# Лабораторная работа № 3.1 Сортировка массивов

**Цель лабораторной работы**

Закрепление теоретических знаний и практических навыков при работе с массивами, изучение алгоритмов сортировки массивов.

**Постановка задачи**

Массив - набор элементов одного и того же типа, объединенных общим именем. Массивы в С# можно использовать по аналогии с тем, как они используются в других языках программирования. Однако С#-массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой. Рассмотрим различные типы массивов.

**Одномерные массивы**

Одномерный массив - это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в С# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Одномерный массив в С# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размер массива. В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

|  |  |
| --- | --- |
| Форма записи | Пояснения |
| базовый\_тип [] имя\_\_массива; Например: int [] a; | Описана ссылка на одномерный массив, которая в дальнейшем может быть использована: 1. для адресации на уже существующий массив; 2. передачи массива в метод в качестве параметра 3. отсроченного выделения памяти под элементы массива. |
| базовый\_тип [] имя\_\_массива = new базовый\_тип [размер]; Например: int []a=new int [10]; | Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю. Замечание. Надо отметить, что в C# элементам массива присваиваются начальные значения по умолчанию в зависимости от базового типа. Для арифметических типов - нули, для ссылочных типов - null, для символов - пробел. |
| базовый\_тип [] имя\_\_массива= {список инициализации}; Например: int []a={0, 1, 2, 3}; | Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. |

Обращения к элементам массива происходи с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, a[0], b[10], c[i]. Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Рассмотрим несколько простых примеров работы с одномерными массивами.

**Многомерные массивы**

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент массива имеет два индекса, первый определяет номер строки, второй - номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля. Объявить двумерный массив можно одним из предложенных способов: тип [,] имя\_\_массива; тип [,] имя\_\_массива = new тип [размер1, размер2]; тип [,] имя\_\_массива={{элементы 1-ой строки}, … , {элементы n-ой строки}}; тип [,] имя\_\_массива= new тип [,]{{элементы 1-ой строки}, … ,{элементы n-ой строки}}; строки}}; Например: int [,] a; int [,] a= new int [3, 4]; int [,] a={{0, 1, 2}, {3, 4, 5}}; int [,] a= new int [,]{{0, 1, 2}, {3, 4, 5}}; Замечания. 1. Как и в случае с одномерными массивами, последние два описания являются избыточными. 2. При работе с многомерными массивами можно использовать приемы, которые мы рассмотрели для одномерных массивов. 3. При обращении к свойству Length для двумерного массива мы получим общее количество элементов в массиве. Чтобы получить количество строк нужно обратиться к методу GetLength с параметром 0. Чтобы получить количество столбцов - к методу GetLength с параметром 1.

**Оператор foreach и его использование при работе с массивами**

Оператор foreach применяется для перебора элементов в специальным образом организованной группе данных, в том числе и в массиве. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять перебор по индексу - мы просто указываем на необходимость перебрать все элементы группы. Синтаксис оператора: foreach ( in ) где имя определяет локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из указанной группы, а тип соответствует базовому типу элементов группы. Ограничением оператора foreach является то, что с его помощью можно только просматривать значения элементов в группе данных, но нельзя их изменять.

**Методы сортировки данных C#**

- Сортировка пузырьком;

- Шейкерная сортировка;

- Сортировка вставками;

- Гномья сортировка;

- Сортировка простым выбором;

- Поразрядная сортировка;

- Сортировка Шелла;

- Пирамидальная сортировка;

- Быстрая сортировка.

**Сортировка пузырьком**

- простейший алгоритм сортировки. Эффективен для небольшого объёма данных, а при сортировке большого объема данных могут быть большие потери времени.

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N+1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде, отсюда и название алгоритма).

**Шейкерная сортировка** (сортировка перемешиванием)

- разновидность пузырьковой сортировки.

Перестановка элементов в шейкерной сортировке выполняется аналогично той же в пузырьковой сортировке, т. е. два соседних элемента, при необходимости, меняются местами. Суть модификации заключается в реализации двунаправленности: алгоритм перемещается, ни как в пузырьковой сортировке – строго снизу вверх (слева направо), а сначала снизу вверх, потом сверху вниз.

**Сортировка вставками** (сортировка простыми включениями)

- алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

На вход алгоритма подаётся некая последовательность. На каждом шаге алгоритма выбирается один из элементов, входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан.

**Гномья сортировка**

- алгоритм сортировки, похожий на сортировку вставками, но в отличие от последней перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком

Алгоритм находит первое место, где два соседних элемента стоят в неправильном порядке и меняет их местами. Он пользуется тем фактом, что обмен может породить новую пару, стоящую в неправильном порядке, только до или после переставленных элементов.

**Сортировка выбором**

Шаги алгоритма:

1) находим номер минимального значения в текущем списке

2) производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)

3) теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

**Поразрядная сортировка**

- алгоритм сортировки за линейное время.  
Алгоритм сортирует числа по разрядам.

Сортировка Шелла

Идея сортировки методом Шелла состоит в том, чтобы сортировать элементы отстоящие друг от друга на некотором расстоянии step. Затем сортировка повторяется при меньших значениях step, и в конце процесс сортировки Шелла завершается при step = 1 (а именно обычной сортировкой вставками).

**Пирамидальная сортировка**

Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево — это такое дерево, у которого выполнены условия:

1) Каждый лист имеет глубину либо d, либо d-1; d — максимальная глубина дерева.

2) Значение в любой вершине не меньше (другой вариант — не больше) значения её потомков.

**Быстрая сортировка**

- Один из самых быстрых известных универсальных алгоритмов сортировки массивов.

Пошаговое описание алгоритма

1. Из массива выбирается элемент a[i]. Как правило, в качестве этого элемента берется центральный элемент массива.

2. Остальные элементы распределяются таким образом, чтобы слева от a[i] оказались все элементы, меньшие или равные a[i]. Элементы, большие или равные a[i], помещаются справа.

3. Производится проверка количества элементов в левой и правой частях массива. Если какая-либо часть (или обе части) содержит более двух элементов, то для этой части (или частей) запускается та же процедура сортировки с помощью рекурсивного вызова.

На небольшом объеме данных наиболее хорошие результаты показывают простейшие алгоритмы сортировки - вставками, пузырьком. Однако отличие во времени от остальных методов не очень существенно. На больших же объемах данных - сортировка пузырьком, шейкерная, гномья и, отчасти, сортировки вставками и выбором показывают существенное отличие во времени.

**Задание на лабораторную работу**

1. Разработать меню для выбора задания (рисунок 1), включая задания из лабораторной работы 3.2.
2. Разработать код для обработки одномерного массива.
3. Разработать код для обработки двумерного массива.
4. **Не использовать** встроенные функции сортировки!

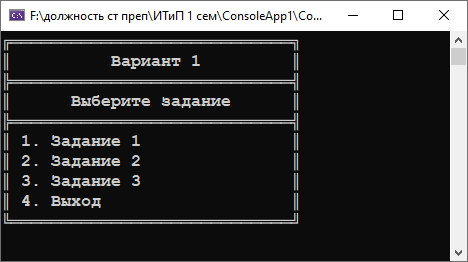


Рисунок 1 – Пример меню выбора задания

**Варианты заданий на лабораторную работу**

**Задание 1.**

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- сумму отрицательных элементов массива;

- произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- сумму положительных элементов массива;

- произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение элементов массива с четными номерами;

- сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные нулю, считать положительными).

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение элементов массива с нечетными номерами;

- сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает единицу. Освободившиеся в конце массива элементы заполнить нулями.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- максимальный элемент массива;

- сумму элементов массива, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- минимальный элемент массива;

- сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер максимального элемента массива;

- произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер минимального элемента массива;

- сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- максимальный по модулю элемент массива;

- сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- минимальный по модулю элемент массива;

- сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер минимального по модулю элемента массива;

- сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер максимального по модулю элемента массива;

- сумму элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a, b], а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, лежащих в диапазоне от A до B;

- сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, равных нулю;

- сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, больших C;

- произведение элементов массива, расположенных после максимального по модулю элемента.

Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом – все положительные (элементы, равные нулю, считать положительными).

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество отрицательных элементов массива;

- сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество положительных элементов массива;

- сумму элементов массива, расположенных после последнего элемента, равного нулю.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает единицу, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, меньших C;

- сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение отрицательных элементов массива;

- сумму положительных элементов массива, расположенных до максимального элемента.

Изменить порядок следования элементов в массиве на обратный.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение положительных элементов массива;

- сумму элементов массива, расположенных до минимального элемента.

Упорядочить по возрастанию отдельного элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

**Задание 2.**

1. Дана целочисленная прямоугольная матрица. Определить:

- количество строк, не содержащих ни одного нулевого элемента;

- максимальное из чисел, встречающихся в заданной матрице более одного раза.

1. Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих не одного нулевого элемента. Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.
2. Дана целочисленная прямоугольная матрица. Определить:

- количество столбцов, содержащих хотя бы один нулевой элемент;

- номер строки, в которой находится самая длинная серия одинаковых элементов.

1. Дана целочисленная квадратная матрица. Определить:

- произведение элементов в тех строках, которые не содержать отрицательных элементов;

- максимум среди сумм элементов диагоналей. параллельных главной диагонали матрицы.

1. Дана целочисленная квадратная матрица. Определить:

- сумму элементов в тех столбцах, которые не содержать отрицательных элементов;

- минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

1. Дана целочисленная прямоугольная матрица. Определить:

- сумму элементов в тех строках, которые содержат хотя бы один отрицательных элемент;

- номера строк и столбцов всех седловых точек матрицы.

Матрица А имеет седловую точку Aij, если Aij является минимальным элементом в i-й строке и максимальным – в j-м столбце.

1. Для заданной матрицы размером 8x8 найти такие k, при которых k-я строка матрицы совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержать хотя бы один отрицательный элемент.
2. Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержать хотя бы один отрицательный элемент.
3. Соседями элемента Aij в матрице назовем элементы Akl, где i – 1 ≤ k ≤ i + 1, j – 1 ≤ l ≤ j + 1, (k,l) ≠ (i,j). Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы размером 10x10.
4. Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей (определение соседних элементов см. в варианте 9). Подсчитать количество локальных минимумов заданной матрицы размером 10x10. Найти сумму модулей элементов, расположенных выше главной диагонали.
5. Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду. Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.
6. Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.
7. Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо или вниз (в зависимости от введенного режима). n может быть больше количества элементов в строке или столбце.
8. Осуществить циклический сдвиг элементов квадратной матрицы размером MxN вправо на k элементов таким образом: элементы первой строки сдвигаются в последний столбец сверху вниз, из него – в последнюю строку справа налево, из нее – в первый столбец снизу вверх, из него – в первую строку; для остальных элементов – аналогично.
9. Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент. Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.
10. Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке. Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.
11. Путем перестановки элементов квадратной вещественной матрицы добиться того, чтобы ее максимальный элемент находился в левой верхнем углу, следующий по величине – в позиции (2, 2), следующих по величине – в позиции (3, 3) и т.д., заполнив таким образом всю главную диагональ. Найти номер первой из строк, не содержащих ни одного положительного элемента.
12. Дана целочисленная прямоугольная матрица. Определить:

- количество строк, содержащих хотя бы один нулевой элемент;

- номер столбца, в котором находится самая длинная серия одинаковых элементов.

1. Дана целочисленная квадратная матрица. Определить:

- сумму элементов в тех строках, которые не содержат отрицательных элементов;

- минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

1. Дана целочисленная прямоугольная матрица. Определить:

- количество отрицательных элементов в тех строках, которые содержать хотя бы один нулевой элемент;

- номера строк и столбцов всех седловых точек матрицы.

Матрица А имеет седловую точку Aij, если Aij является минимальным элементом в i-й строке и максимальным – в j-м столбце.

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2017 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Браузер (Свободно распространяемое программное обеспечение).