# Организационные вопросы

## Оценка лабораторных работ

Лабораторная работа оценивается по 2-балльной шкале (зачтено/не зачтено). Для получения положительной оценки достаточно выполнить работу с соблюдением минимальных требований. Выполнение дополнительных заданий не отменяет необходимость соблюдения минимальных требований.

### Минимальные требования

1. Код должен компилироваться без предупреждений при максимальном уровне предупреждений. Для компилятора GCC это набор флагов: -Wall -Wconversion -Wextra -Wpedantic. Для компилятора MSVC (Visual Studio) это флаг /W4. Для других компиляторов согласуйте настройки компиляции с преподавателем.

2. Нельзя использовать глобальные переменные (константы допустимы).

3. В коде должен использоваться только полноценный английский язык (транслит запрещен). Русский язык разрешено использовать только в комментариях.

4. Запрещается комментировать каждую строчку кода. Допустим один краткий комментарий на блок кода. Разрешается комментировать одиночную строчку кода, только если она действительно делает что-то неожидаемое и хитрое (но помните, хитрый код — плохой код!).

5. Весь код должен удовлетворять единому стилю программирования. Сам стиль можно выбирать по своему вкусу (см., например, [Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html), [GNU C Style](https://www.gnu.org/prep/standards/html_node/Writing-C.html) и другие). То есть запрещено, например, называть одну функцию в стиле [CamelCase](https://ru.wikipedia.org/w/index.php?title=CamelCase&stable=1), а другую — в стиле [snake\_case](https://ru.wikipedia.org/w/index.php?title=Snake_case&stable=1). Исключение допустимо только для названия функции main, которое всегда пишется в нижнем регистре. Данные требования предъявляются к любым именуемым сущностям в программе — к функциям, методам, классам, локальным переменным, параметрам функций и методов, названиям файлов и так далее.

6. В случае удалённого формата сдачи обучающийся предоставляет ссылку на git-репозиторий. В случае очного формата сдачи репозиторий может быть локальным (на компьютере в аудитории или ноутбуке обучающегося). В любом случае репозиторий не может содержать одинокий коммит с целой лабой, а должен показывать историю работы над лабораторной в виде серии коммитов с содержательными заголовками. В репозитории должен быть корректным образом настроен файл .gitignore (т. е. в репозитории должны находиться только файлы с исходным кодом и файлами проекта, никаких промежуточных и итоговых результатов компиляции в нём быть не должно).

7. В случае удалённого формата сдачи для защиты необходимо предоставить небольшой содержательный отчёт, подготовленный в Latex, LibreOffice, Microsoft Word или любой другой системе. Отчёт предоставляется в формате pdf (но быть готовым предоставить в исходном формате по просьбе преподавателя). В отчёте должны присутствовать: титульный лист, выданное согласно варианту задание, протокол тестирования, заключение по выполненной работе и дополнительные разделы, требуемые вашим преподавателем (если есть).

# Лабораторная работа № 2: Сортировки

Лабораторная работа состоит из 2 заданий:

## Задание 1.

Реализовать алгоритмы сортировки для массивов целых чисел согласно своему варианту. *Таблица распределения вариантов приведена в приложении 1*.

|  |  |  |
| --- | --- | --- |
| Группа 1 | Группа 2 | Группа 3 (по желанию) |
| 0. Сортировка пузырьком  1. Сортировка вставками  2. Сортировка выбором | 0. Сортировка Шелла  1. Шейкерная сортировка  2. Быстрая сортировка | 0. Сортировка расчёской  1. Сортировка естественным  двухпутевым слиянием  (naturial two-way merge sort)  2. Пирамидальная сортировка |

## Задание 2.

Посчитать число сравнений и число копирований объектов для массивов длины 1 000, 2 000, 3 000, …, 10 000, 25000, 50000, 100000:

а) в среднем (сгенерировать 100 случайных массивов и посчитать средние значения);

б) для полностью отсортированного массива;

в) для обратно отсортированного массива.

По полученным данным построить графики, сравнить экспериментальные результаты с теорией, сделать выводы.

## Минимальные требования:

1. Запрещено использование функций из заголовочных файлов <alogprithm> и <numeric>.
2. Функции сортировки должны возвращать структуру stats:

struct stats {

size\_t comparison\_count = 0;

size\_t copy\_count = 0;

};

1. Функции сортировки должны принимать std::vector<int> - сортируемый набор элементов.

## Дополнительные задания:

1. Реализуйте сертировку из 3 группы.
2. Дополнительно считайте время сортировок (см. <chrono>).
3. Измените функции так, чтобы они принимали в себа 2 итератора, указывающих на начало сортируемого дианазона, и конец сортируемого дианазона (подобно функциям из <algorithm>).
4. Сделайте ваши функции шаблонными. Продемонстрируйте работоспособность для std::string и вашего собственного класса с перегруженным оператором сравнения.

# Приложение 1. Распределение вариантов сортировок

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер в списке группы** | **Сортировка группы 1** | **Сортировка группы 2** | **Сортировка группы 3** |
| 1 | 0 | 0 | 0 |
| 2 | 2 | 2 | 1 |
| 3 | 0 | 2 | 2 |
| 4 | 1 | 1 | 0 |
| 5 | 1 | 2 | 1 |
| 6 | 2 | 0 | 2 |
| 7 | 0 | 1 | 0 |
| 8 | 0 | 1 | 1 |
| 9 | 0 | 2 | 2 |
| 10 | 1 | 1 | 0 |
| 11 | 2 | 0 | 1 |
| 12 | 0 | 0 | 2 |
| 13 | 2 | 1 | 0 |
| 14 | 2 | 2 | 1 |
| 15 | 1 | 0 | 2 |
| 16 | 1 | 2 | 0 |
| 17 | 1 | 1 | 1 |
| 18 | 1 | 2 | 2 |
| 19 | 0 | 1 | 0 |
| 20 | 2 | 2 | 1 |
| 21 | 1 | 1 | 2 |
| 22 | 1 | 2 | 0 |
| 23 | 1 | 0 | 1 |
| 24 | 0 | 2 | 2 |
| 25 | 0 | 1 | 0 |
| 26 | 1 | 2 | 1 |
| 27 | 1 | 1 | 2 |
| 28 | 1 | 1 | 0 |
| 29 | 2 | 2 | 1 |