Содержание:

Введение

Актуальность темы. Самой распространенной структурой, реализованной практически во всех языках программирования, является массив. Массивы состоят из ограниченного числа компонент, причем все компоненты массива имеют один и тот же тип, называемый базовым. Структура массива всегда однородна. Массив может состоять из элементов типа integer, real или char, либо других однотипных элементов. Из этого, правда, не следует делать вывод, что компоненты массива могут иметь только скалярный тип. Другая особенность массива состоит в том, что к любой его компоненте можно обращаться произвольным образом. Программа может сразу получить нужный ей элемент по его порядковому номеру (индексу).

Цель курсовой работы заключается в рассмотрении методов сортировки одномерных массивов.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить понятие массива;

- рассмотреть виды массивов;

- представить преимущество использования массивов;

- исследовать типовые алгоритмы обработки одномерных массивов и задачи, решаемые с их помощью;

- рассмотреть задачи с использованием типовых алгоритмов обработки одномерных массивов;

- разработать алгоритм решения задачи, посредством программы;

- представить руководство программиста и оператора.

Объект работы – одномерный массив.

Предмет работы – методы сортировки одномерных массивов.

Степень разработанности проблемы. В процессе исследования были проанализированы работы, посвященные понятию, видам массивов. Весомый вклад в исследование данных вопросов внесли отечественные ученые: Абрамов С.А., Бекишев Г.А., Ван Тассел Д., Вирт Н., Голицына О.Л., Готье Р., Гребенников Л.К., Дж., Вандер Плас, и др.

Методы исследования. Обоснование положений, выводов и рекомендаций, содержащихся в курсовой работе, осуществлено путем комплексного применения следующих методов: теоретический анализ, моделирование.

Курсовая работы состоит из введения, три главы, включающих в себя семь параграфов, заключения, списка использованных источников и приложений.

1 Теоретические аспекты массива

1.1 Понятие массива

Массив (англ. array) (в некоторых языках программирования также таблица, ряд, матрица) — структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу или набору индексов, принимающих целые (или приводимые к целым) значения из некоторого заданного непрерывного диапазона. Одномерный массив можно рассматривать как реализацию абстрактного типа данных вектор.

Размерность массива — это количество индексов, необходимое для однозначной адресации элемента в рамках массива. По количеству используемых индексов массивы делятся на одномерные, двумерные, трёхмерные и т. д.

Форма или структура массива — сведения о количестве размерностей и размере (протяжённости) массива по каждой из размерностей; может быть представлена одномерным массивом.

Особенностью массива как структуры данных (в отличие, например, от связного списка) является константная вычислительная сложность доступа к элементу массива по индексу. Массив относится к структурам данных с произвольным доступом.[1]

В простейшем случае массив имеет константную длину по всем размерностям и может хранить данные только одного, заданного при описании, типа. Ряд языков поддерживает также динамические массивы, длина которых может изменяться по ходу работы программы, и гетерогенные массивы, которые могут в разных элементах хранить данные различных типов.

Массив — упорядоченный набор элементов, каждый из которых хранит одно значение, идентифицируемое с помощью одного или нескольких индексов. В простейшем случае массив имеет постоянную длину и хранит единицы данных одного и того же типа, а в качестве индексов выступают целые числа.

Количество используемых индексов массива может быть различным: массивы с одним индексом называют одномерными, с двумя — двумерными, и т. д. Одномерный массив — нестрого соответствует вектору в математике; двумерный («строка», «столбец») — матрице. Чаще всего применяются массивы с одним или двумя индексами; реже — с тремя; ещё большее количество индексов — встречается крайне редко.[2]

Пример фиксированного массива на языке Паскаль

{Одномерный массив целых чисел.

Нумерация элементов от 1 до 15}

a: **array** [1..15] **of** Integer;

{Двумерный массив символов.

Нумерация по столбцам по типу Byte (от 0 до 255)

по строкам от 1 до 5}

multiArray : **array** [Byte, 1..5] **of** Char;

{Одномерный массив из строк.

Нумерация по типу word (от 0 до 65536)}

rangeArray : **array** [Word] **of String**;

Пример фиксированного массива на C/C++

int Array; // Одномерный массив: целых чисел, размера 10;

```
// Нумерация элементов — от 0 до 9.

double Array; // Двумерный массив:

// вещественных чисел двойной точности,

// размера 12 на 15;

// Нумерация: по строкам — от 0 до 11,

// по столбцам — от 0 до 14.
```

В некоторых языках программирования многомерные массивы создаются на основе одномерных, у которых элементы являются массивами.[3]

Пример двумерного массива на JavaScript

```
//Создание двумерного массива чисел:

var array = [

[11, 12, 13, 14, 15, 16], // Первая строка-массив

[21, 22, 23, 24, 25, 26], // Вторая

[31, 32, 33, 34, 35, 36] // Третья

];

// Вывод массива на консоль:

array.forEach((subArray) => { // Для каждого под-массива,

subArray.forEach((item) => { // для каждого его элемента,

console.log(item); // — вывести этот элемент на консоль.

});

});
```

Поддержка индексных массивов (свой синтаксис объявления, функции для работы с элементами и т. д.) есть в большинстве высокоуровневых языков программирования. Максимально допустимая размерность массива, типы и

диапазоны значений индексов, ограничения на типы элементов определяются языком программирования и (или) конкретным транслятором.

В языках программирования, допускающих объявления программистом собственных типов, как правило, существует возможность создания типа «массив». В определении такого типа задаются типы и/или диапазоны значений каждого из индексов и тип элементов массива. Объявленный тип в дальнейшем может использоваться для определения переменных, формальных параметров и возвращаемых значений функций. Некоторые языки поддерживают для переменных-массивов операции присваивания (когда одной операцией всем элементам массива присваиваются значения соответствующих элементов другого массива).

Объявление типа «массив» в языке Паскаль

type

TArrayType = **array** [0..9] **of** Integer;

(* Массивы, имеющие заданные параметры:

1. Размер — 10 ячеек;

2. Тип элементов, пригодных для хранения —

— целые числа диапазона [−32 768; 32 767],

— объявляются типом операндов, называющимся "TArrayType". *)

var

arr1, arr2, arr3: TArrayType;

(* Объявление трёх переменных-массивов одного типа

(вышеуказанного "TArrayType"). *)

В языке программирования APL массив является основным типом данных (при этом нуль-мерный массив называется скаляром, одномерный — вектором, двумерный — матрицей). Помимо присваивания массивов в этом языке поддерживаются операции векторной и матричной арифметики, каждая из которых выполняется одной командой, операции сдвига данных в массивах, сортировка строк матрицы и

т. п.

1.2 Виды массивов

Одномерные массивы. Если за каждым элементом массива закреплен только один его порядковый номер, то такой массив называется линейным, или одномерным.[4]

Двумерный массив - это одномерный массив, элементами которого являются одномерные массивы. Другими словами, это набор однотипных данных, имеющий общее имя, доступ к элементам которого осуществляется по двум индексам. Наглядно двумерный массив удобно представлять в виде таблицы, в которой n строк и m столбцов, а под ячейкой таблицы, стоящей в i-й строке и j-м столбце, понимают некоторый элемент массива a[i][j].

Действительно, если разобраться с тем, что такое a[i] при фиксированном значении i, то увидим, что это одномерный массив, состоящий из m элементов, к которым можно обращаться по индексу: a[i][1], a[i][2], ..., a[i][m]. Схематически это вся i-я строка таблицы. Аналогично, если мы рассмотрим одномерный массив строк, то сможем заметить, что это так же двумерный массив, где каждый отдельный элемент - это символ типа char, а a[i] - это одномерный массив, представляющий отдельную строку исходного одномерного массива строк. Исходя из идеи определения думерного массива можно определить рекурентное понятие многомерного массива:

n-мерный массив - это одномерный массив, элементами которого являются (n-1)-мерные массивы.[5]

Несложно догадаться, что 3-мерный массив визуально можно представить в виде куба с ячейками (похоже на кубик Рубика), где каждый элемент имеет вид a[i][j][k]. А вот с большими размерностями возникают сложности с визуальным представлением, но математическая модель ясна.

По-другому двумерный массив также называют матрицей, а в том случае, когда n=m (число строк равно числу столбцов) матрицу называют квадратной. В матрицах можно хранить любые табличные данные: содержание игрового поля (шашки, шахматы, Lines и т.д.), лабиринты, таблицу смежности графа, коэффициенты системы линейных уравнений и т.д. Матрицы часто используют для решения олимпиадных и математических задач.

Многомерные массивы. Многомерный массив - это массив массивов, т. е. массив, элементами которого являются массивы. Размерность массива - это количество индексов, используемых для ссылки на конкретный элемент массива. Многомерные массивы объявляются точно так же, как и одномерные, только после имени массива ставится более одной пары квадратных скобок. Пример определения двухмерного массива (матрицы) с 10 строками и 30 столбцами: int array[10][30];

Фактически двухмерный массив представляется как одномерный, элементы которого тоже массивы. Константное выражение, определяющее одну из размерностей массива, не может принимать нулевое значение:

int mas[0][7]; // ошибка

int mas[I][7]; // правильно

Можно инициализировать и многомерные массивы. Причем инициализация происходит построчно, т. е. в порядке возрастания самого правого индекса. Именно в таком порядке элементы многомерных массивов располагаются в памяти компьютера.[6]

Для примера рассмотрим, как будет выполнена инициализация трехмерного массива с восемью элементами:

int array[2][2][2]={23, 54, 16, 43, 82, 12, 9, 75}; Проинициализированный массив будет выглядеть так:

[0][0][0]= =23;

[0][0][1]= =54;

[0][1][0]= =16;

……..

[1][1][0]= =9;

[1][1][1]= =75;

Для наглядности при инициализации двухмерного массива список начальных значений следует оформлять в виде таблицы: int array[3][3]={ 34, 23, 67, 38, 56, 73, 37,94,28};

Многомерные массивы могут инициализироваться и без указания одной (самой левой) из размерностей массива. В этом случае количество элементов компилятор определяет по количеству членов в списке инициализации. Например, для массива array будет получен тот же, что и в предыдущем примере результат: int array[][3]={ 34, 23, 67, 38, 56, 73, 37,94,28};

Если необходимо проинициализировать не все элементы строки, а только несколько первых элементов, то в списке инициализации можно использовать фигурные скобки, охватывающие значения для этой строки. Например, если необходимо для массива array задать начальные значения для элементов array[0][0], array[l][0], array[l][l], array[2][0], array[2][l], array[2][2], то это можно сделать следующим образом: int array[][3]=[1]; Здесь переменной int присваивается значение третьего элемента второй строки.

Динамическими называются массивы, размер которых может изменяться вовремя выполнения программы. Обычные (не динамические) массивы называют ещё фиксированными или статическими.[7]

Динамические массивы могут реализовываться как на уровне языка программирования, так и на уровне системных библиотек. Во втором случае динамический массив представляет собой объект стандартной библиотеки, и все операции с ним реализуются в рамках той же библиотеки. Так или иначе, поддержка динамических массивов предполагает наличие следующих возможностей:

Описание динамического массива. На уровне языка это может быть специальная синтаксическая конструкция, на уровне библиотеки - библиотечный тип данных, значение которого объявляется стандартным образом. Как правило, при описании (создании) динамического массива указывается его начальный размер, хотя это и не обязательно.

Операция определения текущего размера динамического массива.

Операция изменения размера динамического массива.

Ниже приведён пример конструкций для работы с динамическими массивами на Delphi.

var // Описания динамических массивов

byteArray : **Array of** Byte; // Одномерный массив

multiArray : **Array of Array of string**; // Многомерный массив

SetLength(byteArray, 1); // Установка размера массива в 1 элемент.

byteArray[0] := 16; // Запись элемента.

SetLength(byteArray, Length(byteArray)+1); // Увеличение размера массива на единицу

byteArray[Length(byteArray) - 1] := 10; // Запись значения в последний элемент.

WriteLn(byteArray[Length(byteArray) - 1]); // Вывод последнего элемента массива.

SetLength(multiArray, 20, 30); // Установка размера двумерного массива

multiArray[10,15] := 12;

SetLength(multiArray, 10, 15); // Уменьшение размера

WriteLn(Length(multiArray), ' ', Length(multiArray[0])

Гетерогенным называется массив, в разные элементы которого могут быть непосредственно записаны значения, относящиеся к различным типам данных. Массив, хранящий указатели на значения различных типов, не является гетерогенным, так как собственно хранящиеся в массиве данные относятся к единственному типу — типу «указатель». Гетерогенные массивы удобны как универсальная структура для хранения наборов данных произвольных типов. Реализация гетерогенности требует усложнения механизма поддержки массивов в трансляторе языка.[8]

1.3 Преимущество использования массивов

Преимущества массивов:

Легкость в использовании. Массивы действительно легко применять, и они присутствуют практически во всех языках программирования. Одна из причин широкого распространения массивов в том, что они используются для реализации простых линейных списков.

Быстрота изменения элементов. Массивы – это всего лишь последовательный список элементов, поэтому изменение одного из элементов происходит исключительно быстро и очень просто, так как можно легко определить

местоположение любого элемента.

Быстрое перемещение по элементам. Элементы массива располагаются в памяти друг за другом. Следовательно, организовав цикл по элементам, мы можем быстро перебрать их один за другим.[9]

Задание типа элементов. При создании массива можно описать его как строковый массив, как целочисленный массив или как-нибудь еще. В действительности, можно хранить в массиве объекты любого типа, а.NET проследит за тем, чтобы к массиву добавлялись объекты только этого типа. Это преимущество отсутствует у других типов коллекции, где можно добавлять объекты разного типа в одну коллекцию.

Недостатки массивов:

Фиксированный размер. После того, как массив создан. Нельзя автоматически изменять его размер, пытаясь добавлять элементы в его конец. Можно использовать оператор ReDim для изменения размера массива, но он медленно работает для больших массивов, и данную операцию придется выполнять в явном виде. В ряде случаев удобнее завести список, который при необходимости меняет размер.

Вставка элементов затруднительна. Для того чтобы добавить элемент между двумя существующими элементами, придется увеличить размер массива, а затем сдвинуть все элементы на одну позицию, чтобы образовать пространство для нового элемента. Одним словом, массивы не предназначены для вставки в них новых элементов.

2 Сортировка одномерного массива

2.1 Типовые алгоритмы обработки одномерных массивов и задачи, решаемые с их помощью

Решение задач, связанных с обработкой одномерных массивов, базируется на элементарных приемах обработки данных. Разбив задачу на логические части, можно значительно ускорить ее решение. Выше было отмечено, что массив — это объединение нескольких однотипных объектов, поэтому, например, группу

студентов можно рассматривать как массив строковых переменных. Типичными задачами работы с данным массивом являются определение наличия в нем заданного элемента, отбор элементов, удовлетворяющих определенным условиям, вставка и удаление элемента и т.д.[10]

Из всего многообразия существующих приемов, позволяющих манипулировать с данными, представленными в одномерных массивах, можно выделить следующие:

1. Нахождение количества элементов при заданном условии.

2. Нахождение суммы значений элементов при заданном условии.

3. Нахождение произведения значений элементов при заданном условии.

4. Поиск экстремальных значений элементов массива (поиск максимального и/или минимального значения).

5. Вставка и/или удаление значения элемента массива.

6. Формирование другого массива B из элементов массива A в соответствии с некоторым критерием.

7. Формирование массива в соответствии с определенными правилами.

8. Обмен значений элементов массива.

9. Сортировка (упорядочивание) элементов массива.

Программные реализации типовых алгоритмов обработки одномерных массивов приведены в таблице 1:

Таблица 1

Программные реализации типовых алгоритмов обработки одномерных массивов

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Заполнение массива	input n
	dim a(n)
	for i=1 to n
	input a(i)
	next i
		const n=10;
		Var a: array [1..n] of integer;
		begin
		for i:=1 to n do readln (a[i]);
		…

Вывод в строку	…for i=1 to n print a(i) ; " " ; next i	… for i:=1 to n do write (a[i]); …

Сумма, произведение элементов	…
	p=1
	for i=1 to n
	s=s + a(i)
	p=p * a(i)
	next i
		…
		s:=0; p:=1;
		for i:=1 to n do
		begin
		s:=s+a[i]); p:=p*a[i]);
		end;
		…

Выбор по условию	…
	p = 1
	for i = 1 to n
	if {условие} then k=k+1:s=s+a(i):p=p*a(i)
	next i
		…
		k:=0; s:=0; p:=1;
		for i:=1 to n do
		if {условие} then
		begin
		k:=k+1; s:=s+a[i]; p:=p*a[i];
		end;
		…

Максимальный (минимальный) элемент	…
	max = a(1): min = a(1)
	for i = 2 to n
	if a(i) > max then max = a(i)
	if a(i) < min then min = a(i)
	next i
		…
		max:=a[1]; min:=a[1];
		for i:=1 to n do
		begin
		if a[i] > max then max:=a[i];
		if a[i] < min then min:=a[i];
		end;

Вставка x на k-ое место	dim a(n + 1)
	…
	for i = n to k step -1
	a (i + 1) = a(i)
	next
	a(k) = x
		Var a: array [1..n+1] of…
		…
		for i:=n downto k do
		a[i+1]:=a[i];
		a[k]:=x;
		…

Ключевые моменты в типовых алгоритмах:

Выбор по условию. В качестве условия может проверяться значение элемента массива на четность, кратность элемента какому-либо числу, положительность, отрицательность, равенство нулю. Может проверяться также и значение индекса элемента массива (например, элементы, стоящие на четных местах и др.).[11]

Максимальный (минимальный) элемент. Кроме максимального элемента часто требуется найти и индекс максимального элемента:

if a[i]>max then begin

max:=a[i]; imax:=i;

end;

Вставка x на k-ое место. Перестановка элементов (для освобождения "места" для вставляемого элемента) происходит с конца массива - последний элемент передвигается на "пустое место", на его место передвигается предпоследний элемент и т.д. Инвертирование *элементов. Цикл* работает n/2 раз, так как за один проход мы меняем сразу два элемента местами.

2.2 Задачи с использованием типовых алгоритмов обработки одномерных массивов

На плоскости изображено N прямоугольников. Каждый прямоугольник задан координатами левой нижней и правой верхней вершин. Определить, имеют ли прямоугольники общую площадь.

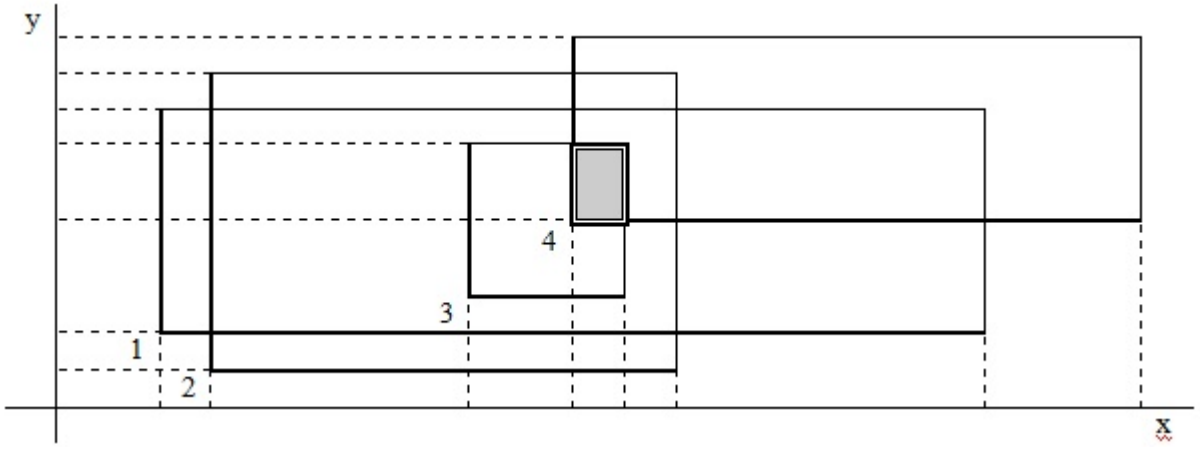


Рис. 1. Плоскость прямоугольника

Идея решения: Если:

- максимальная координата по оси X левых нижних вершин прямоугольников будет меньше минимальной координаты правых верхних вершин и …

- максимальная координата по оси У левых нижних вершин прямоугольников будет меньше минимальной координаты правых верхних вершин, то …

- …общая площадь есть.

В задаче необходимо использовать типовой алгоритм нахождения МАКСИМАЛЬНОГО (МИНИМАЛЬНОГО) ЭЛЕМЕНТА МАССИВА.

Для вычисления общей площади необходимо найти произведение разности: максимальной координаты по оси X левых нижних вершин прямоугольников и минимальной координаты правых верхних вершин и …

…максимальной координаты по оси У левых нижних вершин прямоугольников и минимальной координаты правых верхних вершин.

Программа на Бейсике:

```
input "введите количество прямоугольников"; n

dim x1(n), x2(n),y1(n), y2(n)

for i=1 to n

input x1(i), x2(i), y1(i), y2(i)

next

xmax=x1(1)

xmin=x2(1)

ymax=y1(1)

ymin=y2(2)

for i=1 to n
```

```
if x1(i) > xmax then xmax=x1(i)

if x2(i) < xmin then xmin=x2(i)

if y1(i) > ymax then ymax=y1(i)

if y2(i) < ymin then ymin=y2(i)

next

if xmax<xmin and ymax<ymin then print "общ. площадь есть" else print "общ.
площади нет"
```

Программа на Паскале:

```
var x1, x2, y1, y2: array [1..10] of integer;

n, i, xmax, xmin, ymax, ymin: integer;

begin

writeln ('введите количество прямоугольников');

readln (n);

for i:=1 to n do

readln (x1[i], y1[i], x2[i], y2[i]);

xmax:=x1[1];

xmin:=x2[1];

ymax:=y1[1];

ymin:=y2[2];

for i:=1 to n do

begin

if x1[i] > xmax then xmax:=x1[i];

if x2[i] < xmin then xmin:=x2[i];
```

```
if y1[i] > ymax then ymax:=y1[i];

if y2[i] < ymin then ymin:=y2[i];

end;

if (xmax<xmin) and (ymax<ymin) then writeln ('общая площадь есть')

else writeln ('общей площади нет');

end.
```

Тест:

Таблица 2

Тест

Дано:	3 1,1,9,5 2,3,5,6 4,2,7,4	4 2,2,5,4 4,3,7,6 6,1,11,2 6,4,12,8

Результат: Общая площадь есть Общей площади нет

Задача: Латинским квадратом называется массив, в строках и столбцах которого нет одинаковых элементов. Вывести на экран латинский квадрат размером NxN.

Идея решения: Заполнить 1 строку квадратного массива (NxN) числами от 1 до N. Вторая строка массива получается путем циклического сдвига элементов первой строки и т. д. (табл. 3). Циклический сдвиг можно реализовать, используя типовой алгоритм ВСТАВКИ-УДАЛЕНИЯ (в зависимости от направления циклического сдвига).

Таблица 3

Заполнение латинского квадрата путем циклического сдвига элементов

1 2 3 4 5

5 1 2 3 4

4 5 1 2 3

3 4 5 1 2

2 3 4 5 1

Решение на Бейсике:

```
input "размерность="; n
dim a(n,n)
for j=1 to n
a(1,j)=j
next
rem======сдвиг==========
for i=2 to n
for j=1 to n
a(i,j)=a(i-1,j)
next
x=a(i,n)
for j=n to 2 step -1
a(i,j)=a(i,j-1)
```

next

a(i,1)=x

next

rem=====вывод==========

for i=1 to n

for j=1 to n

print a(i,j);

next

print

next

Решение на Паскале:

```
var a: array [1..10,1..10] of integer;
n,i,j,x: integer;
begin
writeln ('размерность=');
readln (n);
for j:=1 to n do a[1,j]:=j;
{======сдвиг======}
for i:=2 to n do
begin
for j:=1 to n do a[i,j]:=a[i-1,j];
x:=a[i,n];
for j:=n downto 2 do
```

```
a[i,j]:=a[i,j-1];

a[i,1]:=x;

end;

{======вывод======}

for i:=1 to n do

begin

for j:=1 to n do write(a[i,j]);

writeln;

end;

end.
```

3 Разработка алгоритма решения задачи

3.1 Описание программы

Данная программа называется “TEST” (имя исполняемого файла test.exe) и предназначена для анализа методов сортировки одномерного массива. Программа работает на IBM совместимых компьютерах семейства x86 начиная с 286 и выше, в операционной системе типа Ms-DOS 3.0 и выше.

Программа содержит пять основных функций: main(), file(), qsort(), srecmg(), grafix(). Все переменные, используемые в программе, являются локальными.

Функция main() содержит пункты меню и вызывает другие исполняемые функции в зависимости от нажатия предложеных функциональных клавиш F1, F2 и F10. Каждая из этих функций работает автономно и независимо от двух других.

Программа реализована как в псевдографическом, так и в графическом режиме (в связи с чем она может компилироваться только в DOSовских версиях BorlandC++ или BorlandC). В графическом режиме она выводит на экран гистограмму которая характеризует время сортировки массивов двумя способами.

Программа использует как стандартный, так и файловый ввод-вывод информации. Стандартный ввод представлен запросом программы на ввод функциональных клавиш, которые задают характер выполняемого действия. Файловый ввод-вывод используется в функции help(), для вывода на экран информации о разработчике программы , её функциональных клавишах и о возможных ошибках в процессе выполнения. Кроме того программа работает с функцией времени clock() и переменными времени типа clock_t.

Так же программа содержит стандартные функции языка Си, описанные в библиотеках: <string.h>, <stdlib.h>, <time.h>, <stdio.h>, <conio.h>,<graphics.h>. Ниже перечислены библиотеки с функциями и дано краткое описание использованных в программе стандартных функций.

Из библиотеки <time.h>:

clock() – эта функция возвращает время , фиксируемое процессором от начала счета программы , или –1, есле оно не известно . Для возвращения этого времени в секундах применяется формула clock()/CLK_TCK.

Из библиотеки <stdlib.h>:

rand() – эта функция выдаёт псевдо случайное число в диапазоне от 0 до RAND_MAX не меньше 32767 .

exit() – вызывает нормальное завершение программы .

Из библиотеки <stdio.h>:

printf() – эта функция осуществляет вывод строки на экран.

fopen() – эта функция открывает файл с заданным именем и возвращает поток или NULL, если попытка открытия оказалась неудачной .

fclose() – эта функция производит дозапись ещё незаписанных буферизированных данных , сбрасывает нечитанный буферизированный ввод , освобождает все автоматические запрошенные буфера , после чего закрывает поток . Возвращает EOF в случае ошибки и 0 в противном случае.

fgetc() – эта функция возвращает следующую литеру из потока stream в виде unsigned char или EOF, если исчерпан файл или обнаружена ошибка .

puts() – пишет стринг s и литеру новая – строка в stdout . Возвращает EOF в случае ошибки, или неотрицательное значение, если запись прошла нормально.

Из библиотеки <conio.h>

textbackground() – с помощью этой функции устанавливается цвет фона для функции cprintf().

textcolor() – с помощью этой функции устанавливается цвет текста для функции cprintf().

clrscr() – функция очистки экрана, цветом установленным функцией textbackground().

cprintf() – с помощью этой функции осуществляется вывод строки с учётом цветов установленных функциями textbackground(), textcolor().

_setcursortype() – с помощью данной функции осуществляется изменение режима отображения курсора. Данных режимов в Си всего три – NOCURSOR (курсор выключен), SOLIDCURSOR (курсор в виде сплошного блока) NORMALCURSOR (обычный курсор).

getch() – функция getch осуществляет считывание первого единственного символа с клавиатуры, используется при считывании клавиш курсора при перемещении по окну выбора режима работы программы.

gotoxy() – эта функция перемещает курсор в нужную часть экрана, обычно используется перед функцией cprintf().

В этой библиотеке описаны все символические константы цветов используемые функциями textbackground(), textcolor(). В ней также описаны все типы курсоров используемых функцией _setcursortype().

Функция main имеет тип void и является функцией меню. Main выполняет опрос клавиатуры и в зависимости от нажатой функциональной клавиши выполняется соответствующее действие (вызов помощи, тестирования и выход из программы). Эта возможность реализована благодаря конструкции множественного выбора switch. Функция имеет одну локальную переменную press имеющую тип char. Она воспринимает символ с клавиатуры без вывода на экран и используется в конструкции switch при переходе к другой выполняемой функции. В данной функции вызывается вспомогательная функция windows(), которая создаёт

псевдографический интерфейс при запуске программы. При выборе пункта выхода из программы стандартная функция textbackground() создаёт черный экран ,а функция exit() совершает выход из программы.

При вызове функции помощи (help()) программа обращается к этой функции, которая считывает и выводит информацию файловым способом. Вызываемый фаил хранится под именем test.hlp и при его отсутствии выдаётся окно сообщения: «Фаил text.hlp не найден».

Вызов функции построения гистограмм выполняется при нажатии клавиши F2. При этом функция main() обращается к функции file() .

Для построения упорядоченного списка B’ из списка B=<к1,к2,...,kn> при сортировке слиянием список B делится на n подсписков B1=<к1>, B2=<к2> , ... , Bn=<kn> длины 1. Потом совершается процедура прохождения в которой m≥2 упорядоченных списков B1,B2,...,Bm меняются на m/2 (или (m+1)/2) упорядоченных списков которые создаются слиянием B2i-1и B2i (2i≤m) и суммированием Bm с непарным m. Процесс повторяется до появления одной последовательности длины m. Количество действий , необходимых для сортировки слиянием , равна n log2n, потому что за одно прохождение выполняется n сравнений , а всего необходимо осуществить log2n прохождений . Сортировка слиянием достаточно эффективно и используется при больших значения n.

Функция srecmg() является рекурсивной , и сортирует массив a[n]. За каждым вызовом массив для сортировки делится на две части – от a[0] до

a[i-1] и от a[i] до a[n] , каждая из которых сортируется отдельно , а потом выполняется их слияние. Слияние выполняется при помощи вызываемой функции merge() в которую передаётся указатель на массив , текущий номер элемента массива и количество элементов массива . Параметрами функции merge() являются массив w[ ] ,его длина l/2 и длина первой части массива l1 .Функция merge() выполняет слияние подмассивов, образуя на их месте упорядоченный массив w[0],w[1],...,w[l2-2],w[l2-1] .

Быстрая сортировка состоит в том, что список B=<k1,k2,...,kn> реорганизуется в B’,<k1>,B”,где B’ – подсписок B с элементами , не большими k1 , а B” – подсписок B с элементами большими k1. В списке B’,<k1>,B” элемент K1 уже находится на месте где он должен быть в отсортированом списке. Дальше к спискам B’ и B’’ применяется упорядовачивание быстрой сортировкой.

Рекурсивная функция qqsort упорядовачает быстрой сортировкой участок массива целых чисел первый элемент которого указывается показателем v[left], последний – показателем v[right].

Если участок массива более чем из двух элементов, v[left] – low >1, то находится делящий элемент и переносится в V[0]. Переменной last присваивается значение первого элемента массива left. Затем массив делится на две части, элементы которых соответственно больше и меньше last. Далее функция выполняет перезапоминание делящего элемента и вызывает себя для разбитых подсписков.

Обмен элементов выполняется при помощи функции swap() , в которую из функции qqsort() передаётся указатель на массив и элементы , место-положение которых в массиве нужно обратить.

Время построения списка зависит от его начального состояния. Время будет минимальным, если каждый шаг раздела дает подсписки B’, B’’ приблизительно одинаковой длины; тогда необходимо (n log2 n) шагов. Если начальный список мало чем отличается от обычного отсортированного, то необходимо (n^2)/2 шагов. Быстрая сортировка требует дополнительной памяти порядка O (log2n) для исполнения рекурсивной функции qqsort() (неявного стэка).

Функция grafix() имеет тип void и параметр simbol – массив скорости выполнения сортировки. Эта функция вызывается при построении гистограммы и работает в графическом режиме о чем информирует строка initgraph(&gdriver, &gmode, ""), которая устанавливает систему в графический режим , определяет характеристику видеодрайвера. Если переменная errorcode не получает значение grOk, то дальнейшее выполнение программы невозможно так как графический режим не установлен (о чем выводится сообщение). В массиве simvol[] определяется больший элемент , столбец которого устанавливается в 100%.

Строка: bar3d(midx + otst + siz * n, midy -CopySimvol[n], midx + siz* (n+1), midy, 15, 1); создаёт 3D-изображения гистограмм, высота которых определяется массивом CopySimvol[n].

Цвет выводимых элементов изображения устонавливает функция setcolor(), а все выводимые линии строятся функцией line(). Текст выводится при помощи функции outtextxy(). Если текст должен выводиться вертикально то функции settextstyle() задаётся параметр VERT_DIR.

После вывода на экран изображения выполняется опрос клавиатуры и если пользователем была нажата клавиша “ESC”, то программа возвращается в функцию file() и дальше в функцию main(), где снова ожидает нажатия необходимой клавиши.

Функция closegraph() закрывает графический режим.

3.2 Руководство программиста и оператора

Данная программа предназначена для анализа методов сортировки массивов быстрой и слиянием. Программа может работать на IBM совместимых компьютерах семейства x86 начиная с 286 и выше, под управлением операционных систем Ms-DOS 3.0 и выше и Windows 9x. Данная программа компилировалась с использованием Borland C++ 3.1. Компиляция программы в версиях Borland C++ сконфигурированных под Windows(таких как Borland C++4.5, Borland C++5.2 и выше) не возможна так как графический режим [2] функционирует только в версиях сконфигурированных под DOS.

Программа не требует от пользователя ввода массива для его сортировки. Этот массив создается специальной функцией языка Си – генератор случайных чисел. Программа была разработана на компьютере с низкой тактовой частотой(75МГц). А так как в программе используется секундомер, то тактовая частота компьютера, на котором демонстрируется программа, влияет на точность выводимых результатов. Поэтому не советуется пользоваться ею на компьютерах с тактовой частотой выше 150МГц. Хотя в противном случае скорость сортировки значительно увеличивается.

Листинг программы приведён в приложении 1.

Программа не предусмотрена для работы в режиме командной строки. Если вводимая пользователем функциональная клавиша не предусмотрена программой, то она выполняться не будет до тех пор, пока пользователь не введет соответствующий символ. Если программа не находит некоторых нужных для ее выполнения файлов, то выдается окно сообщения об ошибке с текстом причины. Функция error() вызывается везде, где появляется ошибка.(создает окно сообщения). В случае необходимости программу можно остановить в любом месте её исполнения следующими комбинациями клавиш: Ctrl C или Alt X.

Вызов программы осуществляется путём запуска файла test.exe, при этом программа будет работать в интерактивном режиме.

Окно помощи программы содержит: название программы, данные о разработчике, назначение, функциональные клавиши, используемые в программе, и возможные проблемы при ее выполнении.

Основной функцией данной программы является определение времени сортировки массивов методами быстрой и слиянием. Путем незначительных изменений в программе, можно приспособить программу специально для сортировки массивов. Данная программа (test.exe) является единым исполняемым модулем и не требует наличия любых других установленных программных средств. Она так же не требует установки на компьютер, каких бы то ни было специфических аппаратных средств.

Контрольный пример выполнения программы приведён в приложении 2.

Программа может работать лишь в интерактивном режиме. Сортировка массива с большим числом элементом на современном компьютере займет всего несколько секунд и зависит от размера сортируемого массива.

После загрузки программы оператору будет предложено нажать необходимую клавишу для продолжения выполнения программы, для вывода информации о программе или для выхода. Если программа не содержит файла text.hlp или не найден драйвер EGAVGA.BGI, то программа выдаёт окно сообщения об ошибке. Если программа содержит все необходимые элементы, то она выдаст окно сообщения о возможности выполнения анализа сортировки. Если программа получила разрешение на начало процесса сортировки, то она выполняет его и после завершения выводит на экран в графическом режиме результаты о анализе сортировок. В случае необходимости программу можно остановить в любом месте её исполнения следующими комбинациями клавиш: Ctrl C или Alt X. В таком случае программа не выполнит ни каких действий.

Окно помощи программы содержит: название программы, данные о разработчике, назначение, функциональные клавиши, используемые в программе, и возможные проблемы при ее выполнении.

Заключение

С понятием «массив» приходится сталкиваться при решении научно-технических и экономических задач обработки совокупностей большого количества значений. В общем случае массив – это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип.

Массивы могут быть одномерными и многомерными (двух-, трехмерными и т. д.). Одномерный массив — массив, с одним параметром, характеризующим количество элементов одномерного массива. Фактически одномерный массив — это массив, у которого может быть только одна строка, и n-е количество столбцов. Столбцы в одномерном массиве — это элементы массива.

Примером одномерных массивов может быть список фамилий учеников класса, многомерных - таблица умножения, классный журнал, аттестат зрелости. Элементы, образующие массив, упорядочены таким образом, что каждому элементу соответствует номер (индекс), определяющий его местоположение в общей последовательности. Доступ к каждому элементу осуществляется путём индексирования.

В результате выполнения курсовой работы была написана программа, анализирующая сортировку массивов способами быстрой и слиянием. Программа обладает высоким параметром быстродействия, маленьким размером и не требовательна к системным ресурсам компьютера. В качестве недостатка программы можно отнести то, что точность выполнения программы зависит от тактовой частоты компьютера. Этот недостаток можно решить путём изменения количества сортируемых элементов массива. Программа может быть преобразована для использования в целях сортировки массивов, вводимых пользователем.

Список использованных источников

1. Абрамов, С.А. Математические построения и программирование / С.А. Абрамов. - М.: Наука, 2016. - 192 с.
2. Бекишев, Г.А. Элементарное введение в геометрическое программирование / Г.А. Бекишев, М.И. Кратко. - М.: Наука. Главная редакция физико-математической литературы, 2017. - 144 с.
3. Ван, Тассел Д. Стиль, разработка, эффективность, отладка и испытания программ / Ван Тассел Д.. - М.: Мир, 2017. - 332 с.
4. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. - М.: Мир, 2016. - 360 с.

5. Голицына, О.Л. Основы алгоритмизации и программирования: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум; Издание 2-е, 2015. - 432 с.
6. Готье, Р. Руководство по операционной системе UNIX / Р. Готье. - М.: Финансы и статистика, 2014. - 232 с.
7. Гребенников, Л.К. Программирование микропроцессорных систем на языке ПЛ/М / Л.К. Гребенников, Л.А. Летник. - М.: Финансы и статистика, 2014. - 160 с.
8. Дж., Вандер Плас Python для сложных задач. Наука о данных и машинное обучение / Дж. Вандер Плас. - М.: Питер, 2017. - 518 с.
9. Жильцов, В. В. Информационные технологии в проектировании «интеллектуальной» скважины / В.В. Жильцов. - М.: Университет, 2014. - 906 с.
10. Карпов, В.Я. Алгоритмический язык Фортран / В.Я. Карпов. - М.: Наука, 2014. - 192 с.
11. Крамм Программирование в Access для "чайников" / Крамм, Роб. - М.: Диалектика, 2016. - 304 с.
12. Кук, Даррен Машинное обучение с использованием библиотеки H2O / Даррен Кук. - М.: ДМК Пресс, 2017. - 310 с.
13. Линдси, Ч. Неформальное введение в Алгол 68 / Ч. Линдси, Ван Дер Мюйлен, С.. - М.: Мир, 2018. - 408 с.
14. Мельчук, И.А. Автоматический синтаксический анализ / И.А. Мельчук. - М.: Редакционно-издательский отдел Сибирского отделения АН СССР, 2018. - 358 с.
15. Неслуховский, К.С. Пособие по программированию для ЭЦВМ "Минск-32" / К.С. Неслуховский. - М.: Советское радио, 2016. - 296 с.
16. Попов, И. И. Использование семантических подходов в экономических моделях / И.И. Попов. - М.: Университет, 2016. - 646 с.
17. Рихтер Программирование на платформе Microsoft. NET Framework / Рихтер, Джеффри. - М.: Русская Редакция, 2014. - 512 с.
18. Скотт, Т. Основы программирования. Курс программированного обучения / Т. Скотт. - М.: Советское радио, 2016. - 490 с.
19. Фаронов, В.В. Основы Турбо-Паскаля / В.В. Фаронов. - М.: МВТУ-Фесто дидактик, 2015. - 304 с.

Приложения

Приложение 1

ЛИСТИНГ ПРОГРАММЫ

// листинг программы сортировки массивов разработанная студентом

```
#include <stdio.h>
#include <dos.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <time.h>
//-------------< Создание оболочки >-------------
void windows(int w)
{
int n;
_setcursortype(0);
window(1, 1, 80, 25); // Выделение окна
textbackground(BLACK); // Цвет фона
clrscr(); // Очистка экрана
window(1, 25, 80, 25); // Выделение окна
textbackground(GREEN); // Цвет фона
clrscr(); // Очистка экрана
window(1, 25, 80, 25);
textcolor(BLACK); // Цвет текста
if (w == 1) // Проверка на выбор окна
{
n = 21;
```

```
cprintf(" Помощь Тест Выход");
window(2, 25, 4, 25);
textcolor(RED); // Управляющие клавиши
cprintf("F1"); // основного окна
window(13, 25, 15, 25);
cprintf("F2");
window(22, 25, 25, 25);
cprintf("F10");
textbackground(BLUE);
}
else
{
n =22;
cprintf(" Выход из помощи ");
window(3, 25, 6, 25);
textcolor(RED); // Управляющие клавиши
cprintf("Esc"); // окна помощи
textbackground(CYAN);
}
window(1, 1, 80, 25); // Прорисовка рамки
textcolor(WHITE);
cprintf("+---------------------------------- Тест -----------------------------------+");
for (int k = 0; k < n; k++)
```

```
cprintf("¦ ¦");
cprintf("+------------------------------------------------------------------------------+");
if (w == 1)
{
window(2, 2, 79, 2);
puts(" Эта программа демонстрирует сортировку массива двумя методами:");
window(2, 3, 79, 3);
puts(" быстрым методом и методом слияния. После чего определяется время сор-");
window(2, 4, 79, 4);
puts(" тировки массива каждым методом и результат выводится в виде гисто-");
window(2, 5, 79, 5);
puts(" граммы.");
window(2, 6, 79, 6);
window(20, 10, 60, 15);
textcolor(WHITE);
textbackground(LIGHTGRAY);
cprintf("+----------------------------------------------------------+");
cprintf("¦ НЕОБХОДИМЫЕ ФАЙЛЫ ПРИСУТСТВУЮТ ¦");
cprintf("¦ (для тестировния нажмите F2) ¦");
cprintf("+----------------------------------------------------------+");
closegraph();
}
}
```

```
//------------< Окно сообщения ошибок>-----------
void Error()
{
window(20, 10, 60, 15);
textcolor(WHITE);
textbackground(LIGHTGRAY);
cprintf("+----------------- Ошибка ----------------+");
cprintf("¦ ¦ ");
cprintf("¦ ¦");
cprintf("¦ ¦");
cprintf("+-----------------------------------------+");
}
//-------------< Функция помощи >----------------
help()
{
int n = 1;
FILE *hl; // Указатели на файл
char string[78];
if ((hl = fopen("test.hlp", "r")) != NULL) // Проверка на открытие файла
{
windows(0);
window(2, 2, 78, 23);
textcolor(BLACK);
```

```
while (fgets(string, 78, hl) != NULL && n < 23)
{
gotoxy(1, n++); // Построчный вывод файла
cputs(string); // помощи
}
window(36, 1, 44, 1);
printf(" Помощь "); // Вывод заголовка помощи
while (27 != getch());
}
else{
Error();
window(29, 12, 52, 12);
textcolor(BLACK);
cprintf("Файл TEST.HLP не найден");
getch();
windows(1);
}
fclose(hl);
windows(1);
return 0;
}
//--------< Функция построения гистограмм>-------
grafix(double simvol[2])
```

```
{
double CopySimvol[2]; // Масив количества символов
long float max = 0;
int gdriver = DETECT, gmode, errorcode;
int midx = 50; // Обявление переменных
int midy = 410; // с заданними начальными
int i, s; // значениями
int siz = 100;
int otst = 10;
int rovn = 45;
char chis[2];
char buf[10];
initgraph(&gdriver, &gmode, "");
errorcode = graphresult(); // Запись код ошибки
if (errorcode != grOk) // Проверка на ошибку
{
Error(); // Вызов функции окна
window(26, 12, 54, 12);
textcolor(BLACK);
cprintf("Драйвер EGAVGA.BGI не найден");
getch();
windows(1);
return 0;
```

```
}
for (int y = 0; y < 2; y++) // Оприделение максимального
if (max < simvol[y]) // числа
max = simvol[y];
for (int b = 0; b < 2; b++) // Оприделение высоты столбцов
CopySimvol[b] = simvol[b] * 200 / max;
setfillstyle(CLOSE_DOT_FILL,9);
for (int n = 0; n < 2; n++) // Построение столбцов и линий
{
setcolor(BLUE);
bar3d(midx + otst + siz * n, midy - CopySimvol[n], midx + siz* (n+1 ), midy, 15, 1);
setcolor(BROWN);
line(midx + rovn + siz * n, midy + otst, midx + rovn + siz * n, midy + otst * 2);
sprintf(chis, "%d", n + 1);
setcolor(GREEN);
outtextxy((midx + rovn + siz * n) - 2, midy + otst * 2, chis);
setcolor(CYAN);
sprintf(buf, "%lf", simvol[n]);
outtextxy((midx + rovn + siz * n) - 15, midy - CopySimvol[n] - rovn, buf);
}
setcolor(BROWN);
line(midx, 100, midx, midy + otst); // Построение оси Y
line(midx, midy + otst, 280, midy + otst); // Построение оси X
```

```
line(midx - otst, midy - 200, midx, midy - 200); // Построение
line(midx - otst, midy - 100, midx, midy - 100); // линии
settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
setcolor(GREEN);
outtextxy(535, 460, "ESC ");
outtextxy(10, 205, "100");
outtextxy(10, 305, "50");
outtextxy(350, 235, "1. Сортирвка массива быстрым");
outtextxy(350, 250, " методом");
outtextxy(350, 280, "2. Сортирвка массива методом");
outtextxy(350, 295, " слияния");
setcolor(LIGHTBLUE);
outtextxy(300, 423, "метод");
outtextxy(570, 460, "Выход");
settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
outtextxy(220, 30, "ГИСТОГРАММА ");
settextstyle(DEFAULT_FONT, VERT_DIR, 1);
outtextxy(48, 160, "время");
while (27 != getch()); // Проверка на символ ESC
closegraph();
windows(1);
return 0;
}
```

```
/*qsort:сортирует v[left]...v[right] по возрастанию*/
void qqsort( int v[], int left, int right)
{
int i,last;
delay(1);
void swap( int v[], int i, int j);
if(left>=right) /*ничего не делается если*/
return; /* в массиве менее двух эл-тов */
swap(v,left,(left+right)/2); /*делящий эл-нт переносится в v[0]*/
last=left;
for(i=left+1;i<=right;i++) /*деление на части*/
if(v[i]<v[left])swap(v,++last,i);
swap(v,left,last); /*перезапоминается делящий элемент*/
qqsort(v,left,last-1);
qqsort(v,last+1,right);
}
void swap( int v[], int i, int j)
{
long int temp;
temp=v[i];
v[i]=v[j];
v[j]=temp;
}
```

```
/*SRECMG -- РЕКУРСИВНАЯ СОРТИРОВКА СЛИЯНИЕМ*/
void srecmg(a,n)
int a[],n;
{
void merge( int*, int, int);
int i;
delay(1);
if(n>1)
{i=n/2;srecmg(a,i);srecmg(a+i,n-i);merge(a,i,n);}
}
/*merge--слияние двух подсписков*/
#define MAX(x,y) ((y)<(x)?(x):(y))
void merge( int*w, int l1, int l2)
{
int*a,*pa,*pb,i;
a=( int*)calloc(l2+2,sizeof( int));
pa=a;pb=a+l1+1;
for(i=0;i<l1;i++) *pa++=w[i];
for(i=l1;i<l2;i++) *pb++=w[i];
*pa=*pb=MAX(w[l1-1],w[l2-1])+1;
pa=a;pb=a+l1+1;
for(i=0;i<l2;i++)
w[i]=(*pa<*pb?*pa++:*pb++);
```

```
free(a);
}
#define ww 700
//-------< Функция вызова разных методов >-------
file()
{ void qqsort( int *, int,int);
void srecmg( int*, int);
double simvol[2];
int s;
clock_t start,start2,end,end2; int t=0;
int gener1[ww],gener2[ww]; //Генератор случайных чисел
randomize(); //Устанавливает генератор в 0
for ( s=0 ; s < ww ; s++)
{ gener1[s]= ( rand()%100 ); // rand()-функция генератора
gener2[s] =gener1[s];
}
{ start=clock();
qqsort(gener1,0,ww-1);
end=clock();
simvol[0] = ((end - start)/CLK_TCK);
}
{ start2 =clock();
srecmg(gener2,ww);
```

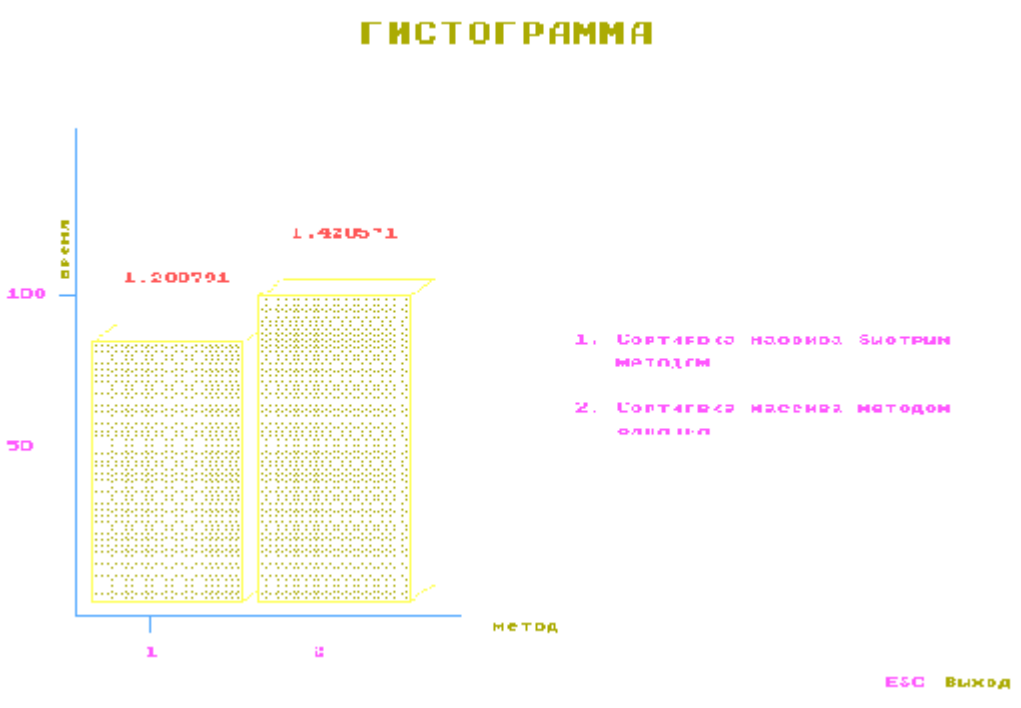
```
end2=clock();
simvol[1] = ((end2 - start2)/CLK_TCK);
}
grafix(simvol); // Вызов функции построения гистограмм
windows(1);
return 0;
}
//-------------------< Меню >--------------------
void main()
{
char press;
windows(1);
while (1)
press = getch(); // Опрос клавиатуры
switch(press)
{
case 59: help(); break; // Вызов помоши
case 60: file(); break; // Запуск гистограммы
case 68: { // Выход из программы
window(1, 1, 80, 25);
textbackground(BLACK);
clrscr();
exit(1);
```

} } }}

Приложение 2

КОНТРОЛЬНЫЙ ПРИМЕР ВЫПОЛНЕНИЯ ПРОГРАММЫ

В качестве примера возьмём исходный файл “Test.exe” и запустим его. На экране появляется окно сообщения о наличии необходимых файлов. Для продолжения выполнения программы пользователь нажимает клавишу **F2**, в результате чего на экране появляется гистограмма, характеризующая скорость выполнения сортировки массивов. Воспользовавшись клавишей **Esc**, пользователь выходит с графического режима в режим отображения меню. При нажатии пользователем клавиши F**1** на экране появляется окно помощи, которое содержит название программы, данные о разработчике, назначение, функциональные клавиши, используемые в программе, и возможные проблемы при ее выполнении. Нажатие



из программы

Пример выводимой

гистограммы

1. Крамм Программирование в Access для "чайников" / Крамм, Роб. - М.: Диалектика, 2016. - 304 с. ↑

2. Скотт, Т. Основы программирования. Курс программированного обучения / Т. Скотт. - М.: Советское радио, 2016. - 490 с. ↑

3. Гребенников, Л.К. Программирование микропроцессорных систем на языке ПЛ/М / Л.К. Гребенников, Л.А. Летник. - М.: Финансы и статистика, 2014. - 160 с. ↑

4. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. - М.: Мир, 2016. - 360 с. ↑

5. Голицына, О.Л. Основы алгоритмизации и программирования: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум; Издание 2-е, 2015. - 432 с. ↑

6. Неслуховский, К.С. Пособие по программированию для ЭЦВМ "Минск-32" / К.С. Неслуховский. - М.: Советское радио, 2016. - 296 с. ↑

7. Рихтер Программирование на платформе Microsoft. NET Framework / Рихтер, Джеффри. - М.: Русская Редакция, 2014. - 512 с. ↑

8. Абрамов, С.А. Математические построения и программирование / С.А. Абрамов. - М.: Наука, 2016. - 192 с. ↑

9. Жильцов, В. В. Информационные технологии в проектировании «интеллектуальной» скважины / В.В. Жильцов. - М.: Университет, 2014. - 906 с. ↑

10. Мельчук, И.А. Автоматический синтаксический анализ / И.А. Мельчук. - М.: Редакционно-издательский отдел Сибирского отделения АН СССР, 2018. - 358 с. ↑

11. Фаронов, В.В. Основы Турбо-Паскаля / В.В. Фаронов. - М.: МВТУ-Фесто дидактик, 2015. - 304 с. ↑