Лабораторная работа № 2: «Классы»

Общие требования

1. Проект на git'е.
2. Наличие интерактивного диалогового интерфейса для проверки корректности разработанной программы.
3. Корректное завершение программы, как в случае штатного выхода, так и в случае невосстановимых ошибок (без утечек и без использования функций мгновенного завершения программы exit, abort, std::terminate и пр.).
4. Использование высокоуровневых подходов (например std::copy вместо memcpy, std::string вместо `char*`, исключений вместо кодов возврата и т.д.).
5. Логичная и удобная структура проекта, где каждая единица (файл/библиотека) обладает своей единой зоной ответственности (каждый класс в своих файлах .h и .cpp, диалоговые функции и main в своих).
6. Наличие средств автосборки проекта (CMake, qmake и прочие, работающие "поверх" Makefile).
7. Статический анализ кода, встроенный инструментарий в IDE (пр. VS2019: Analyze->Run Code Analysis, см. также Project -> Properties -> Configuration Properties -> Code Analysis -> Microsoft -> Active Rules) или внешние инструменты (Sonarqube + extensions, Clang Static Analyzer и д.р.) (обязательно знакомство с инструментом, исправление всех замечаний или обоснование в комментарии почему конкретное замечание исправить нельзя).
8. Динамический анализ на утечки памяти, встроенный инструментарий в IDE / библиотеки (Пр., VS2019) или внешние инструменты (valgrind, Deleaker и т.п.). Отсутствие утечек памяти и прочих замечаний анализатора.
9. Не "кривой", не избыточный, поддерживаемый и расширяемый код (разумная декомпозиция, DRY, корректное использование заголовочных файлов и т.п.).
10. Стандарт языка C++20 или C++23.
11. Сдача работ проводится только в интегрированной среде разработки (IDE) или редакторе с настроенным LSP.

Требования задачи

1. Разработка модульных тестов для классов, полное (не менее 80%) покрытие методов классов модульными тестами.
2. Обязательно использование фреймворков для тестирования решения (gtest, catch2 и пр.).
3. Структура решения:
   - проект со статически линкуемой библиотекой с классом,
   - проект консольного приложения для диалоговой отладки,
   - проект для модульного тестирования.
4. Использование контейнеров из STL, умных указателей и т.п. возможно только для "профессионалов" в C++, подавляющему большинству не рекомендуется (дождитесь задачи №3).
5. Документирование всех публичных методов класса с использованием doxygen. Документация метода должна включать в себя как минимум:
   - описание всех аргументов метода,
   - описание возвращаемого значения,
   - описание всех исключений, которые могут быть выброшены в этом методе (для каждого указать тип исключения и в каких случаях оно может возникнуть).
6. Требования к оформлению классов:

- корректность состояния класса (например, нельзя выделить массив размером N, а в capacity записать M (M!=N)),
- отсутствие избыточности (убрать лишние поля, внутренние методы убрать в private),
- наличие необходимых конструкторов и деструктора,
- корректность сигнатуры методов (см. стандартные сигнатуры операторов),
- сохранение семантики перегружаемых операторов (например, оператор + должен возвращать новое значение, не меняя имеющееся),
- сохранение семантики работы с потоками ввода/вывода для перегружаемых операторов ввода/вывода.

Порядок выполнения работы

Простой класс. Выполнить разработку простого класса и его методов, а также диалоговых функций для проверки класса.

Простой класс должен обладать как минимум следующими методами:
- пустой конструктор (явный или неявный);
- инициализирующие конструкторы, перечисленные в задании;
- методы получения значений атрибутов класса (геттеры);
- методы изменения значений атрибутов класса (сеттеры);
- методы ввода и вывода состояния класса в входной/выходной поток;
- прочие методы, перечисленные в задании.

Сложный класс. Выполнить разработку сложного класса и его методов, а также диалоговых функций для проверки класса. В качестве вектора использовать статический массив.

Сложный класс должен обладать как минимум следующими методами:
- конструктор по умолчанию (явный или неявный);
- инициализирующие конструкторы, перечисленные в задании;
- методы ввода и вывода состояния класса в входной/выходной поток;
- прочие методы, перечисленные в задании.

Перегрузка операторов. Модифицировать классы, перегрузив для них следующие операторы (внутри операторов целесообразно вызывать методы, реализованные в предыдущих пунктах):
- “=” для копирования экземпляра сложного класса;
- “>>” для ввода экземпляра простого класса из входного потока;
- “<<” для вывода экземпляра простого класса в выходной поток;
- “>>” для ввода экземпляра сложного класса из входного потока;
- “<<” для вывода экземпляра сложного класса в выходной поток;
- прочие операторы, определённые в задании (в скобках перед описанием метода).

Динамическая память. Модифицировать сложный класс, используя динамическую память для вектора (использование STL и умных указателей для вектора недопустимо в учебных целях).

Дополнить класс необходимыми методами:
- копирующий конструктор;
- перемещающий конструктор;
- перемещающий оператор “=”;
- деструктор.

Тестирование. В процессе выполнения каждого пункта задания необходимо реализовать тесты для разрабатываемых классов. Тесты должны покрывать все методы разработанных в

некотором пункте классов. При переходе между пунктами тесты необходимо обновлять в соответствии с внесёнными изменениями.

Документация. В процессе выполнения каждого пункта вести документацию разработанных классов (см. требования), особенно перед передачей кода другому студенту.

Примечание: крайне рекомендуется делать и сдавать все пункты строго по очереди. Однако, если вы уверены в своих силах, позволяется делать и сразу финальный вариант программы.

Дополнительные задачи

Прикладная программа. Передать разработанный класс другому студенту, получить взамен чужой класс и реализовать прикладную программу на основе полученного класса. Внесение существенных изменений в чужой класс не допускается. Возможно исправление чужих багов и внесение изменений в интерфейс класса, если вам не повезло с качеством полученного кода или нужно получать больше информации из класса.

Ревью. В процессе выполнения пунктов «Простой класс» - «Динамическая память», имеется возможность записаться на ревью, в процессе которого вы отдадите свой код случайному студенту и взамен получите код от 3-х других студентов. Ваша задача в процессе ревью проанализировать код ваших коллег и оставить к нему замечания, которые по вашему мнению помогут коллегам улучшить их работу. Взамен вы получите 3 ревью на свою работу, которые помогут вам улучшить её. Ревью будет выполняться в несколько волн, на каждую из которых вы можете отправить работу по любому из пунктов (отправка незаконченной части некоторого пункта нежелательна, но допустима; лучше отправляйте логически завершенный кусок кода).

Вариант № 13: «Логический Элемент»

Простой класс: Клемма – содержит информацию о типе клеммы (входная или выходная), количестве соединений (входная клемма может иметь не больше 1 соединения, а выходная клемма – не больше 3 соединений) и состоянии сигнала - значение уровня 0, 1 или X (неопределённое состояние). Входная клемма без единого соединения не может иметь состояние, отличное от X.

Методы простого класса (помимо общих):

- создание экземпляра класса с инициализацией типом клеммы, количеством соединений и состоянием сигнала;
- создание экземпляра класса с инициализацией только типом клеммы (без соединений);
- (++) увеличение количества соединений на 1;
- (–) уменьшение количества соединений на 1;
- (>>) соединение выходной клеммы и входной (состояние передаётся от выходной клеммы к входной клемме);
- разъединение выходной клеммы и входной (предполагается, что клеммы были соединены).

Сложный класс: Логический элемент – определяется множеством клемм.

Методы сложного класса (помимо общих):

- создание экземпляров класса с инициализацией количеством входных и выходных клемм логического элемента;
- создание экземпляров класса с инициализацией массивом описателей клемм заданной длины;
- переопределение состояний всех входных и выходных клемм посредством ввода их новых значений из входного потока;
- ([]) получение клеммы с заданным номером (возврат по ссылке);
- добавить входную или выходную клемму;
- (>>) соединение между собой выходной и входной клемм двух логических элементов (если имеется несколько клемм, то соединяются первые незанятые клеммы);
- форматирование логического элемента в виде псевдографической строки с символами ascii.