Семестр 3 (2019), Список задач 1

В следующих задачах требуется разработать либо один класс (основной), либо два класса (основной и вспомогательный). Для каждого класса должен быть создан заголовочный файл `.h`, в который помещается описание класса, и файл `.cpp`, в который помещаются определения методов класса. Файлы программы должны быть сохранены в отдельном каталоге. Исполняемый файл программы должен собираться с помощью утилиты `make`.

Основной класс должен содержать конструктор по умолчанию, конструктор копирования, операция присваивания. Должна быть разработана дружественная функция для печати текстового представления экземпляра класса.

При некорректных значениях входных параметров публичный метод класса должен возбуждать исключение.

A. Класс `Polynomial`

Класс полиномов с коэффициентами из множества `T`, содержащий следующие методы.

Конструктор преобразования, создающий полином нулевой степени. Конструктор, создающий полином вида вида ax^n.

Методы, реализующие суммирование (операции `+` и `+=`), вычитание (операции `-` и `-=`) и умножение (операции `*` и `*=`) полиномов.

Методы, реализующие прибавление константы (операции `+` и `+=`), вычитание константы (операции `-` и `-=`) и умножения на константу (операции `*` и `*=`).

Операции сравнения полиномов `==`, `!=`.

Операция вызова функции `()`, вычисляющая значение полинома в заданной точке (схема Горнера).

Метод, возвращающий степень полинома.

Варианты:

A.1. В качестве `T` выступает `double`.

A.2. В качестве `T` выступает вспомогательный класс `Ratio` (рациональное число).

A.3. В качестве `T` выступает вспомогательный класс `Complex` (комплексное число).

A.4. В качестве `T` выступает вспомогательный класс `Zp` (элемент поля вычетов $\mathbb{Z}_p$ по простому модулю p; p – фиксированно).

B. Класс `Set`

Класс, представляющий собой битовую реализацию множества целых чисел, и содержащий следующие методы.

Конструктор преобразования, создающий одноэлементное множество.

Методы, реализующие объединение (операции `+` и `+=`), разность (операции `-` и `-=`) и пересечение (операции `*` и `*=`) множеств.

Операции сравнения множеств `==`, `!=`, `>`, `>=`, `<`, `<=`.

Операция вызова функции `()`, позволяющая проверить принадлежность элемента множеству.

Метод, проверяющий множество на пустоту. Метод, возвращающий размер множества.

Варианты:

B.1. Множество неотрицательных целых чисел.

B.2. Элементами множества могут быть как отрицательные, так и неотрицательные целые числа.

C. Класс `Number`

Класс целых чисел, содержащий следующие методы.

Конструктор преобразования из типа `int`.

Операции преобразования типа `(int)` и `(double)`.

Методы, реализующие сумму (операции `+` и `+=`), разность (операции `-` и `-=`) и умножение (операции `*` и `*=`) чисел.

Операции сравнения чисел `==`, `!=`, `>`, `>=`, `<`, `<=`.

Варианты:

C.1. В качестве внутреннего представления числа используется битовый вектор, интерпретируемый как запись числа в двоичной системе счисления.

C.2. В качестве внутреннего представления числа используется его запись в десятичной системе счисления, которая хранится в виде массива `char`.

D. Класс `String`

Класс строк, состоящих из однобайтовых символов, и содержащий следующие методы.

Конструктор преобразования из `const char *`. Конструктор преобразования из `int`, создающий строку, содержащую текстовое представление целого числа. Конструктор преобразования из `double`, создающий строку, содержащую текстовое представление вещественного числа.

Операции преобразования типа `(int)` и `(double)`.

Метод, реализующий добавление символа в конец строки (операции `+` и `+=`).

Метод, реализующий добавление строки в конец строки (операции `+` и `+=`).

Метод, реализующий поиск подстроки в строке. Возвращает номер позиции первого вхождения подстроки в строке или число `-1`).

Метод, реализующий быструю сортировку символов строки.

Метод, проверяющий строку на пустоту. Метод, возвращающий длину строки.

E. Класс `Array`

Класс, реализующий динамический массив элементов типа `T`.

Конструктор преобразования, создающий одноэлементный массив. Конструктор, создающий массив, состоящий из заданного числа одинаковых элементов.

Операция индексирования `[]`, реализующая доступ к элементу массива по его индексу.

Метод, реализующий добавление элемента в конец массива (операции `+` и `+=`).

Метод, реализующий добавление массива в конец массива (операции `+` и `+=`).

Метод, реализующий умножение массива (операции `*` и `*=`) на неотрицательное целое число. Результатом является массив, состоящий из заданного числа подряд идущих копий исходного массива.

Метод, реализующий быструю сортировку массива.

Метод, проверяющий массив на пустоту. Метод, возвращающий размер массива.

Варианты:

E.1. В качестве `T` выступает `double`.

E.2. В качестве `T` выступает вспомогательный класс `Ratio` (рациональное число).

E.3. В качестве `T` выступает вспомогательный класс `Complex` (комплексное число).

E.4. В качестве `T` выступает вспомогательный класс `Zp` (элемент поля вычетов $\mathbb{Z}_p$ по простому модулю p; p – фиксированно).