**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: “Одномерные статические массивы”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение внутреннего строения массива, понимания, как хранятся типы данных в массиве.

**Основные теоретические положения.**

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов.

Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение. Нумерация ячеек начинается с нуля.

Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

Различия в объявлении переменной и объявлении одномерного массива размером 5:

int a = 0;

int a[5] = {5, 4, 3, 2, 1};

Представить массив в виде пронумерованных ячеек можно, например, так:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *i* | 0 | 1 | 2 | 3 | 4 |
| A(*i)* | 5 | 4 | 3 | 2 | 1 |

Объявить двумерный массив размером 25 можно так:

int a[5] [5];

Иногда возникает необходимость работы с одномерным массивом, в котором элементы расположены в порядке возрастания. Для того, чтобы выставить элементы любого массива в порядке возрастания, были придуманы алгоритмы сортировок.

Существует несколько алгоритмов сортировок, каждая имеет свои плюсы и минусы, и каждая работает быстрее в определённых типах массивов. (например, сортировка расчёской работает быстрее в таком массиве, где максимальный и минимальный элементы расположены далеко от друг друга)

Я рассмотрю несколько самых известных сортировок:

* Bubble Sort
* Shaker Sort
* Comb Sort
* Insert Sort
* Quick Sort

**Bubble Sort**

Простейшая из сортировок – сортировка обменом (известна как пузырьковая сортировка). Вся суть метода заключается в попарном сравнении элементов и последующем обмене. Таким образом, если следующий элемент меньше текущего, то они меняются местами, максимальный элемент массива постепенно смещается в конец массива, а минимальный – в начало. Один полный проход по массиву может гарантировать, что в конце массива находится максимальный элемент.

Затем процесс повторяется до тех пор, пока вся последовательность не будет упорядочена. Важно заметить, что после первого прохода по массиву, уже имеется один упорядоченный элемент, он стоит на своем месте, и менять его не надо. Таким образом на следующем шаге будут сравниваться N-1 элемент.

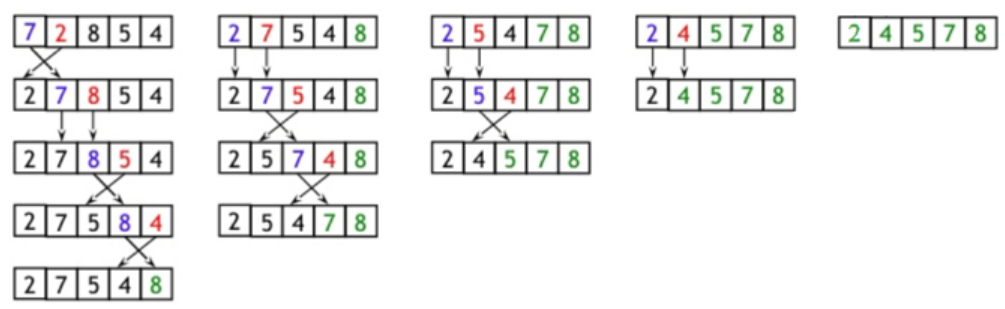
Плюсы алгоритма:

- лёгкий в понимании

Минусы алгоритма:

- медленный

- перестановка только на одну позицию



**Shaker Sort**

Shaker sort – модификация пузырьковой сортировки. Принцип работы этой сортировки аналогичен bubble sort: попарное сравнение элементов и последующий обмен местами, но с некоторыми отличиями: как только максимальный элемент становится на свое место, алгоритм не начинает новую итерацию с первого элемента, а запускает сортировку в обратную сторону. Алгоритм гарантирует, что после выполнения первой итерации, минимальный и максимальный элемент будут в начале и конце массива соответственно.

Затем процесс повторяется до тех пор, пока массив не будет отсортирован. За счет того, что сортировка работает в обе стороны, массив сортируется на порядок быстрее.

Плюсы алгоритма:

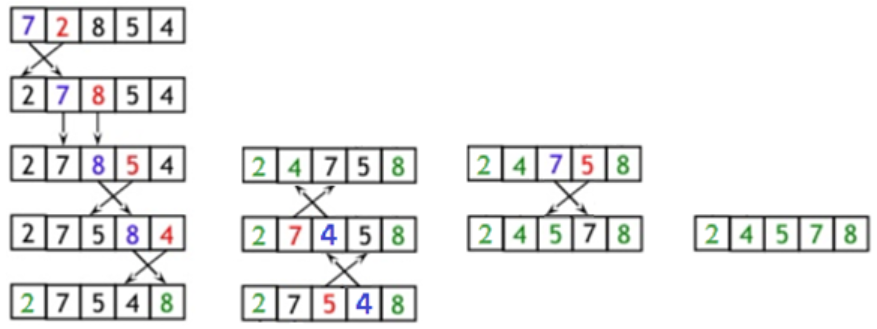
- лёгкий

- быстро сортирует массивы, в которых максимальные и минимальные элементы стоят далеко от своих позиций

Минусы алгоритма:

- медленный

- перестановка только на одну позицию



**Comb Sort**

Очевидный недостаток bubble и shaker sort заключается в том, что элементы переставляются максимум на одну позицию.

Comb sort (сортировка расческой) – ещё одна модификация сортировки пузырьком. Алгоритм был разработан специально для случаев, когда минимальные элементы стоят слишком далеко, или максимальные – слишком близко к началу массива. В сортировке расческой переставляются элементы, стоящие на расстоянии.

Оптимально изначально взять расстояние равным длине массива , а далее уменьшать его на определенный коэффициент, который примерно равен **1.247**. Когда расстояние станет равно 1, выполняется обычная сортировка пузырьком.

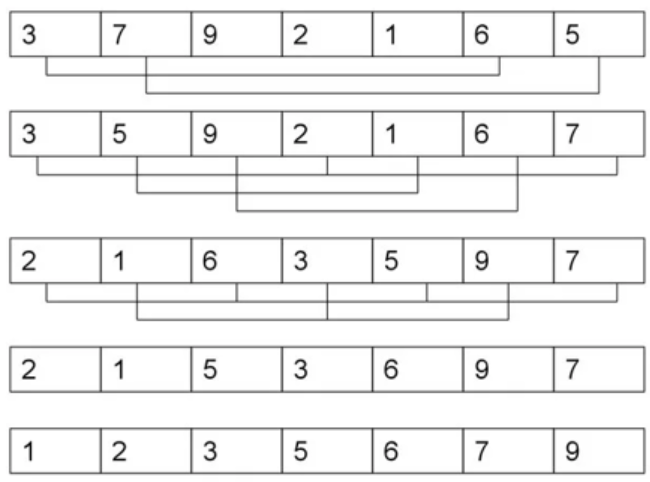
Плюсы алгоритма:

- перестановка не на одну позицию

- быстро сортирует массивы, в которых максимальные и минимальные элементы стоят далеко от своих позиций

Минусы алгоритма:

- медленный



**Insert Sort**

Сортировка вставками (insert sort) – алгоритм сортировки, в котором элементы массива просматриваются по одному, и каждый новый элемент размещается в подходящее место среди ранее упорядоченных элементов.

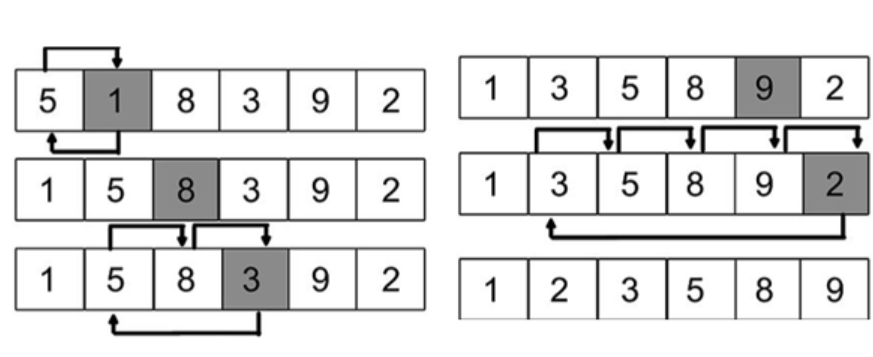
Общая суть сортировки вставками такова:

1) Перебираются элементы в неотсортированной части массива.

2) Каждый элемент вставляется в отсортированную часть массива на то место, где он должен находится.

Плюсы алгоритма:

- лёгкий



**Quick Sort**

Быстрая сортировка (quick sort) – одна из самых быстрых сортировок. Эта сортировка по сути является существенно улучшенной версией алгоритма пузырьковой сортировки.

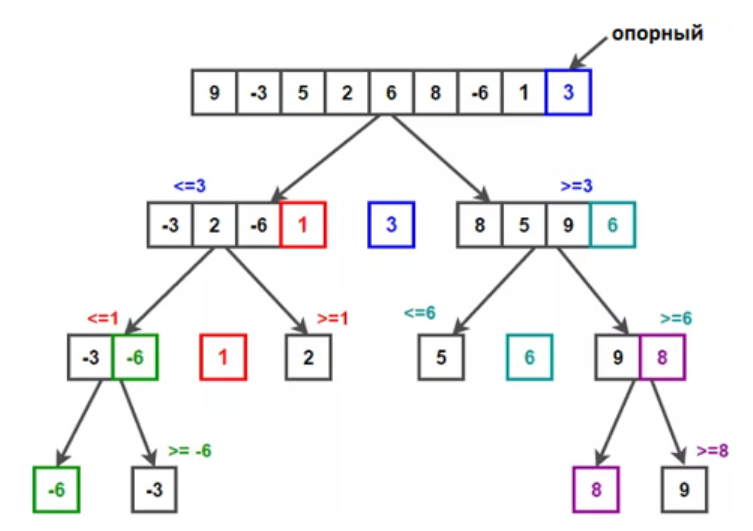
Общая идея алгоритма состоит в том, что сначала выбирается из массива элемент, который называется опорным. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность. Затем необходимо сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующие друг за другом: меньше опорного, раны опорному и больше опорного. Для меньших и больших значений необходимо выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

Плюсы алгоритма:

- быстро работает практически во всех массивах

Минусы:

- нелёгкая реализация



Иногда возникает необходимость отыскать индекс числа в отсортированном массиве. Для этого был придуман алгоритм бинарного поиска:

1) Определение значения в середине массива (или иной структуры данных). Полученное значение сравнивается с ключом (значением, которое необходимо найти).

2) Если ключ меньше значения середины, то необходимо осуществлять поиск в первой половине элементов, иначе – во второй.

3) Поиск сводится к тому, что вновь определяется значение серединного элемента в выбранной половине и сравнивается с ключом.

4) Процесс продолжается до тех пор, пока не будет определен элемент, равный значению ключа или не станет пустым интервал для поиска.

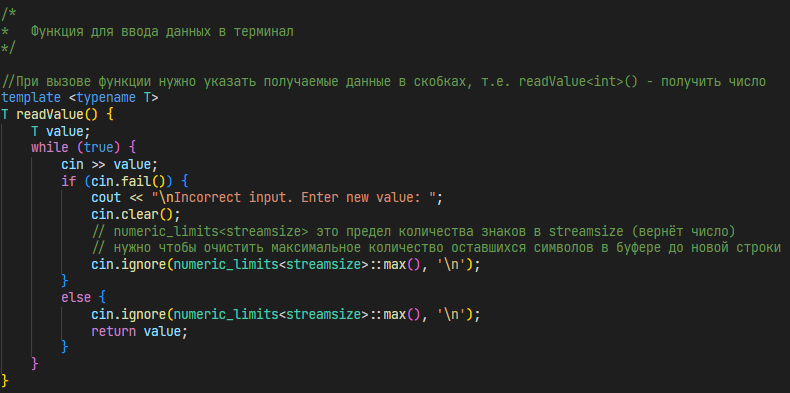
**Постановка задачи.**

Необходимо создать массив и заполнить его случайными числами, затем отсортировать пятью разными алгоритмами, предварительно замерив потраченное на сортировку время.

1. Создание массива с фиксированным количеством элементов N.
2. Заполнение массива через его итерацию и функцию rand();
3. Создание меню выбора сортировки
4. Реализация алгоритмов сортировки

**Выполнение работы.**

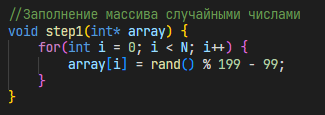
Вспомогательная функция ввода в терминал и получения из него данных в переменную (принимает только указанный тип, иначе выводит ошибку):



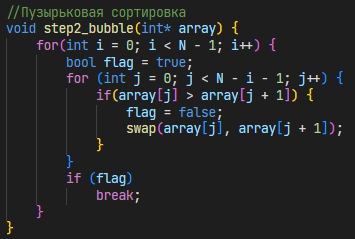
Меню выбора функционала программы. Для перехода нужно указать число 0-11.



Запонение массива:

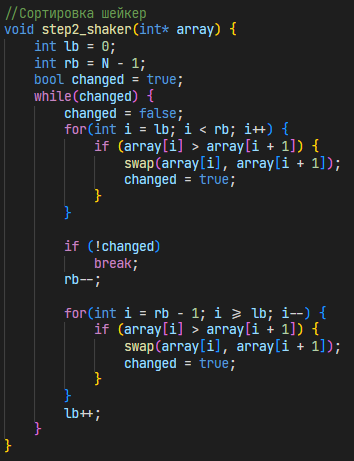


Сортировка массива пузырьком (с флагом):



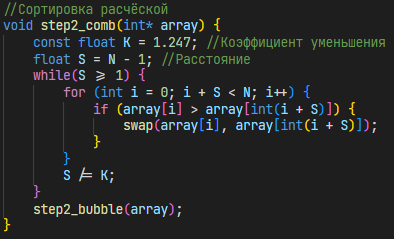
Если в итерации не были найдены числа, стоящие не на своём месте, то flag будет равен true, и алгоритм завершит работу. Таким образом, не придётся проходить остальные итерации i.

Сортировка-Шейкер:



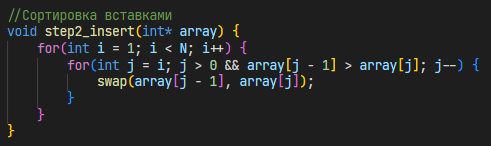
Так как алгоритм гарантирует, что после первой итерации минимальный и максимальный элементы будут находится соответственно в начале и в конце массива, то мы можем установить “рамки”, в которых массив будет сортироваться, чтобы уменьшить время работы алгоритма. Эти рамки это lb (левый край) и rb (правый край).

Сортировка расчёской:



Изначально берётся расстояние, равное длине массива. Далее сравниваются элементы на расстоянии от i. По завершении сортировки выполняется обычная сортировка пузырьком, т.к. Comb Sort сам по себе не гарантирует абсолютную сортировку массива.

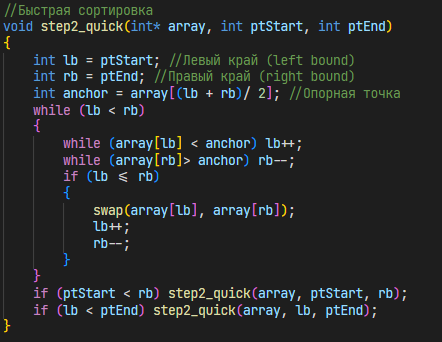
Сортировка вставкой:



Сортировка вставкой проходит по всем элементам массива и переставляет их в нужные позиции (идущие перед i).

Условие проверки вынесено в цикл for.

Быстрая сортировка:



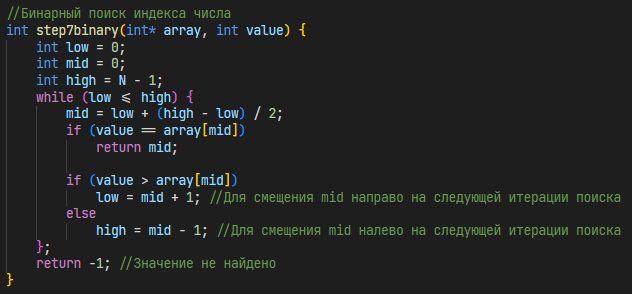
Сначала выбирается опорный элемент anchor (изначально стоит в середине).

Уже в цикле while “рамки” массива сужаются: слева от опорной точки остаются значения, меньшие, чем опорный элемент, а справа те, которые больше;

После этого числа на границах меняются местами (пока границы рамок не встретятся).

Далее выполняется сортировка двух половин через рекурсию, та, которая меньше оп. Точки и та, которая больше оп. Точки.

Бинарный поиск:



Изначально объявляются переменные low, mid, high. Low и high определяют границы поиска, а mid указывает на середину между границами.

В начале каждой итерации вычисляется позиция mid из текущих low и high.

Далее число в середине сравнивается с тем, которое нужно найти:

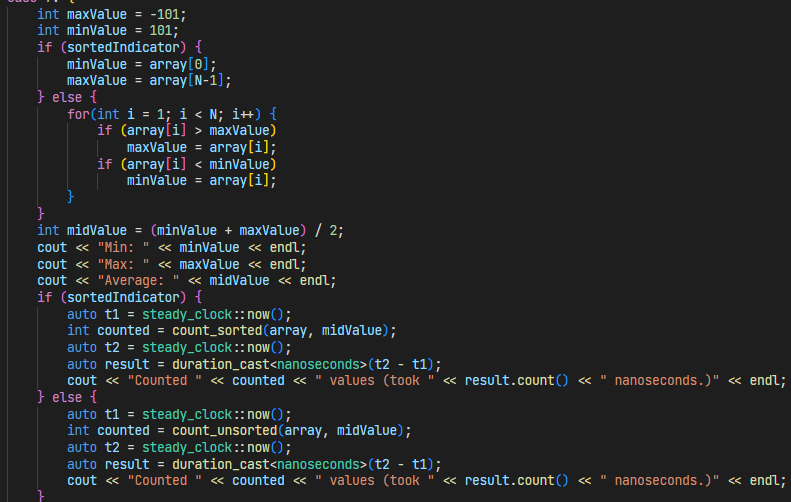
Если число совпадает, то функция возвращает индекс числа.

Если число больше, чем то, что нужно найти, то нижняя граница устанавливается на mid + 1, таким образом устанавливая область поиска во вторую половину

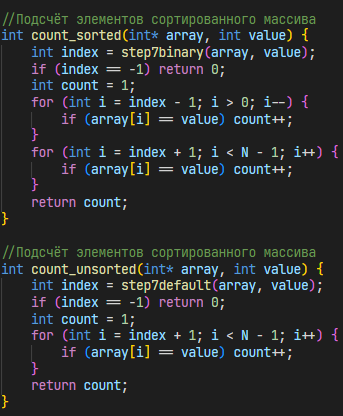
Если число меньше, то верхняя граница устанавливается на mid – 1, таким образом устанавливая область поиска в первую половину.

Далее идёт следующая итерация до тех пор, пока число не будет найдено или границы не встретятся (числа нет в массиве)

Функция поиска среднего значения:



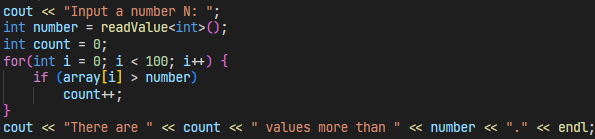
Изначально находим максимальное и минимальное значения (самым быстрым способом, смотря на то, отсортирован массив или нет), далее идёт подсчёт оптимальным способом для двух состояний массива:

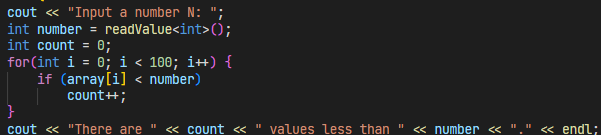


В отсортированном массиве сначала находится индекс искомого числа с помощью бинарного числа, затем идёт подсчёт от этого индекса в обе стороны массива.

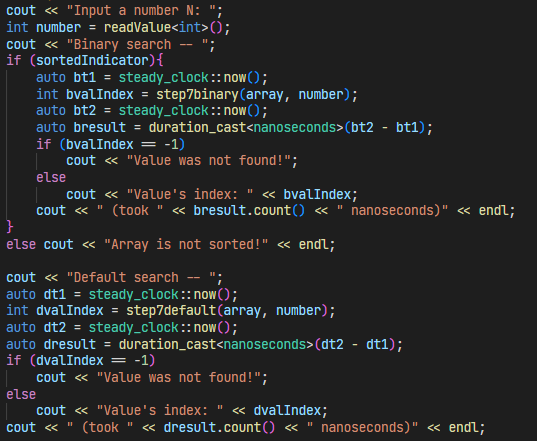
В неотсортированном массиве простой перебор массива.

Функции нахождения количества чисел, больших и меньших, чем N:



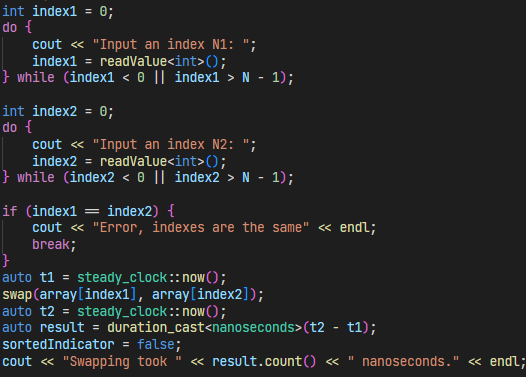


Простой перебор и подсчёт.



Сравнение бинарного поиска (step7binary) и обычного перебора (step7default), замер выполнения функций.

Функция обмена по двум индексам:



Проверка на правильность введённых данных и замер обмена.

**Выводы.**

Я научился работать с массивами, изучил некоторые способы сортировок, их плюсы и минусы, а также научился применять бинарный поиск.