Практическое задание №1 (весна-2018)

Задание состоит из последовательности задач. Оно посвящено реализации концепции итерируемости. Эта концепция позволяет абстрагироваться от реализации итерирования по содержимому объектов, по вычисляемым последовательностям и т.п.

Задача 1. Реализуйте абстрактный класс-шаблон Итерируемое с чисто виртуальными функциями begin() и end() без аргументов, возвращающими итераторы ввода (input iterators [1]). У каждого метода должен быть const-аналог (подумайте, каким должен быть их возвращаемый тип). Тем самым, любой класс, реализующий интерфейс Итерируемое, можно будет использовать в цикле for в стиле c++11 (range-for [2]). Обратите внимание, что разные наследники класса-шаблона Итерируемое могут иметь разные типы итераторов.

[1] input iterator – http://en.cppreference.com/w/cpp/concept/InputIterator

[2] range-for – http://en.cppreference.com/w/cpp/language/range-for

Задача 2. Реализуйте несколько классов итерируемых последовательностей (они должны реализовать интерфейс Итерируемое, то есть быть неабстрактными наследниками этого класса). Вам надо реализовать один пункт списка 1 (номер пункта – сумма цифр студенческого билета % 5), один пункт списка 2 (номер пункта – сумма цифр даты рождения % 5) и один пункт списка 3 (номер пункта – сумма оценок в зачетной книжке % 5), которые приведены ниже. При итерировании состояние итерируемой последовательности не должно меняться.

Список 1 (итерируемая последовательность содержит в памяти целиком все данные, по которым осуществляется итерирование, предусмотрите способ добавления данных в последовательность):

0) Журнал – это последовательность записей журнала, каждая запись состоит из метки времени и строки (сообщения). Итерирование по журналу – это итерирование по последовательности записей журнала.

1) Таблица в базе данных – это множество кортежей (пар атрибут – значение), причем у всех кортежей одной таблицы множество атрибутов совпадает. Итерирование по таблице – это итерирование по кортежам. (конкретный набор атрибутов и их типы вы можете выбрать самостоятельно)

2) Словарь машинного перевода – это отображение слов одного языка во множество слов другого языка. Итерирование по словарю – это итерирование по парам из слова одного языка и соответствующего ему множества слов другого языка.

3) Карта – это набор регионов. Каждый регион содержит набор координат опорных точек и название (строку). Итерирование по карте – это итерирование по регионам.

4) Вложения электронного письма в формате MIME – это последовательность записей. Каждая запись содержит набор пар из названия поля (строка) и его значения (тоже строка). Итерирование по вложениям электронного письма – это итерирование по этой последовательности записей.

Список 2 (итерируемая последовательность не содержит все значения, возвращаемые итератором, эти значения должны вычисляться итератором по мере необходимости):

0) последовательность чисел 1, N, 2, N-1, и т.д. пока первое меньше второго (число N задается при инициализации объекта последовательности)

1) последовательность битов заданного 32-битного числа N (число N задается при инициализации объекта последовательности)

2) последовательность чисел 1, 1, 2, 1, 2, 3, … , 1, 2, …, N (число N задается при инициализации объекта последовательности)

3) последовательность чисел Фибоначчи в порядке возрастания (до достижения INT_MAX)

4) последовательность делителей заданного числа N в порядке возрастания (число N задается при инициализации объекта последовательности)

Список 3 (итерируемая последовательность – это содержимое файла; нельзя предполагать, что его содержимое целиком уместится в памяти; при создании последовательности нужно указать имя файла; при невозможности открыть файл или выполнить файловую операцию должно генерироваться исключение std::runtime_error или иное подобное; при обходе файла можно предполагать, что файл не меняется):

0) последовательность строк текстового файла (до конца файла)

1) последовательность слов текстового файла (до конца файла)

2) последовательность символов текстового файла (до конца файла)

3) последовательность целых чисел бинарного файла из чисел (до конца файла)

4) последовательность целых чисел текстового файла (до первой ошибки чтения или конца файла)

Задача 3. Напишите функцию main(), в которой создайте объекты всех классов итерируемых последовательностей, реализованных в задаче 2. Напишите цикл for в стиле c++11 по каждой из этих последовательностей, которая пуста, и распечатайте содержимое последовательностей (тем самым вы проверите правильность работы для пустой последовательности). Наполните последовательность из списка 1. Напишите еще один цикл for в стиле c++11 по последовательности из списка 1 и распечатайте содержимое последовательности (тем самым вы проверите правильность работы для непустой последовательности).

Задача 4. Добавьте в функцию main() использование как минимум одной функции из <algorithm>, которая обработает каждую из реализованных последовательностей.

Задача 5. Напишите функцию, принимающую константную ссылку на любую итерируемую последовательность и распечатывающую последовательность поэлементно. Вызовите эту функцию из функции main().

Задача 6. Добавьте в интерфейс Итерируемое чисто виртуальную функцию-шаблон filter() и реализуйте ее во всех наследниках этого интерфейса. У функции один параметр – предикат от элемента последовательности. Функция возвращает итератор, который пропускает все элементы последовательности, для которых предикат возвращает ложь. Вызовите эту функцию из функции main().

Требования к коду

Код должен быть type-safety (безопасный в смысле типов). В частности, это означает, что любое использование приведений типов (в том числе, функций ***_cast) разрешается только в тех случаях, когда без них написать код невозможно. Использование friend подпадает под те же условия.

Код должен быть resource-safety (безопасный в смысле использования ресурсов). Ненужные ресурсы (области динамической памяти, файловые дескрипторы, записи таблицы процессов, семафоры и т.п.) должны вовремя отдаваться системе и не накапливаться, если в этом нет необходимости.

Код должен быть exception-safety (безопасный в смысле исключений). Возникающие исключения не должны нарушать инвариантов типов данных.

Обратите внимание на стиль оформления кода: https://ejudge.ru/study/3sem/style.shtml .

Можно предполагать, что программа будет выполняться на платформе Linux x86_64.

Для выполнения задания не требуются какие-либо библиотеки кроме STL. Если вам требуются иные библиотеки, согласуйте их использование с преподавателями.

Все функции в классе, автоматически генерируемые компилятором, нарушающие логику класса, должны быть переопределены или автоматическая генерация должна быть запрещена (=delete).

Везде при необходимости можно определять дополнительные функции и классы.

Об итераторах и итерируемых последовательностях

Последовательность items будем называть итерируемой, если это объект и у него есть методы begin() и end(), возвращающие итераторы такие, что возможен следующий цикл for для обхода последовательности:

for (auto item: items) { ... }

Над итераторами должны быть определены операции ++, *, == и !=. Метод begin() возвращает итератор начала последовательности, метод end() – конца последовательности (если быть точнее, «за концом» последовательности). Операция ++ продвигает итератор на следующий элемент последовательности, если он существует, иначе не меняет его. Операция * возвращает элемент, на который указывает итератор, если он не находится за концом последовательности. Операции == и != сравнивают итераторы. Дойдя до конца, итератор становится равным тому, что равно результату end(). Не дойдя до конца, итератор никогда не возвращает то, что равно результату end().