Пояснения к вариантам задания 2

Ниже приводится неформальное описание возможного интерфейса классов.

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

Описание будет позже.

B1, B2 Однонаправленный список.

— конструктор без параметров
— конструктор копирования
— деструктор
— оператор присваивания
— вставить элемент в начало
— добавить элемент в конец
— вставить элемент по позиции итератора
— извлечь элемент по позиции итератора
— удалить элемент по позиции итератора
— доступ к значению элемента по позиции итератора
— — константный
— — не константный
— поиск элемента по значению — возвращает итератор на позицию
— сортировка списка в диапазоне позиций от двух итераторов

Итератор.

— begin(), end()
— перейти к следующему ++
— значение элемента *
— операторы =, ==, !=, <

Операции добавления и удаления по позиции итератора предполагают получение новой текущей позиции в виде либо нового итератора, либо изменения значения данного итератора. Например, после удаления элемента по позиции итератора, итератор станет указывать на элемент, следуюший за удаленным. Другой вариант — удаляется элемент следующий за позицией итератора, а сам итератор остается на том же месте. То же самое при вставке — где будет итератор — на старом элементе или на новом, только что вставленном.

C1, C2 Двунаправленный список.

Здесь все аналогично, только есть два направленя перемещения по списку.

— конструктор без параметров
— конструктор копирования
— деструктор
— оператор присваивания
— добавть элемент в начало / конец
— вставить элемент по позиции итератора
— извлечь элемент по позиции итератора
— удалить элемент по позиции итератора
— доступ к значению элемента по позиции итератора
— — константный
— — не константный
— поиск элемента по значению — возвращает итератор на позицию
— сортировка списка в диапазоне позиций от двух итераторов

Итератор.

— begin(), end(), rbegin(), rend()
— перейти к следующему ++, предыдущему - -
— значение элемента *
— операторы =, ==, !=, <

Здесь вставка и удаление могут рассматриваться в трех вариантах: по позиции итератора, по позиции до или после итератора.

D1, D2 Кольцевой однонаправленный список.

Идейно как однонаправленный список, но фактически кольцо с одним направлением, т.е. нет фиксированного начала или конца. Можно в любой момент назначить "началом" любой конкретный элемент списка.

— конструктор без параметров
— конструктор копирования
— деструктор
— оператор присваивания
— set_begin — назначить начало по позиции итератора
— вставить элемент по позиции итератора
— извлечь элемент по позиции итератора
— удалить элемент по позиции итератора
— доступ к значению элемента по позиции итератора
— — константный
— — не константный
— поиск элемента по значению — возвращает итератор на позицию
— сортировка списка в диапазоне позиций от двух итераторов

Итератор.

— begin(), end()
— перейти к следующему ++
— значение элемента *
— операторы =, ==, !=, <

E1, E2 Кольцевой двунаправленный список.

Аналогично однонаправленному кольцу, только можем двигаться в обоих направлениях.

— конструктор без параметров
— конструктор копирования
— деструктор
— оператор присваивания
— set_begin — назначить начало по позиции итератора
— вставить элемент по позиции итератора
— извлечь элемент по позиции итератора
— удалить элемент по позиции итератора
— доступ к значению элемента по позиции итератора
— — константный
— — не константный
— поиск элемента по значению — возвращает итератор на позицию
— сортировка списка в диапазоне позиций от двух итераторов

Итератор.

— begin(), end(), rbegin(), rend()
— перейти к следующему ++ или предыдущему - -
— значение элемента *
— операторы =, ==, !=, <

Также есть разнообразие в трактовке добавления и удаления как в двунаправденном списке.

F1, F2 Очередь на базе списка с константным двунаправленным итератором просмотра очереди.

В очереди у нас формально есть доступ только головному элементу. Но можно дать право просматривать очередь без внесения в нее изменений, т.е. через константный итератор, который позволяет только полчать значения элементов, но не изменять их.

— конструктор без параметров
— конструктор копирования
— деструктор
— оператор присваивания
— добавить элемент в конец
— извлечь голову очереди
— удалить голову очереди
— доступ к значению головы очереди
— — константный

— — не константный

Итератор.
— begin(), end()
— перейти к следующему ++
— значение элемента *
— операторы =, ==, !=, <

В системах обслуживания часто используются так называемые очереди с приоритетами. Это подразумевает, что элементы очереди имеют некоторый параметр “приоритет”, который может меняться при нахождении элемента в очереди. При повышении приоритета элемент может извлекаться из середины низкоприоритетной очереди и добавляться в высокоприоритетную очередь. Такм образом, мы можем добавить
— извлечь элемент из очереди по позиции итератора

Итератор
— получить ссылку на приоритет элемента (например, !)