МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний університет «Одеська юридична академія»

Факультет кібербезпеки та інформаційних технологій

Кафедра інформаційних технологій

Протокол практичної роботи №3

з дисципліни об'єктно-орієнтоване програмування

на тему: «Робота з масивами»

Виконав студент групи

ІПЗ-212

Корнійчук М. М.

Прийняв

Рудніченко М. Д.

Одеса, 2022

ЗМІСТ

[ВСТУП](#_ki7i591yqske) 3

[ТЕОРЕТИЧНА ЧАСТИНА](#_335513nxkbjl) 4

[ПРАКТИЧНА ЧАСТИНА](#_9cm35ikas1ko) 7

[ВИСНОВОК](#_xlxs31hqwy0n) 12

[ЛІТЕРАТУРА](#_5m79jhxx931e) 13

# 

# ВСТУП

Мета роботи – навчитися працювати з масивами засобами Java.

# ТЕОРЕТИЧНА ЧАСТИНА

**Методы поиска элементов одномерного массива.**

Применяются два метода поиска элементов одномерного массива:

Метод последовательного перебора – для неупорядоченного массива. Метод бинарного поиска - для упорядоченного массива.

**Метод последовательного перебора**

В общей постановке задача поиска заключается в нахождении наименьшего индекса (i) элемента массива, равного заданному значению х, т.е. в определении местоположения конкретного значения х в массиве b. В алгоритмах поиска последовательным перебором широко используется прием так называемого барьерного элемента. Его суть заключается в включении в качестве n элемента массива значения x; (a(n)=x), что упрощает запись условия выхода из цикла. Если при поиске последовательным перебором наибольшее число сравнений n, то при двоичном количество этих сравнений составляет log2 n.

В практике обработки информации в целях значительного ускорения поиска данные предварительно упорядочивают в порядке возрастания или убывания значений. Затем к такой последовательности применяется метод двоичного (бинарного) поиска. По данному методу вся последовательность делится пополам, устанавливается, к какой половине принадлежит значение х . Таким образом одна половина массива отсекается из дальнейшего поиска. Та половина, в которой ведется поиск, в свою очередь, делится пополам и повторно выясняется, в какой половине массива необходимо вести поиск значения х. Данная процедура продолжается либо до совпадения значения элемента массива a и х , либо до достижения конца интервала поиска.

**Алгоритмы сортировки массивов**

Сортировка - это процесс обработки последовательности данных с целью их перестановки в определенном порядке. Например, перестановка может быть в порядке убывания значений последовательностей (сортировка от большего к меньшему) или в порядке возрастания (сортировка от меньшего к большему). В качестве признака сортировки выбираются специальные ключи записи. Скажем, при сортировке записей рабочих ключами могут быть стаж работы, возраст, уровень квалификации, цех и т.д. Примеры, рассматриваемые в данном параграфе, показаны применительно к значениям массивов без выделения специальных ключей, так как целью изложения является разбор распространенных алгоритмов внутренней сортировки, т.е. без использования внешних устройств.

Обычно сортировка преследует цель облегчения последующего поиска нужных данных в отсортированной (сгруппированной) последовательности объектов. В этом смысле алгоритмы сортировки присутствуют практически во всех задачах обработки данных. Поэтому методам сортировки и анализу их эффективности посвящено достаточно работ научного и практического характера.

Главными требованиями к алгоритмам сортировки массивов являются экономное использование памяти и быстродействие. Экономия памяти предполагает, что сортировка выполняется на том же месте, где хранится исходный массив. Пересылать элементы из одного массива «a» в другой массив «b» не разрешается.

Методы сортировки на том же месте в зависимости от приема перемещения элементов массива делятся на три основных класса: включением; выбором; обменом.

Сортировка простыми включениями (метод вставки) заключается в следующем. Элементы исходного массива условно рассматриваются в двух последовательностях: готовая упорядоченная последовательность a(l),a(2),...,a(i-l) и входная последовательность a(i),a(i+l),...,a(n), значения которых в порядке следования необходимо вставить в соответствующие места готовой последовательности путем сдвига вправо (вниз) тех или иных элементов. Для такой вставки в цикле, начиная от i=2 (готовая последовательность содержит 1 элемент, j=l) с шагом 1, каждый элемент a (i) входной последовательности передается и вставляется в нужное место готовой последовательности. Если сортируются элементы массива в порядке возрастания, то при вставке x=a(i) место вставки определяется как только найден элемент готовой последовательности a(j) <= х. Сортировка выбором наименьшего элемента основана на следующем алгоритме:

определяются наименьший элемент и его координата в заданном массиве;

этот наименьший элемент меняется местами с первым элементом а(1);

затем определяются наименьший элемент и его координата среди оставшихся n-1 элементов. Он меняется со вторым элементом а (2);

и так данная процедура повторяется со всеми элементами, пока не останется только один наибольший элемент. Когда необходимо выполнить сортировку в порядке убывания значений элементов массива, то определяется максимальный элемент вместо минимального и применяются описанные действия.

Заметим, что в целом алгоритм сортировки выбором является предпочтительней алгоритма сортировки простым включением. Вместе с тем применительно к более или менее рассортированным массивам сортировка простым включением работает несколько быстрее.

Сортировка простым обменом также является распространенным, легким для понимания и коротким с точки зрения записи алгоритма методом. Его сущность заключается в сравнении и обмене пары смежных (соседних) элементов до полного их упорядочения.

Данный метод называется также методом "пузырька". (Назван так, поскольку после каждой итерации наибольший элемент "всплывает" наверх.). На каждой итерации i=l, 2, n-1 последовательно сравниваются пары соседних элементов a(j) и a(j+1), j=l, 2, 3, ...,n-i, и если a(j)>a(j+1), то они переставляются. Таким образом, после каждой итерации по i наибольший элемент оказывается на своем месте в конце массива ("всплывает" вверх). Данный алгоритм имеет недостаток, заключающийся в том, что обычно последние проходы никак не влияют на порядок элементов массива, так как они уже рассортированы. Для улучшения алгоритма устанавливается специальный флажок, фиксирующий какой-либо обмен. Если не было обмена за весь проход, то это свидетельствует об упорядоченности элементов. Широко применяется также сортировка с использованием индекса. Суть ее заключается в следующем:

создается дополнительный одномерный целочисленный массив, в который заносятся индексы данного массива, в том порядка, в котором они находятся в данном массиве.

в процессе упорядочения элементов данного массива изменяется расположение не элементов данного массива, а индексов в массиве индексов.

# ПРАКТИЧНА ЧАСТИНА

Завдання 1. Вариант 2

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1) сумму положительных элементов массива;

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

Код програми (рис. 1)



Рисунок . Код програми

Результат роботи програми наведено на рис. 2.

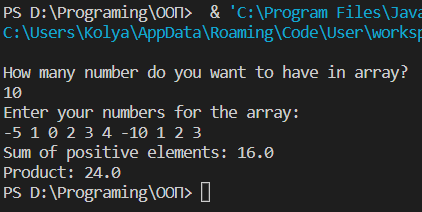


Рисунок . Приклад роботи програми до завдання 1

Завдання 2. Дана целочисленная прямоугольная матрица. Определить:

1)количество столбцов, не содержащих ни одного нулевого элемента.

2)сформировать одномерный массив из сумм элементов тех строк, которые содержат хотя бы один положительный элемент.

Код програми (рис. 3)



Рисунок . Код програми до завдання 2

Результат роботи програми наведено на рис. 4.

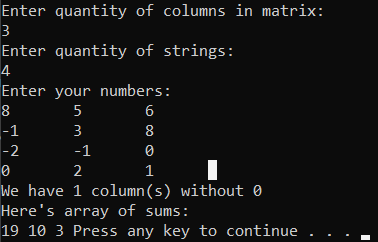


Рисунок 4. Результат виконання програми для завдання 2

# ВИСНОВОК

В ході роботи я поняття масиву та особливості його функціонування як структури даних. Дізнався, які особливості має використання масивів у Java. Навчився створювати двовимірні масиви і маніпулювати даними, що знаходяться в них, для вирішення прикладних задач.

# 

# ЛІТЕРАТУРА

1. Васильев А. Н. Самоучитель Java с примерами и программами. 3-е издание. — СПб.: Наука и Техника, 2016. — 368 с.: ил.

Посилання на github: <https://github.com/chuguystyr/University.OOP/tree/%D0%9F%D0%A0-1>