Задание 2 для первого семестра 2 курса

Требуется разработать и реализовать параметризованный класс списочного типа. Класс должен содержать конструктор без параметров, конструктор копирования, другие конструкторы по смыслу решаемых задач.

Набор методов класса должен соответствовать идеологии работы с соответствующей схемой хранения данных, т.е. обеспечивать требуемую дисциплину доступа к данным (возможно, с некоторыми модификациями) и также иметь итератор как средство перебора элементов структуры и идентификации положения отдельных элементов.

Структура, задающая узел списка, реализуется как внутренний класс этого списка, члены этой структуры могут быть public. Все объекты самого списка помещаются в private секцию, а в public секции описываются только интерфейсные методы, определящие дисциплину работы с элементами данных.

Итераторы реализуются двумя способами (в разных вариантах заданий) — как внутренние классы и как внешние по отношению к основному классу. Должен быть реализован по крайней мере один тип итератора — неконстантный. Другие типы (обратный, константный и т.д.) — по мере необходимости.

Для класса должен быть переопределен оператор вывода <<, который в наглядной форме изображает состояние данного класса.

Для контроля за некорректными ситуациями следует использовать механизм исключений и в некоторых случаях логические возвращаемые значения.

Для класса должен быть реализован оператор присваивания и также переопределены некоторые операции, например, сложение как объединение множеств элементов.

Как правило, для класса надо будет реализовать процедуры сортировки содержимого по некоторым заданным критериям и поиска требуемого элемента (возвращается итератор на найденный элемент).

После реализации собственного списочного класса, его работу надо сравнить с STL реализациями списочных контейнеров (std::list и/или std::forward_list или других подобных). Это означает, что надо придумать некоторую "задачу" для тестирования своей реализации и также решить эту задачу с использованием классов STL.

Подобная тестирующая задача может, например, заключаться в многократном добавлении и удалении элементов в списочную структуру, присваивании списочных структур друг в друга по некоторым правилам, автоматической проверке правильности полученного финального состочния контейнера. Например, заполнить список последовательными значениями $1, \ldots, n$, проходом по списку удалить все нечетные значения, а четные поделить пополам, выполнить присваивание этого списка другому списку, переставить элементы списка в обратном порядке, проверить, что получившийся список содержит последовательные числа $n/2, \ldots, 1$ а исходный содержит числа $1, \ldots, n/2$.

Требования к формулировкам и реализациям конкретного задания обсуждаются в рабочем порядке.

Варианты структур данных

A. Динамический массив-аллокатор (подробности в лекциях). Реализация на основе списка блоков фиксированного размера. Операции захвата элеента и освобождения элемента.

B1. Однонаправленный список. Внешний итератор. Доступ, вставка, удаление по позиции итератора. Добавление в начало и конец.

B2. Однонаправленный список. Внутренний итератор. Доступ, вставка, удаление по позиции итератора. Добавление в начало и конец.

C1. Двунаправленный список. Внешний итератор. Доступ, вставка, удаление по позиции итератора. Добавление и удаление в начале и в конце.

C2. Двунаправленный список. Внутренний итератор. Доступ, вставка, удаление по позиции итератора. Добавление и удаление в начале и в конце.

D1. Кольцевой однонаправленный список. Внешний итератор. Доступ, вставка, удаление по позиции итератора.

D2. Кольцевой однонаправленный список. Внутренний итератор. Доступ, вставка, удаление по позиции итератора.

E1. Кольцевой двунаправленный список. Внешний итератор. Доступ, вставка, удаление по позиции итератора.

E2. Кольцевой двунаправленный список. Внутренний итератор. Доступ, вставка, удаление по позиции итератора.

F1. Очередь на базе списка с константным итератором просмотра очереди (внешний). Доступ, добавление, удаление по правилам очереди. Двунаправленный итератор для доступа на чтение зачений по всем элементам.

F2. Очередь на базе списка с константным итератором просмотра очереди (внутренний). Доступ, добавление, удаление по правилам очереди. Двунаправленный итератор для доступа на чтение зачений по всем элементам.

G1. Дек на базе кольцевого списка. Итератор перемещения "окна" (внешний). Итератор показывает на два соседних элемента в кольце. Эти элемента и есть head и tail. Доступ, добавление, удаление по правилам дека для текущего положения окна. Перемещение итератора — переход к соседней паре.

G2. Дек на базе кольцевого списка. Итератор перемещения "окна" (внутренний). Итератор показывает на два соседних элемента в кольце. Эти элемента и есть head и tail. Доступ, добавление, удаление по правилам дека для текущего положения окна. Перемещение итератора —- переход к соседней паре.

Типы объектов

Релизация тестируется на нескольких различных типов хранимых объектов — "простых" и "сложных". Простые объекты — это примитивные числовые типы. Более сложные — структурные типы вроде вектора или числовых классов из первого задания. Наконец, реализация должна работать для "рекурсивного" типа, т.е. стек стеков целых чисел, список списков векторов и т.п. (например, Stack<Stack<int> > и т.п.).

График выполнения задания

Работа над заданием предполагает еженедельный контроль по следущему графику (естественно, его можно выполнять с опережением)

неделя 1: разработка и согласование описания (интерфейса) класса. Нужно представить описание класса с указанием прототипов требуемых методов (функций) и внутренних объектов класса.

неделя 2: реализация некоторых методов и первичный тест на простых типах данных, проверка работы механизма исключений Нужно представить тест с проверкой работы отдельных функций класса: по принципу "операция – распечатка состояния". В частности, должна быть готова функция распечатки (оператор «). Проверка на разных простых типах. Проверка на некорректных обращениях.

неделя 3. реализация более полного набора методов, тест совместной работы этих методов, тест на сложных типах данных. Тест с "нагрузкой" - выполняются массовые операции с автоматической проверкой результата. Например, добавили 1000000 последовательных целых чисел. Прошлись итератором и удалили четные. Прошлись итератором и проверили, что осталисть только нечетные и при этом ни одно нечетное число не пропало. Другие подобные тесты, смысл которых в многократном выполнении операций в середине и по краям списка. Присваивание списков друг другу. Некоторые ошибки (по памяти) могут не проявиться в однократной операции, а массовая проверка их обнаружит. Тест на "рекурсивных" данных типа список списков векторов :))) Т.е. элементами списка являются тоже списками. Тестирование правильности копирования и доступа. При доступе к элементу мы также получаем возможность работать с этим элементом как с контейнером. Как будет работать процедура распечатки?

неделя 4: Сравнение с STL библиотекой. Получить из данного начального состояния списка другое состояние, определяемое некоторыми правилами, с помошью ваших реализаций и с помощью классов библиотеки STL. Постановка задачи обсуждается индивидуально в рабочем порядке.

Таким образом, отчетность по этому заданию будет содержать 4 контрольные точки, соответственно данным этапам. В абсолютных датах контрольный срок сдачи задания — начало ноября.