# Лабораторная работа №2 по курсу «Программирование (объектно-ориентированное программирование)»

## Техническое задание

## 1. Постановка задачи

Написать программу на языке C++, реализующую следующие алгоритмы:

1) Построение алфавитного указателя по заданному тексту и размеру страницы (размер страницы задается в словах).
2) Построение разреженной матрицы по заданной последовательности (предполагается, что в последовательности содержится 80-85% нулевых элементов).

## 2. Функциональные требования

2.1. Программа должна строить по тексту и размеру одной страницы строить специальный тип данных, алфавитный указатель (**AlphabetPointer**), представляющий собой множество объектов типа пар «ключ-значение», в котором ключ является словом, встречающимся в тексте, а значением последовательность целых чисел, обозначающий на каких страницах встречается указанное слово.

2.2. Программа должна преобразовывать последовательность (**Sequence**) в разреженный вектор (**SparseSeq**), являющийся набором пар типа «ключ-значение», в которых ключом является индекс элемента в исходной последовательности, а значением непосредственно сам элемент исходной последовательности. При этом SparseSeq не должен хранить в качестве значений нулевые элементы (подразумевая, что все элементы, у которых нет ключа, являются нулевыми).

2.3. Все основные типы данных покрыть unit-тестами. Программа должна иметь возможность по вводу пользователя проверять данные тесты.

2.4. Программа должна иметь консольный пользовательский интерфейс, имитирующий командную оболочку (в качестве ввода пользователю предлагается ввести команду для взаимодействия с программой).

## 3. Требования к типам данных

3.1. Базовый тип данных, **DictionaryTree<Key, Value>**, должен представлять из себя модификацию бинарного дерева поиска, в каждом узле которого должны храниться ключ и значение. Сравнение элементов данного дерева производить по ключу.

Краткая спецификация DictionaryTree:

| Название | Сигнатура | Описание |
|---|---|---|
| Атрибуты | | |
| Ключ | `Key selfKey;` | Ключ узла дерева. |
| Значение | `Value value;` | Значение узла дерева. |
| Правое поддерево | `DictionaryTree *right;` | Указатель на правое поддерево. |
| Левое поддерево | `DictionaryTree *left;` | Указатель на левое поддерево. |
| Методы | | |

| | | |
|---|---|---|
| Конструктор инициализации узла | `DictionaryTree(Key k, Value v);` | Создает дерево с заданным ключом и значением. |
| IsEmpty | `bool isEmpty();` | Возвращает true, если дерево пустое, false в противном случае. |
| Add | `void add(Key k, Value val);` | Добавляет новую пару в дерево. Если такой ключ уже есть в дереве – исключение. |
| Get | `Value get(Key key);` | Возвращает значение элемента по заданному ключу. Исключение в случае отсутствия такого ключа. |
| Contains | `bool contains(Key key);` | Возвращает true, если такой ключ существует в дереве, false в противном случае. |
| Remove | `void remove(Key key);` | Удаляет данный узел из дерева. |
| Size | `int size();` | Возвращает количество элементов в дереве. |
| UpdateValue | `void updateValue(Key key, Value newValue);` | Если ключ key есть в дереве, то обновляет значение, отвечающее этому ключу. Если же такого key нет, то является аналогом Add. |
| Concatenate | `DictionaryTree<Key, Value> *unitedWith(DictionaryTree<Key, Value> *tree1);` | Возвращает указатель на дерево, являющееся объединением двух других деревьев. |
| Thread | `Sequence<std::pair<Key, Value>> *thread(std::string order);` | Прошивка дерева. Принимает параметр обхода (напр. “NLR”) возвращает последовательность пар «ключ-значение». |

3.2. Тип данных «Словарь», **Dictionary<Key, Value>**, представляет собой тип данных, инкапсулирующий в себе структуру DictionaryTree<Key, Value>. Является более высокоуровневой абстракцией, не зависящей от способа реализации DictionaryTree<Key, Value>.

Краткая спецификация класса Dictionary<Key, Value>:

| Название | Сигнатура | Описание |
|---|---|---|
| Атрибуты | | |
| Бинарное дерево поиска | `DictionaryTree<Key, Value> *dictionaryTree;` | Представляет собой инкапсулированный объект типа `DictionaryTree`. |
| Методы | | |
| Конструктор создания пустого словаря | `Dictionary();` | |
| Конструктор копирования | `Dictionary(const Dictionary<Key, Value> &dictionary);` | |
| IsEmpty | `bool isEmpty();` | Проверка на пустой словарь. |

| | | |
|---|---|---|
| Get | `Value get(Key key);` | Возвращает значение элемента по заданному ключу. |
| Add | `void add(Key key, Value val);` | Добавляет новую пару в словарь. |
| Contains | `bool contains(Key key);` | Проверка на наличие . |
| Remove | `void remove(Key key);` | Удаление элемента из словаря по заданному ключу. |

Класс Dictionary должен инкапсулировать и все остальные методы класса DictionaryTree.

3.3. Тип данных «Алфавитный указатель», **AlphabetPointer**, должен представлять собой класс, в котором инкапсулирован процесс преобразования строкового типа данных (std::string) в словарь, в котором ключ представляет собой строку – слово, встречающееся в тексте, а значением является последовательность (Sequence) целых чисел, номеров страниц, на которых встречается слово.
Данный класс должен быть представлен единственным методом – конструктором, в котором реализуется соответсвующий алгоритм обработки.

3.4. Тип данных «Разреженный вектор», **SparseSeq<T>**, должен представлять собой класс, в котором в качестве его поля содержится словарь типа Dictionary<int, T>. Ключом является целое число, представляющее индекс ненулевого элемента в изначальной последовательности. Класс должен предоставлять 2 способа обработки исходной последовательности: стандартный способ с циклическим проходом коллекции элементов и реализацию в стиле map-reduce.
Краткая спецификация класса SparseSeq:

| Название | Сигнатура | Описание |
|---|---|---|
| Атрибуты | | |
| Словарь | `Dictionary<int, T> *storage;` | Словарь для хранения элементов разреженного вектора. |
| Length | `int length;` | Представляет собой размер изначальной последовательности. |
| Нулевое значение | `T null;` | Представляет собой нулевой элемент в данном векторе. |
| isNull | `bool (*isNull)(T);` | Указатель на функцию, возвращающую true в случае совпадения элемента с нулевым значением. |
| Методы | | |
| Конструктор | `SparseSeq(Sequence<T> *seq, T null, bool (*isNull)(T), bool mapReduceOn = false);` | Основной конструктор, в котором происходит преобразование изначальной последовательности seq в разреженный вектор (mapRdeueceOn задает способ обработки). |
| Конструктор копирования | `SparseSeq(const SparseSeq<T> &seq);` | |
| Length | `int getLength();` | Возвращает размер исходной последовательности. |

| | | |
|---|---|---|
| Get | `T get(int index);` | По заданному индексу возвращает значение элемента в векторе. |
| GetNull | `T getNull();` | Возвращает нулевое значение, установленное для этого вектора. |

Основные отношения между классами должны соответствовать следующей диаграмме (рис.1).

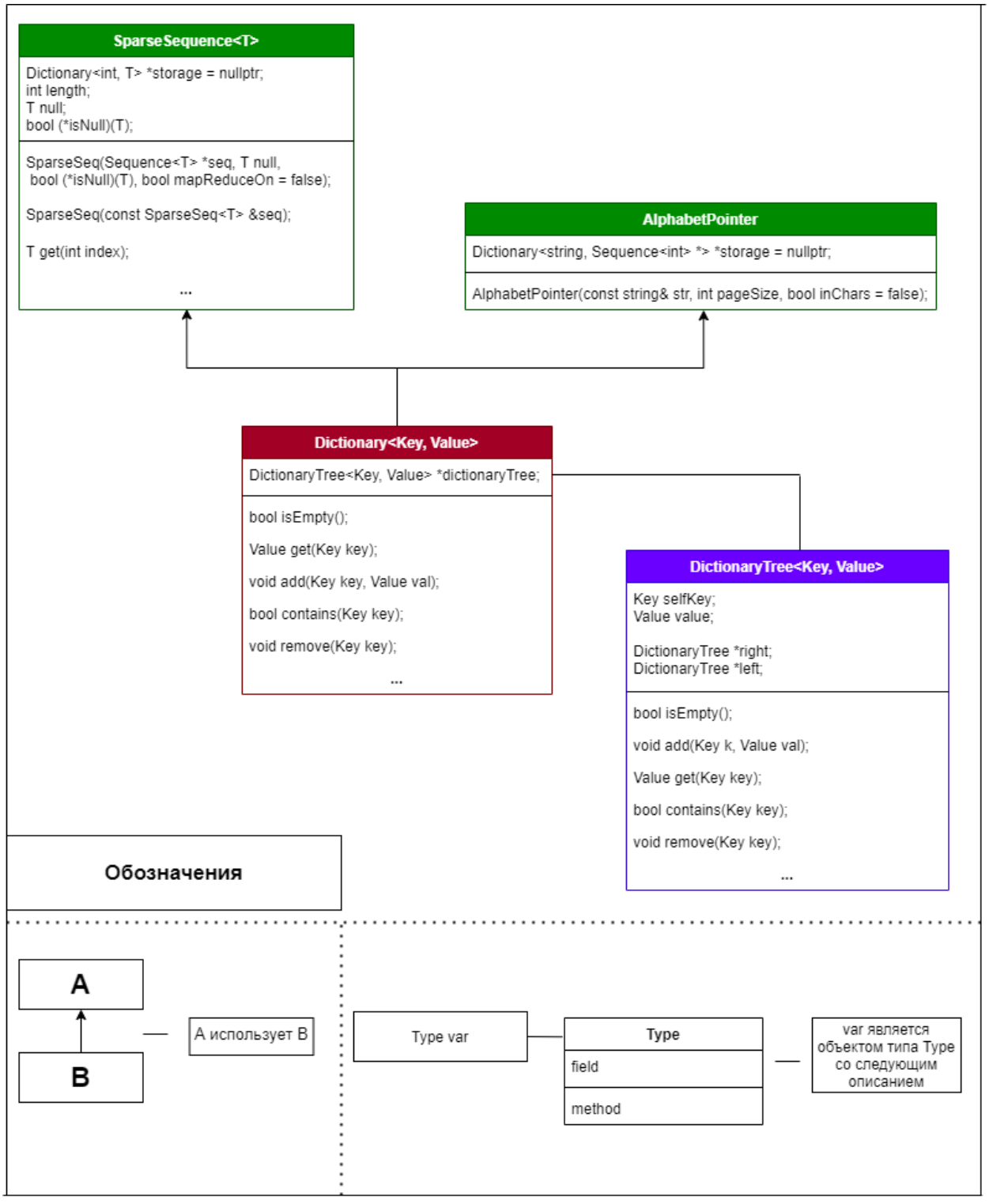


*Рис. 1 (диаграмма классов)*