Лабораторная работа № 3: «Иерархия Классов»

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. На выбор предоставляется две категории вариантов: простые и сложные. Сложные варианты обладают дополнительным множителем для баллов, получаемых за пункты задания «UML», «Реализация», «Прикладная программа», «Тестирование» и «Многопоточность». Множитель, указанный в варианте, является ориентировочным и может меняться по усмотрению преподавателя. При выборе варианта также обратите внимание на то, что для простых вариантов отсутствуют дополнительные задания «динамическая загрузка» и «плагины».
2. Разработка модульных тестов для классов, удовлетворительное (не менее 50%) покрытие методов классов модульными тестами.
3. Использование фреймворков для тестирования решения (gtest, catch2 и пр.). Тестирование встроенными средствами языка запрещено.
4. Логичная структура решения, разделение на зоны ответственность (отдельные компоненты вынести в отдельные библиотеки), следование принципам SOLID.
5. Документирование всех публичных методов всех классов с использованием doxygen. Требования к документации аналогичны требованиям во 2-й ЛР..
6. Строгое следование схемам MVC или MVP при проектировании программы. То есть классы программы необходимо разделить на 3 категории (библиотеки):
   - Классы внутренней логики программы (model);

- классы отображения (view);
- классы управления/представления (controller/presenter);
- допускаются объединение view и controller/presenter в один компонент.

7. Динамический анализ (thread sanitizer и т.п.) многопоточной программы на отсутствие состязаний (race condidion). Наличие состязаний в финальной версии программы не допускается.
8. Основной язык программирования — C++. По согласованию с преподавателем, принимающим лабораторные работы, допускается использование Java или C#.

Порядок выполнения работы

UML. Выполнить проектирование диаграммы классов реализуемой программы в нотации UML (рекомендуется использовать специализированный редактор, например, modelio) и разработку соответствующих диаграмме прототипов классов (хедеров). Допустима генерация UML-диаграммы из кода или кода из UML-диаграммы, однако в любом случае диаграмма классов и их прототипы должны полностью соответствовать друг другу.

Реализация. Выполнить реализацию всех классов, отвечающих за логику программы. Для проверки реализованных классов использовать тесты или простую проверочную main функцию.

Шаблон. Выполнить реализацию указанного в задании контейнера в виде шаблонного класса. Разработанный шаблонный класс должен обладать основными методами (вставка, поиск, удаление и т.д.) и предоставлять полноценный интерфейс доступа при помощи итераторов. Класс итератора должен соответствовать выбранной категории (random access, bidirectional, forward или прочие). Выбор категории итератора необходимо обосновать. В учебных целях, при реализации своего шаблонного контейнера, запрещается использовать контейнеры STL. Использовать умные указатели можно.

Прикладная программа. Реализовать прикладную программу для работы с разработанными классами. Возможно выполнение пункта в 1 из 3 вариантов (каждый последующий оценивается в большее количество баллов): • диалоговая программа; • псевдографическая программа (обязательно интерактивное навигирование при помощи клавиш без нажатия Enter, например, с использованием библиотеки ncurses); • графическая программа.

Тестирование. Разработать тесты для классов логики. Написание unit тестов для интерфейса пользователя (контроллера/представления и отображения) не требуется.

Документация. Составить документацию для всех классов. Для простых вариантов вместо этого составить документацию для всех разработанных классов логики. Документировать классы пользовательского интерфейса не требуется.

Многопоточность. Модифицировать программу таким образом, чтобы указанный в задании алгоритм выполнятся параллельно в несколько потоков. Необходимо использовать предоставляемые языком примитивы синхронизации для избежания состязаний. Сравнить скорость выполнения одной и той же операции в однопоточном и многопоточном режимах в зависимости от объёма данных (построить график).

Примечание: Пункты задания от «UML» до «Прикладная программа» необходимо выполнять и сдавать строго в указанной последовательности. Приступать к выполнению следующего пункта до сдачи предыдущего крайне не рекомендуется, так как при обнаружении ошибок на более ранних этапах придётся переделывать всю программу целиком от первых и до последних этапов.

Дополнительные задачи

Динамическая загрузка. Обеспечить динамическую загрузку для наследников расширяемого (см. в задании) класса из динамических библиотек (.so/.dll). При запуске программа должна загружать все библиотеки, представленные в некотором каталоге, считывать их метаинформацию и использовать при дальнейшей работы. Перенести всех реализованных наследников расширяемого класса в динамические библиотеки в качестве примера.

Плагин. Обменяться своим кодом с другими студентами и выполнить реализацию своего плагина для кода другого студента. Плагин представляет из себя новог наследника расширяемого класса, собранного в виде динамической библиотеки. Модифицировать код другого студента не допускается.

Вариант № 11: «Файловая система»

Разработать приложение, позволяющее организовать работу с файлами некоторой файловой системы. Информация о некотором файле (обычном файле, каталоге или специальном файле) хранится в его описателе.

Описатель обычного файла содержит следующую информацию: дата и время создания файла, последней модификации; права доступа к файлу (можно/нельзя читать из файла, писать в файл, исполнять); размер файла в байтах и виртуальный адрес размещения файла на диске.

Описатель каталога содержит следующую информацию: права доступа к каталогу (можно/нельзя читать из каталога, писать в каталог); размер каталога в байтах и виртуальный адрес размещения каталога на диске; указатель на описание структуры каталога.

Описатель специального файла содержит следующую информацию: права доступа к файлу (можно/нельзя читать из файла, писать в файл); тип файла (байт или блок-ориентированный); размер файла в байтах и виртуальный адрес драйвера устройства.

В качестве виртуального адреса использовать адрес динамически выделенной памяти и отображать его в виде шестнадцатеричного числа.

Каждый файл имеет уникальную характеристику – идентификатор файла (последовательность латинских букв не более 24 символов). Информация обо всех файлах корневого каталога сведена в таблицу[1], каждый элемент которой содержит идентификатор файла и указатель на его описатель. Элементы таблицы упорядочены в алфавитном порядке.

Структура каталога в свою очередь может быть описана так же таблицей[2]. Таким образом, указатель на описатель структуры каталога в описателе каталога – указатель на таблицу файлов. Специальные файлы могут размещаться только в основном (корневом) каталоге.

Обеспечить выполнение следующих операций:

- Для таблицы:
  - включить новый элемент, не нарушая упорядоченности;
  - найти элемент по заданному идентификатору файла;
  - удалить элемент, заданный идентификатором файла;
  - показать содержимое таблицы.
- Для любого файла:
  - вывести информацию о файле;
  - определить (вернуть в качестве результата) тип файла;
  - получить права доступа к файлу; изменить права доступа;
  - получить размер файла; изменить размер файла (с изменением выделенной области).
- Для приложения:
  - включить новый файл в файловую систему с учётом маршрутного имени и прав доступа;
  - внести изменения в права доступа к файлу или размер файла;
  - удалить файл из файловой системы с учётом маршрутного имени и прав доступа (при удалении каталога удаляются все файлы и подкаталоги);
  - вывести информацию об именах файлов, их типе и правах доступа;
  - найти все файлы с заданным именем (относительным), используя класс-итератор[3].

---

[1] Шаблонный класс — упорядоченная таблица.

[2] Шаблонный класс — упорядоченная таблица.

[3]Указанную операцию реализовать в многопоточном режиме. Обработка очередного каталога выносится в новый поток в том случае, если на данный момент имеются свободные потоки (по количеству потоков процессора). В ином случае, очередной каталог обрабатывается в том же потоке, что и родительский.