

Лабораторная работа 3

Сортировка

Цель работы

Изучить особенности работы со структурами, рекурсией и шаблонами в C++.

Стандарт языка

C++17 и новее.

Описание

Программа должна сортировать массивы различных типов данных, используя алгоритм быстрой сортировки. Должна поддерживаться возможность сортировки по возрастанию и убыванию.

Типы данных

Программа должна сортировать данные 3 типов:

1. int
2. float
3. phonebook

Пример записи структуры phonebook в файле:

<фамилия><пробел><имя><пробел><отчество><пробел><телефон>

где:

- <фамилия>, <имя>, <отчество> — строки не длиннее 20 символов, не содержат пробелов, табуляции, переводов строк или ноль-символов;
- <телефон> — целое неотрицательное число, меньше 10^{11} .

Сортировка структур проводится в следующем порядке (если совпадают первые элементы, то сравниваются следующие):

фамилия->имя->отчество->телефон.

Каждое поле `phonebook` должно храниться отдельным полем структуры/класса.

Числа сортируются в арифметическом порядке, однобайтовые строки символов сортируются по коду символа (регистрозависимо).

Реализация по файлам

Весь код программы должен быть реализован в нескольких файлах.

Файл	Что должно быть в файле (порядок условный)
<code>main.cpp</code>	<p>Подключение заголовочных файлов стандартной библиотеки, необходимых для этого файла</p> <p>Подключение своих заголовочных файлов</p> <p>[Optional] Свои функции, которые относятся к <code>main</code></p> <p><code>int main(аргументы)</code></p>
<code>quicksort.h</code>	<p>Подключение заголовочных файлов стандартной библиотеки, необходимых для этого файла</p> <p>Подключение своих заголовочных файлов</p> <p>Определение шаблона функции сортировки со следующими аргументами шаблона</p> <p><code>template <typename T, bool descending></code> <code>void quicksort(ваши_аргументы)</code></p>
<code>phonebook.h</code>	<p>[Optional] Подключение заголовочных файлов стандартной библиотеки, необходимых для этого файла</p> <p>[Optional] Подключение своих заголовочных файлов</p> <p>Объявление структуры/класса <code>phonebook</code></p>

	Сигнатуры функций/операторов для работы с phonebook
phonebook.cpp	[Optional] Подключение заголовочных файлов стандартной библиотеки, необходимых для этого файла Подключение своих заголовочных файлов Реализация функций/методов/операторов для работы с phonebook

Формат аргументов командной строки

Аргументы программе передаются через командную строку:

сpp1 <имя_входного_файла> <имя_выходного_файла>

Формат входного файла

Во входном файле в первой строке указан **тип данных**. Допустимые значения (без кавычек): “int”, “float”, “phonebook”.

Во второй строке указан **режим сортировки**. Допустимые значения: “ascending”, “descending”.

На третьей строке указано кол-во значений в файле.

После этого на каждой строке задан элемент указанного типа. По одному значению на строку, каждая строка завершается символом перевода строки.

Корректность входных данных гарантируется.

Пример входного файла:

phonebook	int
ascending	descending
2	2
aa bb cc 255	5
Aa bB cc 265	3

--	--

Формат выходного файла

Отсортированный массив. По одному значению на строку, каждая строка завершается символом перевода строки.

Пример выходного файла:

Aa bB cc 265	5
aa bb cc 255	3

Требования к программе

Программа должна:

1. быть написана на C++ по заданному стандарту;
2. выполнять поставленную в ТЗ задачу;
3. не использовать внешние библиотеки;
4. всегда корректно освобождать память;
5. всегда корректно закрывать файлы;
6. обрабатывать ошибки:
 - a. файл не открылся;
 - b. не удалось выделить память;
 - c. на вход передано неверное число аргументов командной строки.

В этих случаях необходимо выдавать сообщение об ошибке и корректно завершаться с ненулевым кодом возврата (см. "return_codes.h");

7. никогда ничего не писать **в поток вывода**;
8. выводить сообщения об ошибке в **поток вывода ошибок**.

Ограничения

1. Запрещено использование `exit(...)` в коде.
2. Запрещено создавать VLA-массивы (но можно VLA-указатели).
3. Ограничивается использование глобальных переменных (кроме констант) - необходимость их использования вы должны обосновать на защите. Ваш код должен быть максимально приспособлен к переносимости в другие проекты и/или использованию другими разработчиками.
4. Запрещается подключать системные библиотеки через `#include "..."`.
5. Запрещается использовать `setlocale(...)`. Учимся писать небольшие комментарии пользователю по-английски.
6. Запрещается использовать `system("pause")`.
7. Если вы создаёте свои макросы, то их название не должно быть `"DEBUG"`, `"_DEBUG"`, `"NDEBUG"` и прочие, определяемые компиляторами имена. Допустимы любые другие названия из заглавных букв и символов подчёркивания.

Комментарии по проверкам

Дабы избежать ситуации “Я же отправил что-то, почему мне не зафиксировал дедлайн?”, а на деле там открытие и закрытие файлов, то в этот раз есть ограничения, что должно быть отправлено на проверку (из минимума).

Отправка 1: Работающий на значениях типа `int` алгоритм (можно опустить шаблон или явно задать специализацию под `int` на этом этапе). Проверки на открытие входного файла, на выделение памяти под массив данных. Сортировка, вывод результата.

Отправка 2: Реализация алгоритма сортировки с шаблоном, должно работать на всех типах данных по заданию. Должны быть все проверки.

Отправка 3: Багфиксим реализации, добираем баллы.

Если во входном файле вы встречаете тип данных, с которым пока что не умеете работать, то нужно завершать программу с ошибкой NOT_IMPLEMENTED.

Полезные check-листы и ссылки:

- [Проверки](#)
- [Отправка на GitHub](#)
- [Про критерии оценивания](#)
- [Суперпопытка](#)

Попытка 1. Комментарии

[Дедлайн] Все, кто хоть что-то прислал к первой попытке, вне зависимости от качества посылки получают фиксацию дедлайна.

[Неработоспособный код] Зафиксированный дедлайн + 0 баллов за работоспособность = ваш код не собрался у проверяющего *или* ваш код не может отсортировать даже массив из `int`-ов *или* вы сдали совсем не то, что вас просили. Получить комментарий из-за чего именно можно в лс проверяющего. Частые ошибки: неправильное подключение заголовочных, некоторые заголовочные для функций, использующихся в коде, не подключены.

[Мало баллов] Если вы видите, что стоят оценки по небольшой части критериев, значит ваше решение почти неработоспособно (постоянно падает, не умеет работать с нашими входными файлами) и проверка проводилась по-минимуму.

[Работает только с `int` /`int+float`] Те работы, которые поддерживают лишь работу с `int` или `int+float`, были проверены не полностью.

[Code style] Критерий code style в этой работе складывается из 2 частей – соответствие .clang-format и пункту «Реализация по файлам» из ТЗ. Подробнее про последнее: в main не должно быть реализации quicksort – для этого у вас есть quicksort.h и при желании можете создать себе quicksort.cpp (создание доп. файлов с кодом не запрещено). Также в phonebook.h должны быть лишь объявление структуры/класса phonebook, а реализация всех функций и методов – в phonebook.cpp. Исключением являются функции/методы в 1 строку (вида return что-то), для них допустимо оставить реализацию в phonebook.h.

[Работа с памятью] Здесь работы можно поделить на 2 части: работа в стиле Си или в стиле C++. Неполный балл стоит в случаях:

Си: память не освобождается; память выделяется без проверки, что она выделилась правильно; память выделяется постоянными realloc.

C++: память не освобождается; память выделяется без проверки, что она выделилась правильно; проверка на то, выделилась правильно, сделана неправильно; заполнение массива (читай vector) происходит через постоянные push_back и подобное.

[Работа с файлами] Здесь ошибок существенно меньше, чем в предыдущем пункте. Основные ошибки – файл не закрывается, открывается не в том режиме.

[Обработка ошибок] Ошибка большинства - если память под массив не выделилась, то создается пустой файл. В остальном всё как в прошлых работах.

[Шаблоны] В ТЗ дан четкий **прототип шаблона** для функции сортировки. И те, кто его не придерживается или придерживается, но по факту сортирует в одну сторону, получают неполный балл по данному критерию.

[Тесты] Балл по тестам вычисляется как (сумма отношений кол-ва пройденных тестов в категории к общему числу тестов в категории) / 6 /

(коэффициент адекватности решения). Категории: { int_ascending, float_ascending, phonebook_ascending, int_descending, float_descending, phonebook_descending }. Коэффициент адекватности решения – то, что не позволяет вам просто взять своё решение с курса алгосов и получить полный балл. Подробнее об этом можно узнать на лекции.

И небольшое напоминание – ваша функция сортировки должна уметь сортировать поданный на вход массив данных в нужном направлении, а не «я буду сортировать в одном направлении, а потом просто при выводе в файл буду с конца печатать массив, если надо». За такие работы баллы, очевидно, частично не засчитываются. Подсказка: решение вида «отсортирую в одну сторону и если надо, то разверну массив» пройдет лишь часть тестов и упрется в time limit.