Задания к работе 1 по языкам и методам программирования.

1. Реализовать класс encoder. В классе определить и реализовать:
   - Конструктор, принимающий ключ шифрования (массив байтов типа unsigned char const *) и размер этого массива
   - Деструктор
   - Конструктор копирования
   - Метод encode, который принимает путь ко входному файлу (типа char const *), выходному файлу (типа char const *) и флаг, отвечающий за то, выполнять шифрование или дешифрование (типа bool) и выполняет процесс шифрования/дешифрования файла
   - поле, хранящее значение ключа
   - mutator для значения ключа

Шифрование/дешифрование файлов выполняется алгоритмом RC4. Структура содержимого файлов произвольна.
Продемонстрировать работу класса, создав текстовый файл и произведя шифрование/дешифрование данного файла.

2. Реализовать класс logical_values_array. В классе определить и реализовать:
   - поле _value (типа unsigned int), которое хранит значение логической величины
   - accessor для поля value
   - конструктор, принимающий значение типа unsigned int (равное по умолчанию 0) и инициализирующий переданным значением поле value
   - методы, соответствующие всем стандартным логическим операциям: инверсия, конъюнкция, дизъюнкция, импликация, коимпликация, сложение по модулю 2, эквивалентность, стрелка Пирса, штрих Шеффера. Примечание: если одну операцию возможно выразить через другую, то необходимо реализовывать одно через другое (например, эквивалентность можно реализовать через сложение по модулю 2 и инверсию)
   - статический метод equals, сравнивающий два числа по отношению эквивалентности
   - метод get_bit, который возвращает значение бита по его позиции (является параметром)
   - метод, принимающий значение типа char *; по значению адреса в параметре должно быть записано двоичное представление поля value в виде строки в стиле языка программирования C. Примечание: конвертация должна быть основана на использовании битовых операций.

Продемонстрируйте работу реализованного функционала.

3. Реализовать класс комплексного числа. В классе определить и реализовать:
   - поля, соответствующие действительной и мнимой части комплексного числа (типа double)
   - конструктор, который принимает значения действительной и мнимой части (оба параметра по умолчанию равны 0)
   - методы, производящие операции сложения, вычитания, умножения и деления комплексных чисел
   - метод, возвращающий модуль комплексного числа
   - метод, возвращающий аргумент комплексного числа

Продемонстрируйте работу реализованного функционала.

4. Реализовать класс матрицы квадратной матрицы с вещественными значениями. В классе определить и реализовать:

- поля для элементов матрицы (типа double **) и размерности матрицы (типа size_t)
- конструктор, инициализирующий состояние матрицы на основе размерности (первый параметр) и значений (список аргументов переменной длины) построчно; вместо недостающих параметров в списке аргументов переменной длины по умолчанию берутся нули
- конструктор копирования
- оператор присваивания
- деструктор
- методы, осуществляющие сложение матриц, умножение матриц, умножение матрицы на число, умножение числа на матрицу, вычитание матриц
- метод, возвращающий значение определителя матрицы, вычисленное методом Гаусса
- метод, возвращающий транспонированную матрицу
- метод, возвращающий обратную матрицу
- * перегруженный оператор [], возвращающий значение элемента по его индексам строки и столбца (индексирование начинается с 0), с возможностью его модификации

Продемонстрируйте работу реализованного функционала.

5. Реализовать класс двоичной приоритетной очереди. В классе определить и реализовать:

- поля для элементов приоритетной очереди, виртуального размера и физического размера приоритетной очереди, а также для правила вычисления отношения порядка на пространстве приоритетов
- конструктор, инициализирующий приоритетную очередь как пустую очередь с буффером на 16 элементов
- конструктор копирования
- оператор присваивания
- деструктор
- метод добавления значения по приоритету (с временно́й сложностью O(log(N)))
- метод удаления значения с минимальным приоритетом (с временно́й сложностью O(log(N)))
- метод поиска элемента с минимальным приоритетом (с временно́й сложностью O(1))
- метод слияния двух приоритетных очередей без разрушения исходных (с временно́й сложностью $O(N_{1 \text{ or } 2} + \text{const})$

Тип значений элементов приоритетной очереди - char * (строка в стиле языка программирования C), приоритетов - натуральные числа типа size_t.

Продемонстрируйте работу реализованного функционала.