Лабораторная работа №2

по курсу «Языки программирования и методы программирования» (информатика, 3 семестр)

Техническое задание

**Используемые термины и сокращения**

ТД Тип данных

|  |  |
| --- | --- |
| АТД | Абстрактный тип данных |
| BД | B дерево |
| АВЛД | AVL дерево |

# Постановка задачи

Написать на языке С++ АТД «Словарь», взяв за основу BД и АВЛД.

В каждом узле дерева находиться пара «ключ-значение», которая также реализована в виде АТД. Реализовать в словаре итераторы.

На основе словаря организовать разреженный вектор с возможностью обработки с помощью функций, принимающих на вход итераторы.

Все базовые ТД покрыть Unit-тестами и разработать консольный пользовательский интерфейс.

# Функциональные требования

* 1. ТД «Разреженный вектор» должен позволять хранить значения любых типов. Для этого класс SparseVector должен быть объявлен как шаблонный. Пример:

template<typename T>

class SparseVector {

private:

unsigned int size;

...

T defaultValue;

private:

void resetUsedIndex();

void refreshItems();

public:

SparseVector(...);

…

}

T – это параметр, обозначающий тип вектора.

* 1. ТД «Разреженный вектор» (SparseVector) должен обладать, по крайней мере, следующими атрибутами и методами:
  2. Атрибуты SparseVector:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Сигнатура** | **Назначение** |
| 2.7.1 | size | unsigned int size | Длина разреженного вектора |

* 1. Методы (операции) SparseVector:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Сигнатура** | **Назначение** |
|  | append | void append(T value) | Добавление элемента в конец вектора. |
|  | prepend | void prepend(T value) | Добавление элемента в начало вектора. |
|  | insert | void insert(unsigned int index, T value) | Вставка в вектор значения value типа T по индексу index |
|  | pop | void pop(unsigned int index) | Удалить элемент с индексом, если такого нет, то выбросить исключение |
|  | operator[] | T& operator[](unsigned int index) | вставка, изменение и получение элемента |
|  | find | bool find(size\_t index) | Проверить, есть ли элемент с индексом index в векторе, вернуть True/False |
|  | swap | void swap(unsigned int index1, unsigned int index2) | Поменять элементы местами с проверкой индекса на граничные значения |
|  | remove | void remove() | Очистить целиком разреженный вектор |
|  | begin | typename array::Array<T>::ArrayIterator begin() | Итератор, указывающий на начало вектора |
|  | end | typename array::Array<T>::ArrayIterator end() | Итератор, указывающий на конец вектора |

* 1. Для реализации итераторов в словаре и разреженном векторе реализовать итератор по АТД «Последовательность».
  2. АТД «Словарь» должен позволять хранить пары «ключ-значение» любых типов. Для этого класс IDictionary должен быть объявлен как шаблонный. Пример:

template <typename K, typename V>

class IDictionary {

protected:

unsigned int size;

unsigned int t{};

...

public:

IDictionary();

…

}

K – это параметр, обозначающий тип ключа;

V - это параметр, обозначающий тип значения.

* 1. АТД «Словарь» (“IDictionary”) должен обладать, по крайней мере, следующими атрибутами и методами:
  2. Атрибуты

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Сигнатура** | **Назначение** |
|  | size | unsigned int size | Длина словаря (количество пар «ключ-значение») |

* 1. Методы (операции):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Сигнатура** | **Назначение** |
|  | insert | void insert (K key, V value) | Добавление пары в словарь. |
|  | get | KVElement<K, V> &get(K key) | Получить значение по ключу с возможностью изменения. Если пары с таким ключом в словаре нет, выбросить исключение «Key doesn't exist» |
|  | operator[] | KVElement<K, V> &operator[](K key) | Перегрузить оператор квадратные скобки для вставки, изменения и получения элемента |
|  | contains | bool contains(K key) | Проверка на вхождение ключа в словарь |
|  | remove | void remove(K key) | Удалить пару с ключом, если таковой нет, выбросить исключение «Key doesn't exist» |
|  | values | sequence::SequenceIterator<V> values() | Получение последовательности значений |
|  | items | sequence::SequenceIterator<Pair<K, V>> items() | Получение последовательности пар «ключ-значение» |

# Требования к структурам данных и алгоритмам

* 1. Должны быть организованы следующие структуры данных:
* «Разреженный вектор» SparseVector
* Получение размера разреженного вектора
* Добавление элемента
* Удаление элемента
* Получение элемента по индексу (добавление элемента по умолчанию в случае отсутствия индекса)
* Перегрузка оператора «квадратные скобки» для получения/добавления/изменения элемента
* Вывод разреженного вектора
* Обработка разреженного вектора с помощью итераторов
* «Словарь» IDictionary
* Получение размера словаря
* Добавление пары «ключ-значение»
* Удаление пары «ключ-значение»
* Поиск по ключу
* Получение значения по ключу (с выбросом исключения в случае отсутствия ключа)
* Получение значения по ключу (добавление значения по умолчанию в случае отсутствия ключа)
* Перегрузка оператора «квадратные скобки» для получения/добавления/изменения значения
* Вывод словаря в консоль
  1. За основу IDictionary взять BД и АВЛД

# Требования к интерфейсу

Реализовать консольное приложение для демонстрации работы SparseVector и “Словарей”

# Требования к форматам входных и выходных данных

## Требования к форматам входных данных

Вводить вектор в формате Value1 Value ,…,ValueN

## Требования к форматам выходных данных

Выводить словарь в формате {Key1: Value1, Key2: Value2, …, KeyN: ValueN}.

Выводить разреженный вектор в формате [Element1, Element2, …, ElementN].

1. **Требования к Unit-тестам**

Покрыть Unit-тестами