Домашнее задание №5

Общие требования

1. Наличие интерактивного диалогового интерфейса для проверки корректности разработанной программы.
2. 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.).
3. Использование высокоуровневых подходов (например `std::copy` вместо `memcpy`, `std::string` вместо `char*`, исключений вместо кодов возврата и т.д.).
4. Использование стандартных библиотек и функций языка C++ предпочтительнее, чем использование библиотек и функций языка C (`std::copy` вместо `memcpy`, `std::abs` вместо `abs`, `cstring` вместо `string.h` 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. Предпочтительны инструменты, работающие «поверх» Makefile, например: CMake, qmake и др. Использование Makefile напрямую не желательно, но допустимо.
7. Статический анализ кода с использованием средств, встроенных в IDE (например, в VS2019: Analyze -> Run Code Analysis, см. также Project -> Properties -> Configuration Properties -> Code Analysis -> Microsoft -> Active Rules), или внешних инструментов (Sonarqube + extensions, Clang Static Analyzer и др.). Обязательным является знакомство с инструментом, исправление всех замечаний или обоснование, 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.

Требования задачи

1. Разработка модульных тестов для классов, удовлетворительное (не менее 50%) покрытие методов классов модульными тестами.
2. Использование фреймворков для тестирования решения (gtest, catch2 и пр.). Тестирование встроенными средствами языка запрещено.
3. Логичная структура решения, разделение на зоны ответственности — реализация отдельных компонентов в отдельных библиотеках, следование принципам SOLID.
4. Документирование всех публичных методов всех классов с использованием doxygen. Документация метода должна включать, как минимум, описание всех аргументов метода, описание возвращаемого значения и описание всех исключений, которые могут быть выброшены (для каждого из них необходимо указать тип и описать случаи, в которых оно может возникнуть).
5. Корректность состояния классов, отсутствие избыточности, наличие необходимых конструкторов и деструктора, корректность сигнатуры методов, сохранение семантики перегружаемых операторов и корректность их сигнатуры, сохранение семантики работы с потоками ввода/вывода для перегружаемых операторов сдвига.
6. Строгое следование схемам MVC или MVP при проектировании программы. Т.е. классы программы необходимо разделить на 3 категории (библиотеки):
   - классы внутренней логики программы (model);
   - классы отображения (view);
   - классы управления/представления (controller/presenter).

   Допускается 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.

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 (заголовочных файлов). Допусти-

ма 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 функцию `main`.

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. Возможно использование умных указателей.

Прикладная программа

Реализовать прикладную программу для работы с разработанными классами одним из следующих способов:

- 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 разработанных классов логики. Документировать классы пользовательского интерфейса не требуется.

Многопоточность

Модифицировать программу таким образом, чтобы указанный в задании алгоритм выполнялся параллельно в несколько потоков. Необхо-

димо 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).

Условие

Разработать приложение, позволяющее организовать работу по учёту студентов некоторого подразделения института. Информация об учащемся, хранится в некотором описателе учащегося.

Описатель **младшекурсника** содержит следующую информацию:
- фамилия и инициалы студента;
- «индекс группы»;
- номер профилирующей кафедры;
- оценки за прошедшую сессию (максимум — 5 чисел).

Описатель **старшекурсника** содержит следующую информацию:
- фамилия и инициалы студента;
- «индекс группы»;
- номер профилирующей кафедры;
- оценки за прошедшую сессию (максимум — 4 числа);
- направление темы УИР (учебно-исследовательской работы);
- место выполнения УИР;
- оценки руководителя и комиссии за УИР.

Описатель **выпускника** содержит следующую информацию:
- фамилия и инициалы студента;
- «индекс группы»;
- номер профилирующей кафедры;
- направление темы ДП (дипломного проекта);
- место выполнения ДП;
- оценки руководителя, рецензента и ГЭК (государственной экзаменационной комиссии) за ДП.

Каждый студент имеет уникальную характеристику — шифр студента (число).

Информация обо всех студентах сведена в таблицу[1], каждый элемент которой содержит шифр и указатель на его описатель.

Обеспечить выполнение следующих операций:

[1] Шаблонный класс — таблица, перемешанная сложением

- Для таблицы:
  - Включить новый элемент.
  - Найти элемент по заданному шифру.
  - Удалить элемент, заданный шифром.
  - Показать содержимое таблицы.
- Для любого студента:
  - Вывести информацию о студенте.
  - Получить (вернуть в качестве результата) категорию студента.
  - Получить (вернуть в качестве результата) информацию об оценках в соответствии с категорией студента; записать оценки соответствующего семестра.
  - Изменить индекс учебной группы.
  - Перевести студента в другую категорию учащихся.
- Для старшекурсников и выпускников:
  - Получить (вернуть в качестве результата) информацию о теме индивидуальной работы (УИР или ДП) студента; изменить информацию о теме индивидуальной работы.
  - Получить информацию о месте выполнения работы; изменить информацию о месте выполнения работы (УИР или ДП).
- Для приложения:
  - Оформить поступление студента в институт;
  - Оформить перевод студента на новый семестр (добавление новой записи с возможным изменением статуса);
  - Отчислить студента из института (исключение записи из таблицы);
  - Получить среднюю оценку студентов по группам, используя классс-итератор[2].

[2]Указанную операцию реализовать в многопоточном режиме (каждая группа студентов должна обрабатываться в отдельном потоке)