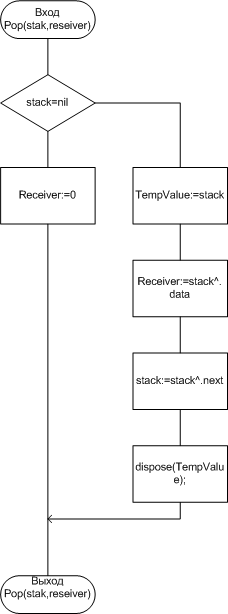
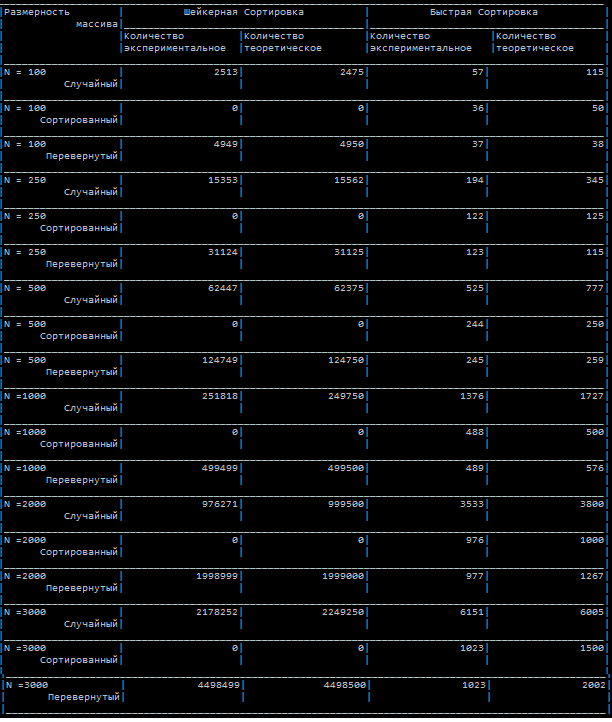


Рисунок 7 - Схема алгоритма pop

# Результаты расчетов и тестирование программы



Приложение А

(обязательное)

Исходный код программы

**Program** SortEnd;

{$APPTYPE CONSOLE}

**Uses**

SysUtils,math,windows;

**Const**

ACSize:**array**[1..6] **of** integer =

(100,250,500,1000,2000,3000);

ACTypeStr:**array**[1..3] **of** string =

('Случайный','Сортированный','Перевернутый');

**Type**

TArray=**array of** integer;

TArrayRes=**array**[1..18,1..4] **of** integer;

**Var**

MassForTest:Tarray;

i:integer;

total:TArrayRes;

column:integer;

str:integer;

SizeArray:integer;

TypeArray:integer;

**Procedure** Cocktailsort(SortMass:Tarray;SizeArray:integer;

**var** transposition:integer);

**Var**

low,high:integer;

TempValue:integer;

i:integer;

**begin**

//устанавливаем начало и конец массива

low:=0;

transposition:=0;

high:=SizeArray-2;

**while** low < high **do**

**begin**

// проход массива от начала к концу

**for** i:=low **to** high **do**

**begin**

**if** SortMass[i]>SortMass[i+1] **then**

**begin**

TempValue:=SortMass[i];

SortMass[i]:=SortMass[i+1];

SortMass[i+1]:=TempValue;

inc(transposition);

**end**;

**end**;

// отсекли отсортированный элемент

high:=high-1;

// проход массива от конца к началу

**for** i:= high **downto** low **do**

**begin**

**if** SortMass[i]>SortMass[i+1] **then**

**begin**

TempValue:=SortMass[i];

SortMass[i]:=SortMass[i+1];

SortMass[i+1]:=TempValue;

inc(transposition);

**end**;

**end**;

// отсекли отсортированный элемент

low:=low+1;

**end**;

**end**;

**procedure** QuickSort(SortMass:TArray; SizeArray:integer;

**var** transposition:integer);

**Type**

TStak=^TRStak;

TRStak = **record**

data:integer;

next:TStak;

**end**;

**var**

LeftBorder,RightBorder,i,j:integer;

Stack:TStak;

mid:integer;

TempValue:integer;

//Помещение в стек

**procedure** push(**var** stack:TStak; sender:integer);

**Var**

TempValue:TStak;

**begin**

**new**(TempValue);

TempValue^.data:=sender;

TempValue^.next:=stack;

stack:=TempValue;

**end**;

//Извлечение из стека

**procedure** pop (**Var** Receiver:integer;**var** stack:TStak);

**Var**

TempValue:TStak;

**begin**

**if** stack=nil **then**

Receiver:=0

**else**

**begin**

TempValue:=stack;

Receiver:=stack^.data;

stack:=stack^.next;

dispose(TempValue);

**end**;

**end**;

**begin**

//Основной алгоритм

transposition:=0;

Stack:=nil;

push(Stack,SizeArray-1);

push(Stack,0);

**repeat**

pop(LeftBorder,Stack);

pop(RightBorder,Stack);

**if** RightBorder-LeftBorder=1 **then**

**begin**

**if** SortMass[LeftBorder]>SortMass[RightBorder] **then**

**begin**

TempValue:=SortMass[LeftBorder];

SortMass[LeftBorder]:=SortMass[RightBorder];

SortMass[RightBorder]:=TempValue;

inc(transposition);

**end**;

**end**

**else**

**begin**

mid:=SortMass[LeftBorder+((RightBorder-LeftBorder) **div** 2)];

i:=LeftBorder;

j:=RightBorder;

**repeat**

**while** (mid>SortMass[i]) **do**

i:=i+1;

**while** (SortMass[j]>mid) **do**

j:=j-1;

**if** i<=j **then**

**begin**

TempValue:=SortMass[i];

SortMass[i]:=SortMass[j];

SortMass[j]:=TempValue;

inc(transposition);

i:=i+1;

j:=j-1;

**end**;

**until** i>j;

**if** LeftBorder<j **then**

**begin**

push(Stack,j);

push(stack,LeftBorder);

**end**;

**if** i<RightBorder **then**

**begin**

push(Stack,RightBorder);

push(stack,i);

**end**;

**end**;

**until** stack=nil;

**end**;

**procedure** FillingAnArray( **var** mass:TArray;SizeArray:integer;TypeFillng:integer);

**Var**

i,j:integer;

**begin**

setlength(mass,SizeArray);

randomize;

//Первый тип -рандом

**if** TypeFillng=1 **then**

**for** i:=0 **to** SizeArray-1 **do**

mass[i]:=random(100)+1

**else**

//второй тип - сортированный

**if** TypeFillng=2 **then**

**for** i:=0 **to** SizeArray-1 **do**

mass[i]:=i

**else**

**if** TypeFillng=3 **then**

**begin**

J:=SizeArray;

**for** i:=0 **to** SizeArray-1 **do**

**begin**

dec(j);

mass[i]:=j;

**end**;

**end**;

**end**;

**procedure** Theoretical\_calculation(SizeArray:integer;TypeFilling:integer;

TypeSort:integer;**Var** Theoretical:integer );

**begin**

Theoretical:=0;

//Первая сортировка - шейкерная

**if** TypeSort= 1 **then**

**begin**

//Первый тип -рандом

**if** TypeFilling =1 **then**

Theoretical:=Round((SizeArray-1)\*SizeArray/4)

**else**

//второй тип - сортированный

**if** TypeFilling=2 **then**

Theoretical:=0

**else**

**if** TypeFilling=3 **then**

Theoretical:=Round((SizeArray-1)\*SizeArray/2);

**end**

**else**

//быстраая

**if** TypeSort= 2 **then**

**begin**

//Первый тип -рандом

**if** TypeFilling =1 **then**

Theoretical:=Round(SizeArray\*ln(SizeArray)/4)

**else**

//второй тип - сортированный

**if** TypeFilling=2 **then**

Theoretical:=Round(SizeArray/2)

**else**

**if** TypeFilling=3 **then**

Theoretical:=Round(SizeArray\*ln(SizeArray)/12);

**end**

**end**;

**begin**

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

str:=1;

**for** SizeArray:=1 **to** 6 **do**

**begin**

**for** TypeArray:=1 **to** 3 **do**

**begin**

column:=1;

FillingAnArray(MassForTest,ACSize[SizeArray],TypeArray);

Cocktailsort(MassForTest,ACSize[SizeArray],total[str,column]);

inc(column);

Theoretical\_calculation(ACSize[SizeArray],TypeArray,1,total[str,column]);

inc(column);

QuickSort(MassForTest,ACSize[SizeArray],total[str,column]);

inc(column);

Theoretical\_calculation(ACSize[SizeArray],TypeArray,2,total[str,column]);

inc(column);

inc(str);

**end**;

**end**;

write('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

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

write('|Размерность | Шейкерная Сортировка |');

writeln(' Быстрая Сортировка |');

write('| массива|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|');

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

write('| |Количество |Количество |');

writeln('Количество |Количество |');

write('| |экспериментальное |теоретическое |');

writeln('экспериментальное |теоретическое |');

write('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

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

//ячейки

str:=1;

**For** SizeArray:=1 **to** 6 **do**

**begin**

**for** typeArray:=1 **to** 3 **do**

**begin**

write('|N =',ACSize[SizeArray]:4,' |');

**for** i:=1 **to** 4 **do**

**begin**

**if** i = 2 **then**

write(total[str,i]:20,'|')

**else**

write(total[str,i]:19,'|');

**end**;

writeln;

write('|',ACTypeStr[typeArray]:19,'|');

**for** i:=1 **to** 4 **do**

**begin**

**if** i = 2 **then**

write(' |')

**else**

write(' |');

**end**;

writeln;

write('|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');

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

inc(str);

**end**;

**end**;

readln;

**end**.