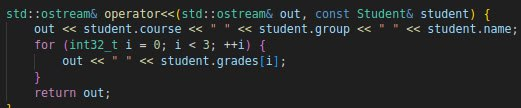
Отчёт

В процессе практики были реализованы сортировки (7 штук), они адаптированы к 5 типам данных, для сортировки студентов была применена перегрузка оператора сравнения, научили > < << работать с типом данных студенты.

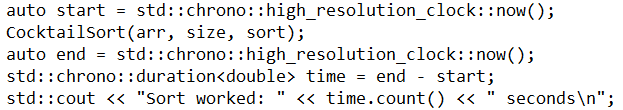




При сравнении значений внутри сортировок был использована функция сравнения, она была вынесена для гибкости кода.

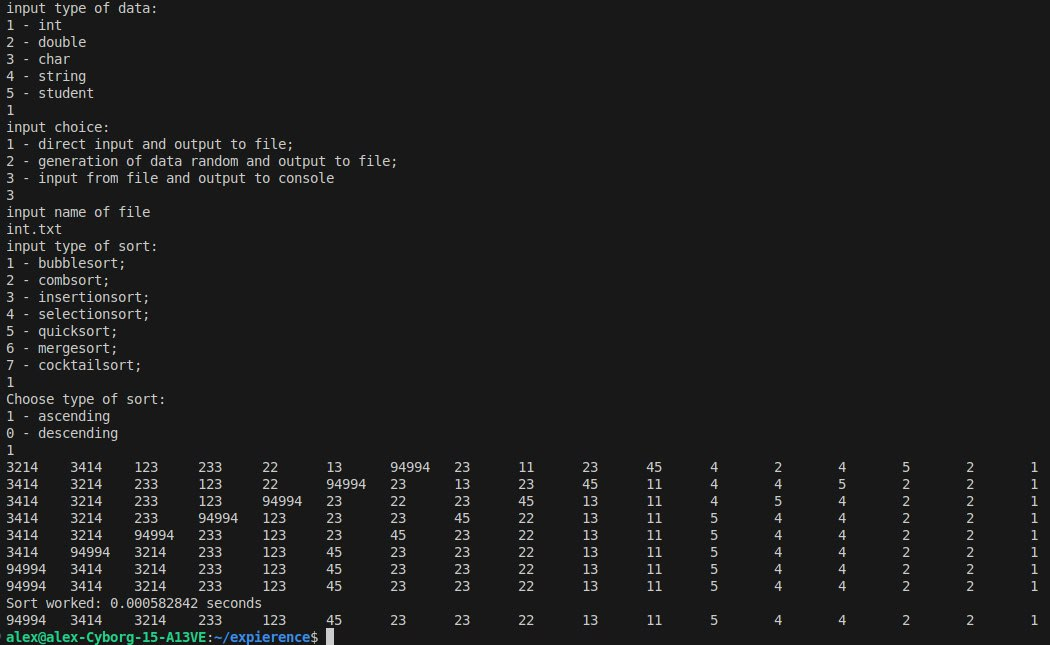


Так же полученная программа предусматривает измерение времени выполнения функций сравнение, это было реализовано при помощи библиотеки хроно.



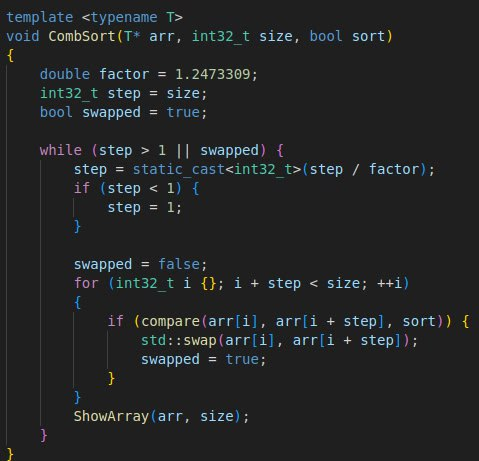
Bablsort

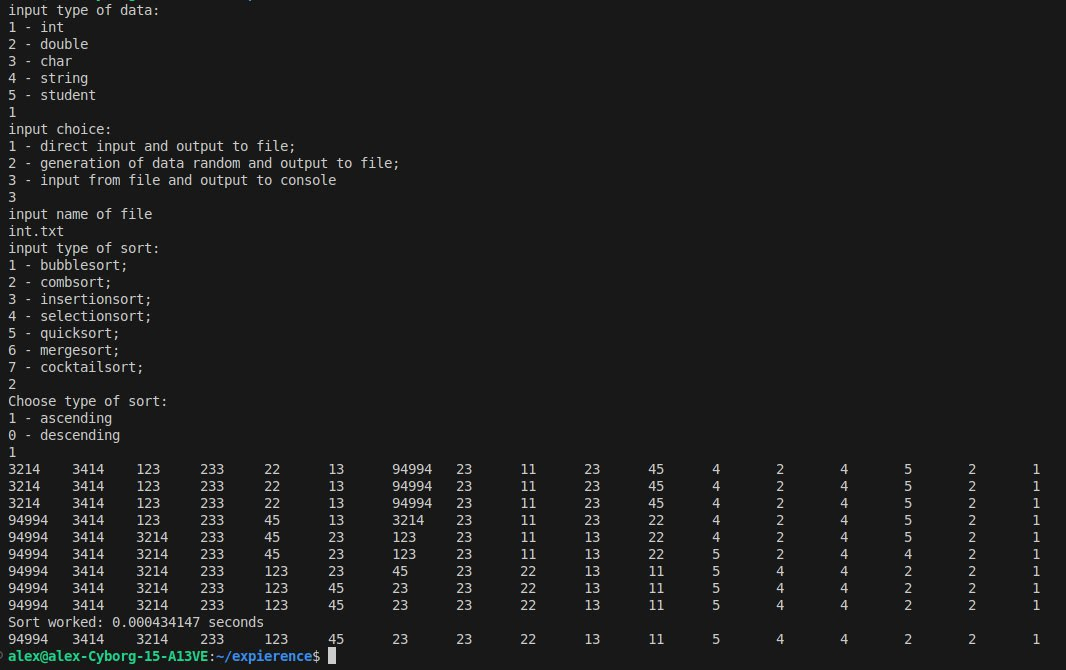




Алгоритм многократно проходит по массиву, последовательно сравнивая пары соседних элементов и меняя их местами, если они стоят в неправильном порядке, в результате чего самые "тяжелые" элементы постепенно "всплывают" в конец списка. Ключевое усовершенствование заключается в проверке: если за целый проход не произошло ни одного обмена, значит массив уже отсортирован, и алгоритм досрочно прекращает работу, чтобы не выполнять лишних действий. бабл

Combsort

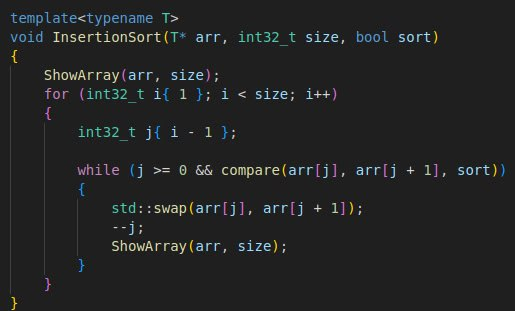


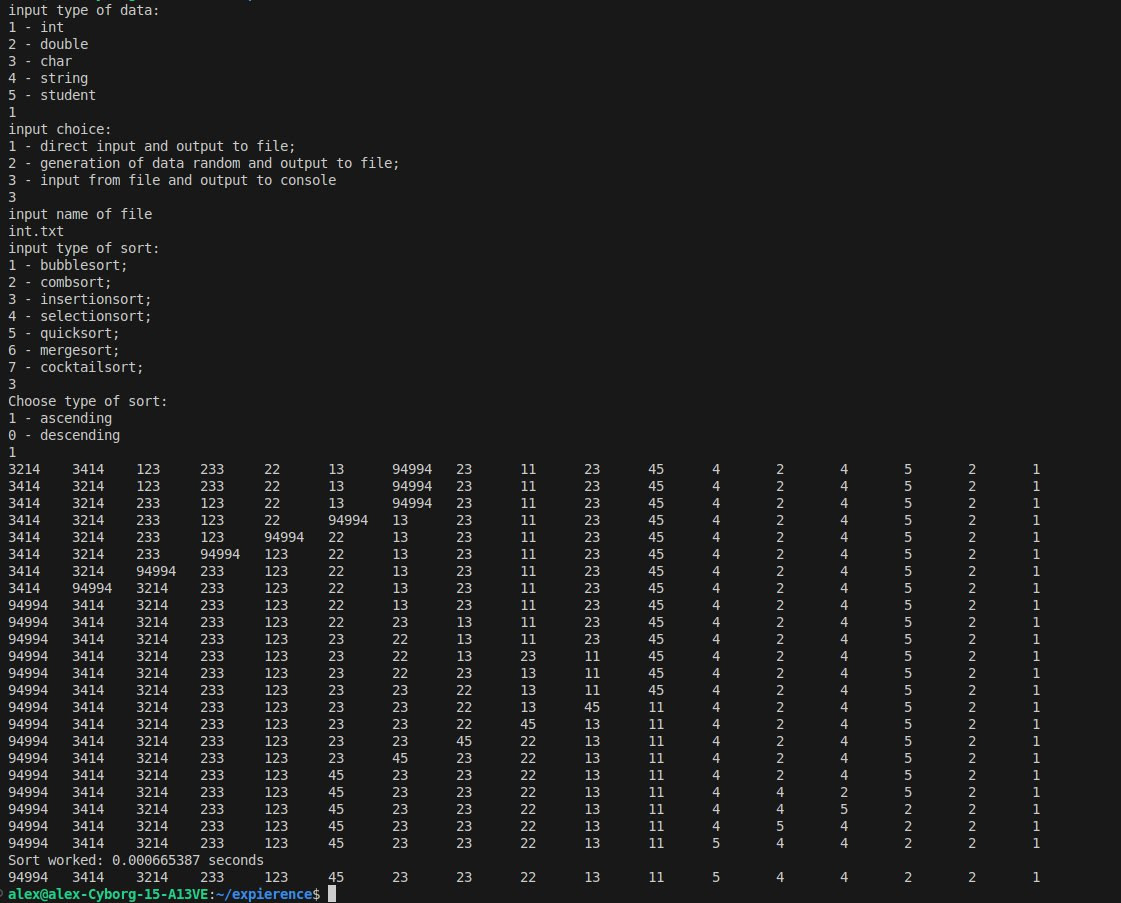


Сортировка расческой работает как улучшенный "пузырек", но вместо сравнения соседних элементов, она сравнивает элементы на определенном расстоянии друг от друга, называемом "шагом". Изначально шаг большой, что позволяет быстро перемещать значения из одного конца массива в другой, как бы "расчесывая" большие беспорядки. С каждым проходом шаг постепенно уменьшается, пока не станет равен 1, после чего алгоритм делает финальные проходы, как обычная пузырьковая сортировка.

Главное отличие от пузырьковой сортировки заключается в этом изменяемом шаге. Пузырьковая сортировка всегда сравнивает только соседние элементы (шаг равен 1), из-за чего она очень медленно справляется с "черепахами" — маленькими значениями в конце массива, которые могут сдвинуться лишь на одну позицию за проход. Сортировка расческой, благодаря большому начальному шагу, может перебросить такую "черепаху" через большую часть массива за один обмен, эффективно решая эту проблему и значительно ускоряя процесс сортировки для большинства наборов данных.

Insertionsort





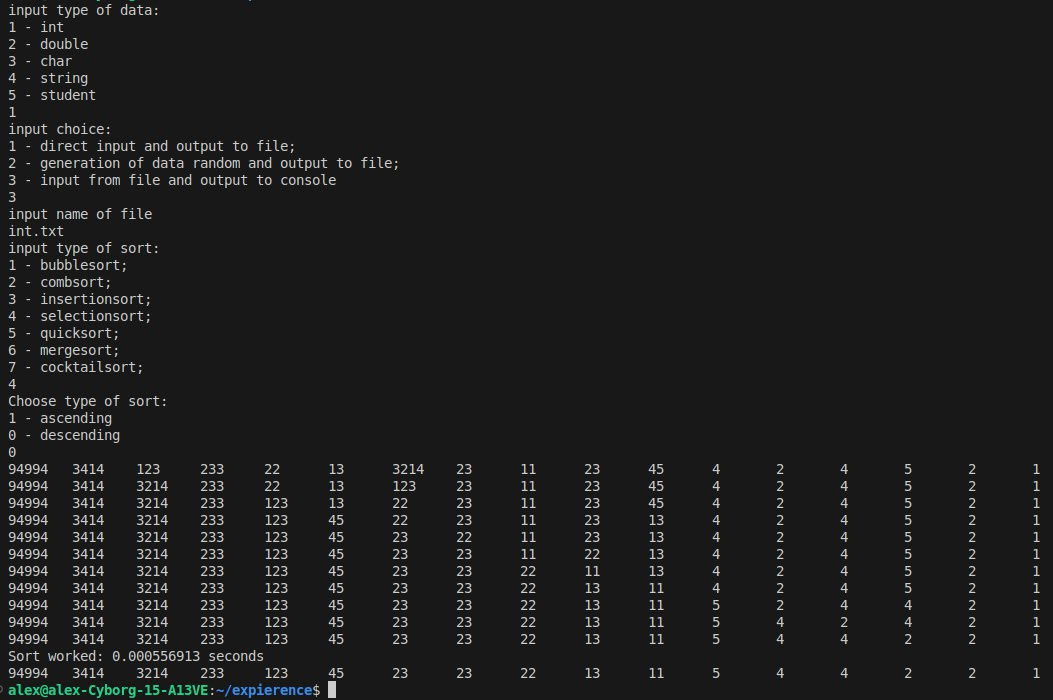
Сортировка вставками работает по принципу: Алгоритм мысленно делит массив на две части: отсортированную слева и неотсортированную справа. Он берет первый элемент из неотсортированной части и начинает двигаться по отсортированной части справа налево, ища для него подходящее место. При этом все элементы, которые больше взятого, сдвигаются на одну позицию вправо, освобождая место, куда и вставляется наш элемент. Этот процесс повторяется до тех пор, пока все элементы из правой части не будут по одному вставлены в левую.

В сравнении с пузырьковой сортировкой, сортировка вставками обычно работает быстрее. Пузырьковая сортировка на каждом проходе просто меняет местами соседние элементы, чтобы самый большой "всплыл" в конец, и ей часто приходится делать много лишних сравнений и обменов по всей длине массива. Сортировка вставками же более "умная": как только она находит правильное место для элемента в уже отсортированной части, она прекращает его двигать. Это особенно эффективно, если массив уже частично отсортирован — в таком случае работы у нее будет совсем немного.

Если сравнивать с сортировкой расческой, то вставки проигрывают на больших и хаотично перемешанных массивах. Расческа специально создана для того, чтобы быстро перемещать элементы на большие расстояния благодаря своему изменяемому шагу, эффективно устраняя "черепах" (маленькие значения в конце). Сортировка вставками, как и пузырек, работает "локально", сдвигая элементы только на одну позицию за раз. Поэтому, если маленький элемент находится в самом конце массива, вставкам придется проделать огромную работу, чтобы переместить его в начало, в то время как расческа справится с этой задачей гораздо быстрее.

Selectionsort





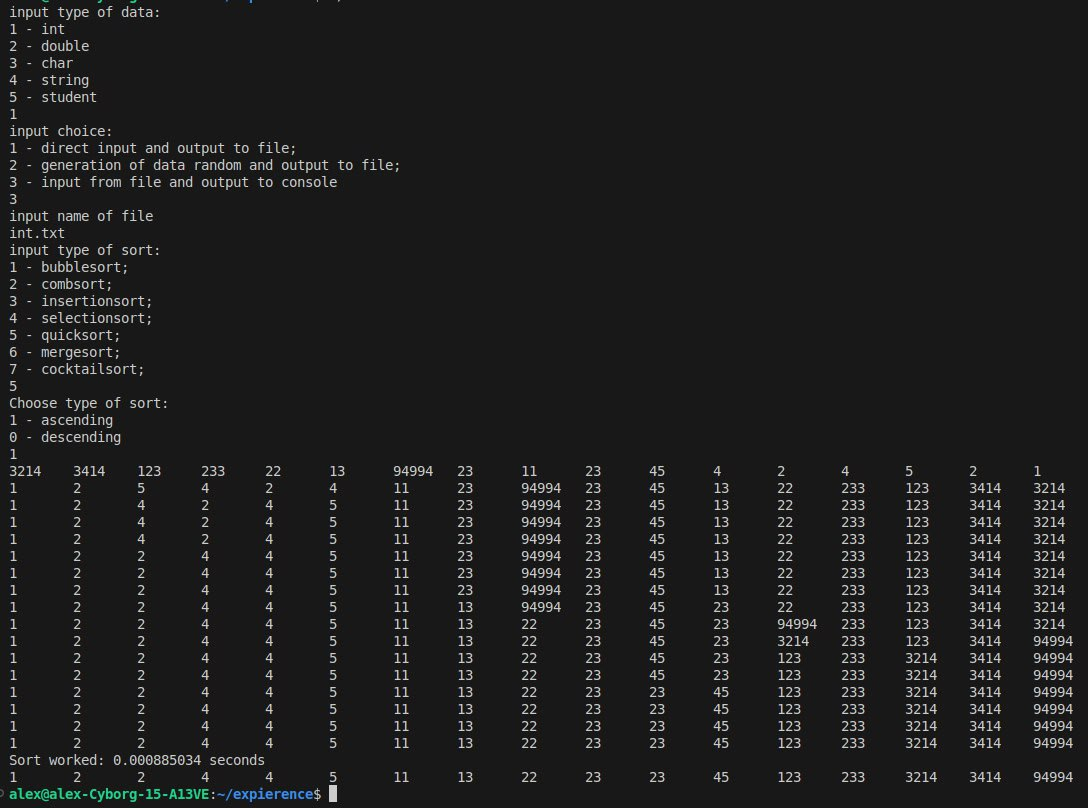
Сортировка выбором работает, многократно проходя по неотсортированной части массива, чтобы найти самый маленький элемент. Как только этот элемент найден, он меняется местами с первым элементом в этой неотсортированной части, тем самым расширяя отсортированную область на один элемент. Этот процесс поиска минимального элемента и его перемещения в отсортированную зону повторяется до тех пор, пока весь массив не будет отсортирован.

По сравнению с пузырьковой сортировкой, сортировка выбором гораздо экономнее относится к операции обмена. В то время как "пузырек" может менять элементы местами на каждом шагу, сортировка выбором делает только один обмен за целый проход, что делает ее эффективнее, если запись в память — медленная операция. Однако оба алгоритма одинаково медленны по общему времени выполнения на больших массивах, так как требуют квадратичного числа сравнений.

При сравнении с сортировкой вставками, сортировка выбором действует совершенно иначе. Сортировка выбором на каждом шаге просматривает всю оставшуюся часть массива, чтобы найти единственно верный (например, минимальный) элемент и делает всего один обмен, чтобы поставить его на свое законное место. Вставки же, наоборот, берут следующий элемент и начинают его "проталкивать" назад по уже отсортированной части, выполняя потенциально много сдвигов (обменов), пока не найдут для него подходящую позицию. Из-за этого сортировка выбором выигрывает, когда операция записи данных "дорогая", но проигрывает в "интеллекте": она не умеет адаптироваться и всегда выполняет одинаковый объем работы, в то время как сортировка вставками может быть очень быстрой на частично упорядоченных массивах.

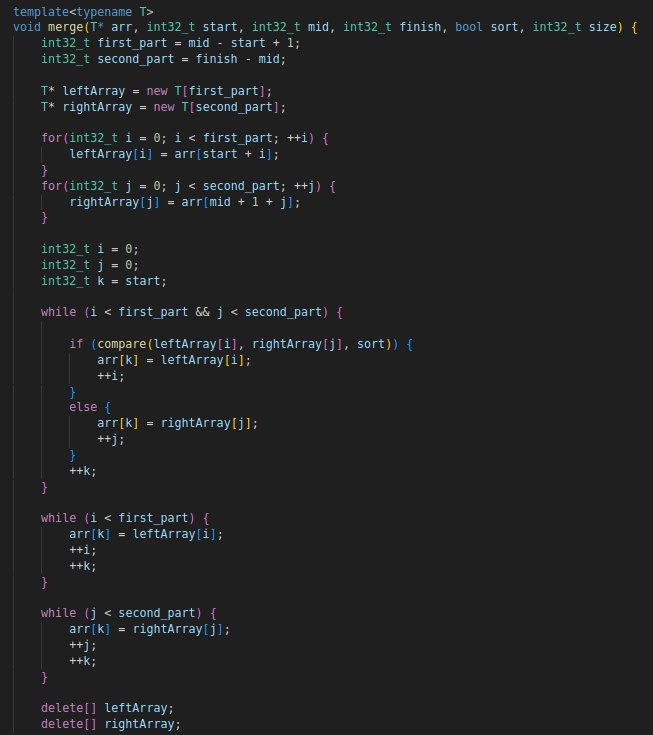
Quicksort

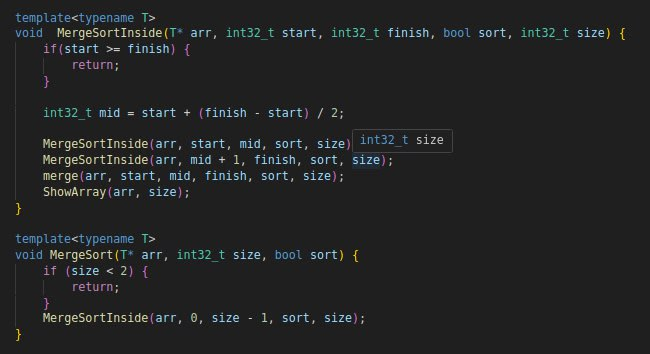


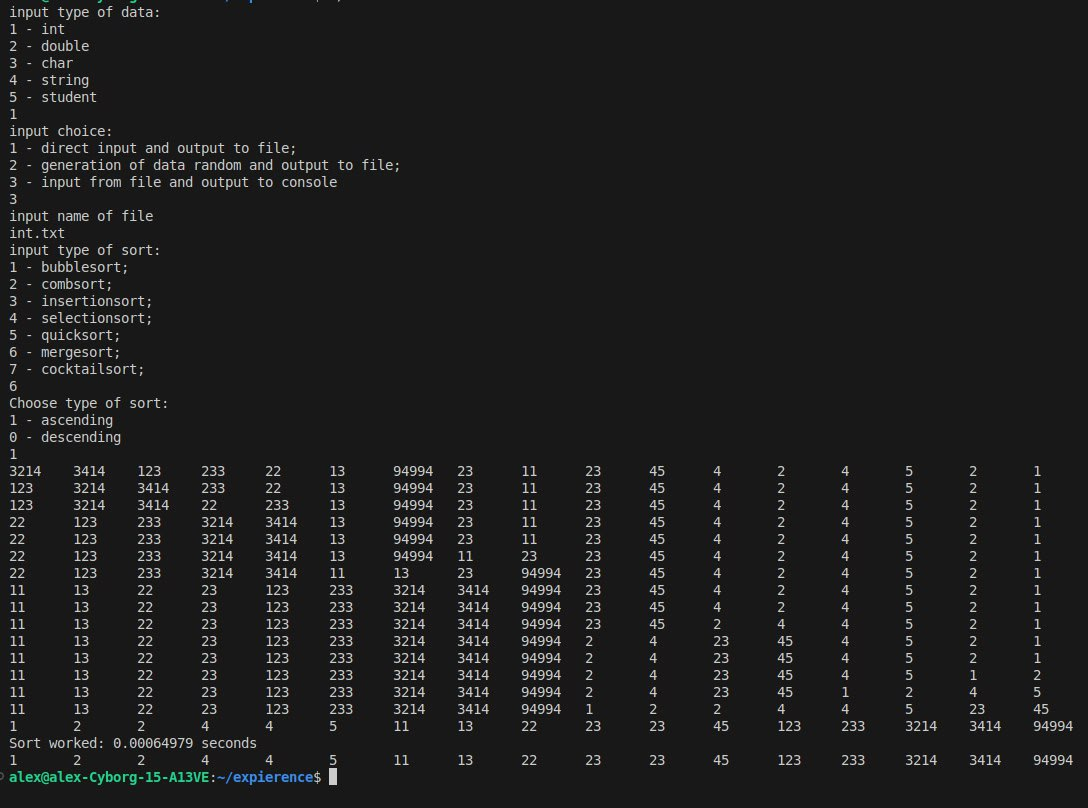


Реализация быстрой сортировки, использующая схему разделения Хоара, работает по принципу "разделяй и властвуй": она выбирает "опорное" значение из середины массива и с помощью двух указателей, движущихся с разных концов навстречу друг другу, быстро раскидывает все элементы. Левый указатель ищет элемент, который больше опорного, а правый — который меньше, после чего эти два "не на своем месте" элемента меняются местами. Этот процесс продолжается, пока указатели не встретятся, эффективно разделяя массив на две части — "меньшую" и "большую", — после чего алгоритм рекурсивно применяет ту же самую логику к каждой из этих двух частей, пока весь массив не будет отсортирован.

Mergesort

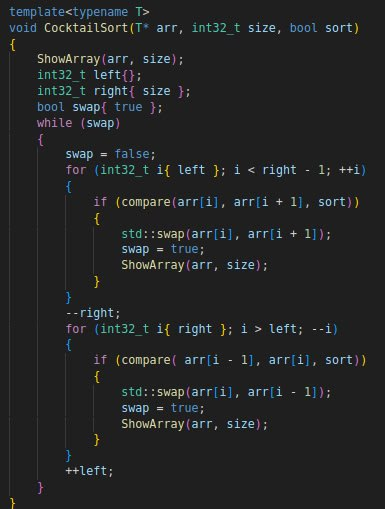


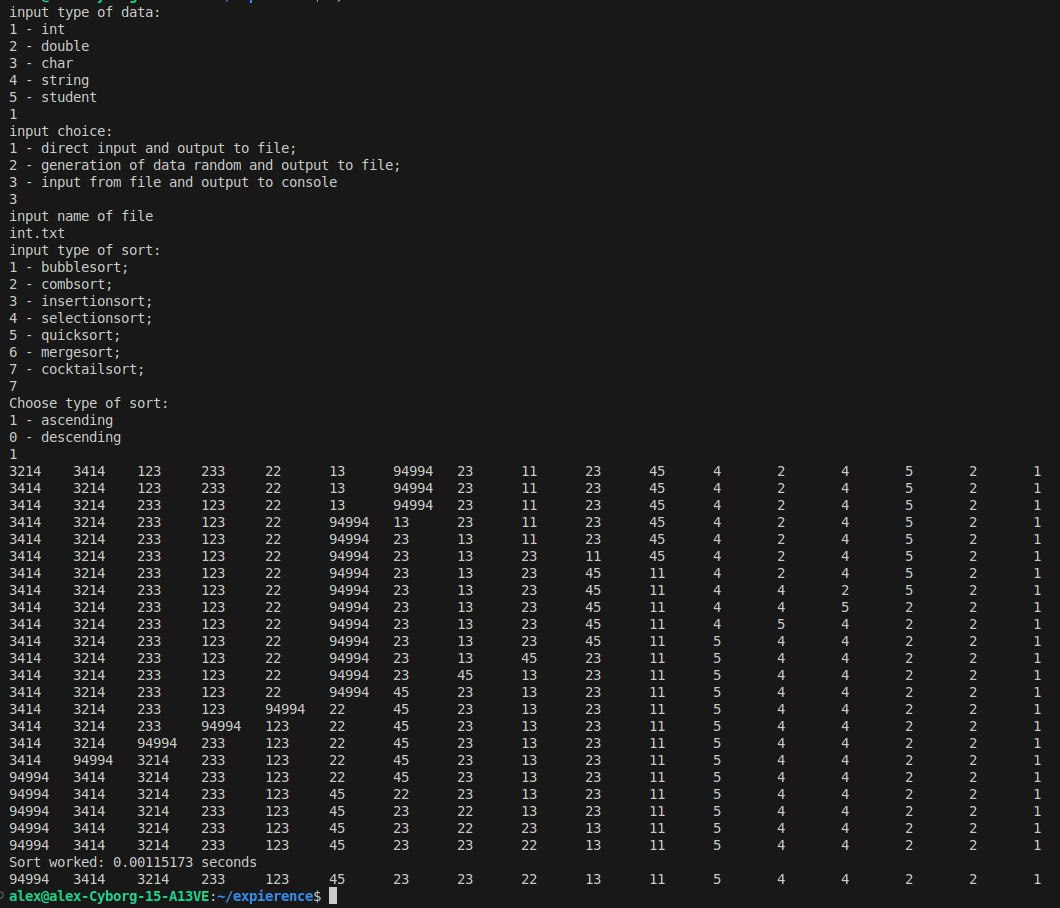




Сортировка слиянием работает по принципу "разделяй и властвуй": сначала она рекурсивно дробит массив пополам до тех пор, пока не останутся одиночные, уже отсортированные элементы. После этого начинается обратный процесс, где алгоритм последовательно "сливает" соседние отсортированные части: он берет два таких фрагмента и, поочередно сравнивая их первые элементы, копирует меньший из них в основной массив, пока не объединит их в один, более крупный отсортированный кусок. Этот процесс повторяется на всех уровнях, пока весь массив не станет единым отсортированным целым.

Coctailsort





Шейкерная сортировка, или Cocktail Sort, является двунаправленной версией пузырьковой сортировки, которая более эффективно решает проблему "черепах" (маленьких элементов в конце массива). Алгоритм сначала проходит по массиву слева направо, "всплывая" самый большой элемент в конец неотсортированной части, а затем сразу же меняет направление и проходит справа налево, "утапливая" самый маленький элемент в начало. С каждым таким двойным проходом отсортированные границы массива сужаются с обеих сторон, и алгоритм останавливается, как только полный проход "туда-обратно" не приводит ни к одному обмену, что означает полную отсортированность массива.

ЗАКЛЮЧЕНИЕ

Самой быстрой в большинстве случаев является Быстрая сортировка (QuickSort). Подход «разделяй и властвуй» очень эффективен и позволяет ей обгонять другие алгоритмы, особенно на больших объемах данных. Ее единственный минус — в очень редких, неудачных ситуациях она может сильно замедлиться.

Самой устойчивой и надежной является Сортировка слиянием (Merge Sort). «Устойчивость» означает, что если у вас есть два одинаковых элемента (например, две пятерки), то после сортировки они останутся в том же порядке, в каком были изначально. Это важно, когда вы сортируете не просто числа, а сложные данные. Кроме того, она всегда работает с предсказуемой скоростью и никогда не «проваливается», как это может случиться с быстрой сортировкой, что делает ее очень надежным выбором.

Самыми медленными являются Сортировка выбором и Пузырьковая сортировка (включая ее шейкерный вариант). Они очень неэффективны, потому что выполняют огромное количество лишней работы.