Вариант 1. (2 человека)

Разработать шаблон класса TMatrix для работы с матрицами размера M x N элементов произвольного типа. Полями класса должны быть размеры матрицы и вектор, состоящий из векторов чисел элементов типа T.

Разработать конструкторы:

- Конструктор «пустой» матрицы
- Копирующий конструктор
- Конструктор матрицы заданных размеров
- Разработать методы:
- Транспозиция матрицы
- Ранг матрицы

Перегрузить операции:

- Операция получения строки матрицы с заданным номером по индексу
- Сложение матриц (выбрасывать исключение, если не подходят размеры)
- Вычитание матриц (выбрасывать исключение, если не подходят размеры)
- Перемножение матриц (выбрасывать исключение, если не подходят размеры)
- Умножение/деление матрицы на число (выбрасывать исключение при попытке деления на ноль)
- Вывод в поток

Продемонстрировать использование класса на матрице с целочисленными элементами и матрице с элементами типа double

Вариант 2. (2 человека)

Разработать шаблон класса SquareMatrix для работы с квадратными матрицами размера N x N элементов произвольного типа.

Полями класса должны быть размеры матрицы и вектор, состоящий из векторов элементов типа T.

Разработать конструкторы:

- Конструктор с целым параметром N, который создаёт матрицу размером NxN

Создать методы:

- Определитель
- Обратная матрица (для вырожденной матрицы генерируем исключение)
- Решение СЛАУ, где правая часть СЛАУ – вектор значений типа T

Переопределить операции:

- Деление матрицы на матрицу (A/B=A*(матрица, обратная к B))

- Продемонстрировать использование класса на матрице с целочисленными элементами и матрице с элементами типа double

Вариант 3 (1 человек)

Создать класс IntegerSet, который представляет собой множество целых значений. Значения сохраняются в массиве, размер которого увеличивается по необходимости.

Поля класса:

- Указатель на динамически созданный массив целых чисел: int *elements;
- Целое число – емкость множества, в котором хранится текущий размер массива. Оно изменяется, если размер массива – недостаточный для добавления новых элементов
- Целое число – количество элементов множества;

Класс должен содержать методы

1. Конструктор по умолчанию, который инициализирует массив из десяти элементов, изменяет параметр емкости множества и устанавливает значения элементов массива в ноль.
2. Копирующий конструктор, который устанавливает размер массива элементов текущего элемента равным размеру массива копируемого объекта, копирует все значения массива, а также копирует значения ёмкости и количества элементов.
3. Перегруженный оператор присвоения, который устанавливает размер массива элементов текущего элемента равным размеру массива объекта в правой части, копирует все значения массива, а также копирует значения ёмкости и количества элементов и возвращает указатель на текущий элемент.
4. Деструктор, удаляющий массив.
5. Метод size(), возвращающий количество элементов
6. Метод isEmpty(), который возвращает истину, если множество не содержит ни одного элемента.
7. Метод contains(int), который принимает целый параметр и возвращает истину, если значение содержится в IntegerSet.
8. Метод add(int), который принимает целый параметр и добавляет его в множество (если оно ещё не содержится в нём). Если массив уже полностью заполнен, тогда следует создать новый массив целых чисел вдвое большего размера, чем текущий массив, скопировать в него все значения, удалить старый массив и направить указатель elements на новый массив
9. Метод remove(int), который принимает целое значение и удаляет его из множества (если оно там есть), при этом количество элементов уменьшается на единицу и последующие элементы массива сдвигаются влево
10. Метод getAsVector(), который возвращает вектор целых чисел. Содержащий все значения в множестве. Порядок не имеет значения.

Перегрузить операторы

- Перегруженный оператор ‘+’, который возвращает новое множество – объединение двух множеств
- Перегруженный оператор ‘*’, который возвращает новое множество – пересечение двух множеств
- Перегруженный оператор ‘/’, который возвращает новое множество – симметричную разность двух множеств

Вариант 4. (2 человека)

Создать класс Polynom – многочлен с рациональными коэффициентами

Коэффициенты многочлена хранятся в односвязном списке.

Создать конструкторы, перегрузить операции сложения, вычитания, умножения, операцию / использовать для отыскания частного от деления многочленом, % - для отыскания частного от деления многочленов

Создать методы:

- evaluate (отыскания значения по схеме Горнера)
- derivative – производная
- primitive – первообразная
- newton – отыскание корня многочлена по методу Ньютона

Вариант 5. Многоугольник (3 человека)

Определить класс Point с компонентами double x, double y. Создать метод double distanceTo(const Point&), определяющий расстояние до другой точки.

Определить класс Polygon с полем – массивом вершин.

- Создать конструктор Polygon(vector<Point>), в котором происходит проверка, что вершины образуют ломаную линию **без самопересечений.**

Создать методы:

- Площадь многоугольника
- Периметр многоугольника
- Для заданной точки на плоскости определить, принадлежит ли она многоугольнику

Определить, является ли многоугольник выпуклым

- *Изобразить его на плоскости

Вариант 6. Очередь с приоритетом на основе двоичного дерева поиска (класс-шаблон) **(2 человека)**

Создать класс Node<T> – узел дерева с полями T data, Node *left, Node *right с конструкторами

Создать двоичное дерево поиска class BSTree с полем Node *root

Создать методы

- Enqueue – добавить элемент в очередь
- IsEmpty – проверка на пустоту
- Dequeue – извлекает из очереди наибольший элемент и возвращает его в качестве результата

Вариант 7 (2 человека) Длинное целое число

Класс BigInteger содержит единственное поле – строку, состоящую из цифр

Перегрузить все арифметические операции и операции сравнения.

Вариант 8 (1 человек) Игра в крестики-нолики. Проект предусматривает создание игры в крестики-нолики, таким образом, что играть могут как два пользователя, так и человек с компьютером.

Вариант 9 (1 человек) Поиск выхода из лабиринта. Лабиринт задаётся при помощи текстового файла.

Вариант 10 (3 человека) Собственная реализация хеш-таблицы. Задача подразумевает реализацию хеш-таблицы, а также двоичного дерева поиска с замерами времени работы.