Фонд оценочных средств  
по дисциплине  
«Языки программирования и   
методы программирования»

Приложение 1

Лабораторная работа №1

(информатика, 3 семестр)

Техническое задание

Используемые термины и сокращения

|  |  |
| --- | --- |
| АТД | Абстрактный тип данных |
|  |  |
|  |  |

# Постановка задачи

На языке C++ реализовать АТД «Последовательность». Написать unit-тесты.

# Функциональные требования

* 1. АТД должен позволять хранить элементы любых типов. Для этого класс должен быть объявлен как шаблонный. Пример:

template<typename TElement> class Sequence {

public:

int GetLength();

…

<TElement> GetFirst();

<TElement> GetLast();

<TElement> Get(int index);

Sequence<TElement> GetSubsequence(int i, int j);

…

}

TElement – это параметр, обозначающий тип элемента[[1]](#footnote-1).

АТД «Последовательность» («Sequence») должен обладать, по крайней мере, следующими атрибутами и методами:

* 1. Атрибуты

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Сигнатура** | **Назначение** |
|  | Length | int getLength() | Длина последовательности (количество элементов) |
|  | IsEmpty | int getIsEmpty() | Признак того, является ли последовательность пустой |

* 1. Методы (операции):

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название** | **Сигнатура** | **Назначение** |
|  | Get | <TElement> Get(int index) | Получение элемента по индексу. |
|  | GetFirst | <TElement> GetFirst() | Получить первый элемент последовательности |
|  | GetLast | <TElement> GetLast | Получить последний элемент последовательности |
|  | GetSubsequence | Sequence<TElement> GetSubsequence(int startIndex, int endIndex) | Получить подпоследовательность: начиная с элемента с индексом startIndex и заканчивая элементом с индексом endIndex |
|  | Append | void Append(<TElement> item) | Добавить элемент в конец последовательности |
|  | Prepend | void Prepend(<TElement> item) | Добавить элемент в начало последовательности |
|  | InsertAt | void InsertAt(int index, <TElement> item) | Вставить элемент в позицию index (существующие элементы сдвигаются вперед на 1). |
|  | Remove | void Remove(<TElement> item> | Удаляет элемент из последовательности. Если элемента не было, ничего не происходит. |

# Требования к структурам данных и алгоритмам

* 1. Для хранения данных реализовать следующие варианты:
     1. Использовать массивы C++
     2. Использовать связные списки

Выполнить реализацию, использующую массивы (ArraySequence), и реализацию, использующую связные списки (ListSequence). Сделать оба эти класса наследниками абстрактного класса Sequence.

* 1. Все методы, принимающие на вход индексы, должны выбрасывать исключение в ситуациях, когда индекс выходит за допустимые границы: меньше 0 или больше или равен числе элементов в последовательности.

# Требования к интерфейсу

Нет.

# Требования к форматам входных и выходных данных

Требования к форматам входных данных

Нет.

Требования к форматам выходных данных

Нет.

# Требования к unit-тестам

* 1. Написать, по крайней мере, один тест[[2]](#footnote-2), реализующий следующий сценарий:
     1. Создать пустую последовательность (TElement = int).
     2. Проверить, что длина = 0
     3. Добавить (Append) элемент «23».
     4. Проверить, что длина = 1
     5. Проверить, что GetFisrt возвращает «23»
     6. Проверить, что GetLast возвращает «23»
     7. Проверить, что Get(0) возвращает «23», а Get(-1) и Get(1) – выбрасывают исключение.
     8. Добавить (Append) элемент «43»
     9. Проверить, что длина = 2
     10. Проверить, что GetFisrt возвращает «23»
     11. Проверить, что GetLast возвращает «43»
     12. Проверить, что Get(0) возвращает «23», Get(1) – «43», а Get(-1) и Get(2) – выбрасывают исключение.
     13. Добавить (Prepend) элемент «53»
     14. Проверить, что длина = 3
     15. Проверить, что GetFisrt возвращает «53»
     16. Проверить, что GetLast возвращает «43»
     17. Проверить, что Get(0) возвращает «53», Get(1) – «23», а Get(-1) и Get(3) – выбрасывают исключение.
     18. Проверить, что GetSubsequence(1,1) возвращает последовательность, у которой:
         1. длина =1
         2. первый и последний элемент – 23

1. В первом приближении можно выполнить «обычную» реализацию, на примере какого-либо простого типа элементов, скажем, int. Обобщение до поддержки любого типа будет почти механическим. [↑](#footnote-ref-1)
2. Имеется ввиду, что должен быть хотя бы один тест, и этот тест должен реализовывать указанный сценарий (а не то, один или несколько тестов должны его реализовывать). [↑](#footnote-ref-2)