1. Реализовать на C++ стек целых чисел (глубина стека задаётся аргументом конструктора). Необходимые методы: создание стека (конструктор), удаление стека (деструктор), копирование стека (конструктор копирования и оператор присваивания), перемещение стека (конструктор перемещения и оператор присваивания перемещением), добавление числа в стек, удаление числа из стека.
2. Реализовать на C++ стек строк std::string (глубина стека задаётся аргументом конструктора). Необходимые методы: создание стека (конструктор), удаление стека (деструктор), копирование стека (конструктор копирования и оператор присваивания), перемещение стека (конструктор перемещения и оператор присваивания перемещением), добавление строки в стек, удаление строки из стека.
3. Реализовать на C++ очередь целых чисел (длина очереди задаётся аргументом конструктора). Необходимые методы: создание очереди (конструктор), удаление очереди (деструктор), копирование очереди (конструктор копирования и оператор присваивания), перемещение очереди (конструктор перемещения и оператор присваивания перемещением), добавление числа в очередь, удаление числа из очереди.
4. Реализовать на C++ очередь строк std::string (длина очереди задаётся аргументом конструктора). Необходимые методы: создание очереди (конструктор), удаление очереди (деструктор), копирование очереди (конструктор копирования и оператор присваивания), перемещение очереди (конструктор перемещения и оператор присваивания перемещением), добавление строки в очередь, удаление строки из очереди.
5. Реализовать на C++ однонаправленный список целых чисел. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление числа в список, удаление числа из списка, сортировка списка «пузырьком» в порядке возрастания. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
6. Реализовать на C++ однонаправленный список строк std::string. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление строки в список, удаление строки из списка, сортировка списка «пузырьком» в алфавитном порядке. Реализация списка – классическая ссылочная, с использованием «умных» указателей.

7. Реализовать на C++ двунаправленный список целых чисел. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление числа в список, удаление числа из списка, сортировка списка методом быстрой сортировки в порядке возрастания. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
8. Реализовать на C++ двунаправленный список строк std::string. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление строки в список, удаление строки из списка, сортировка списка методом быстрой сортировки в алфавитном порядке. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
9. Реализовать на C++ двунаправленный список целых чисел. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление числа в список, удаление числа из списка, сортировка списка методом сортировки слиянием в порядке возрастания. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
10. Реализовать на C++ двунаправленный список строк std::string. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление строки в список, удаление строки из списка, сортировка списка методом сортировки слиянием в алфавитном порядке. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
11. Реализовать на C++ бинарное дерево поиска различных целых чисел. Необходимые методы: инициализация дерева (конструктор), удаление дерева (деструктор), поиск числа в дереве, добавление числа в дерево, удаление числа из дерева. Реализация дерева – с использованием «умных» указателей. Также необходимо реализовать метод, который выводит в командную строку элементы дерева «по этажам»: т.е. на первую строку – корень, на вторую – потомков корня, на третью – потомков потомков корня и т.д.
12. Реализовать на C++ бинарное дерево поиска различных строк std::string. Необходимые методы: инициализация дерева (конструктор), удаление дерева (деструктор), поиск строки в дереве, добавление строки в дерево, удаление строки из дерева. Реализация дерева – с использованием «умных» указателей. Также необходимо реализовать метод, который выводит в командную строку элементы дерева «по этажам»: т.е. на первую строку – корень, на вторую – потомков корня, на третью – потомков потомков корня и т.д.

13. Реализовать на C++ двунаправленный список целых чисел. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление числа в список, удаление числа из списка, сортировка списка «пузырьком» в порядке возрастания. Реализация списка – классическая ссылочная, с использованием «умных» указателей. Также необходимо реализовать метод, который в отсортированном списке находит (и выводит, например, в командную строку) все пары элементов, сумма которых равна заданному числу – причём работать этот метод должен с линейной сложностью и с использованием не более чем константной дополнительной памяти (время работы метода необходимо засекать и выводить в командную строку). Для проверки метода необходимо реализовать функцию, которая заполняет список заданным количеством целых чисел, сгенерированных случайным образом.
14. Реализовать на C++ однонаправленный список целых чисел. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление числа в список, удаление числа из списка. Реализация списка – классическая ссылочная, с использованием «умных» указателей. Также необходимо реализовать метод, который «разворачивает» список (т.е. преобразует его, располагая все элементы в обратном порядке, превращая голову списка в хвост, а хвост в голову) – причём работать этот метод должен с линейной сложностью и с использованием не более чем константной дополнительной памяти (время работы метода необходимо засекать и выводить в командную строку). Для проверки метода необходимо реализовать функцию, которая заполняет список заданным количеством целых чисел, сгенерированных случайным образом.
15. Реализовать на C++ однонаправленный список целых чисел. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление числа в список, удаление числа из списка. Реализация списка – классическая ссылочная, с использованием «умных» указателей. Также необходимо реализовать метод, который находит два элемента a и b списка, находящиеся на расстоянии i и j элементов от начала, что разность a – b максимальна (при этом i <= j). Метод должен работать с линейной сложностью и с использованием не более чем константной дополнительной памяти (время работы метода необходимо засекать и выводить в командную строку). Для проверки метода необходимо реализовать функцию, которая заполняет список заданным количеством целых чисел, сгенерированных случайным образом.

16. Реализовать на C++ однонаправленный список треугольников с целочисленными вершинами на плоскости с возможностью отбрасывания пересекающихся элементов. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление треугольника в список (с проверкой, что он не пересекается с уже имеющимися треугольниками), удаление треугольника из списка. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
17. Реализовать на C++ двунаправленный список треугольников с целочисленными вершинами на плоскости с возможностью отбрасывания пересекающихся элементов. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление треугольника в список (с проверкой, что он не пересекается с уже имеющимися треугольниками), удаление треугольника из списка. Реализация списка – классическая ссылочная, с использованием «умных» указателей.
18. Реализовать на C++ однонаправленный список связных множеств точек на плоскости с целочисленными координатами (две точки принадлежат одному связному множеству, если максимум модуля разности координат не превышает единицы). Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление точки в список (в этом случае либо добавляется новое связное множество из одной точки, либо точка добавляется к одному из множеств списка, либо точка объединяет два или более множеств). Реализация списка – классическая ссылочная, с использованием «умных» указателей.
19. Реализовать на C++ двунаправленный список связных множеств точек на плоскости с целочисленными координатами (две точки принадлежат одному связному множеству, если максимум модуля разности координат не превышает единицы). Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление точки в список (в этом случае либо добавляется новое связное множество из одной точки, либо точка добавляется к одному из множеств списка, либо точка объединяет два или более множеств). Реализация списка – классическая ссылочная, с использованием «умных» указателей.

20. Реализовать на C++ однонаправленный список отрезков на прямой. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление отрезка в список (в этом случае либо добавляется новый отрезок, либо добавляемый отрезок объединяется с имеющимся в списке отрезком, либо отрезок объединяет два или более отрезка в один), удаление отрезка из списка (в этом случае либо со списком ничего не происходит, либо удаляется/обрезается один из отрезков списка, либо один из отрезков списка делится надвое). Реализация списка – классическая ссылочная, с использованием «умных» указателей.
21. Реализовать на C++ двунаправленный список отрезков на прямой. Необходимые методы: инициализация списка (конструктор), удаление списка (деструктор), добавление отрезка в список (в этом случае либо добавляется новый отрезок, либо добавляемый отрезок объединяется с имеющимся в списке отрезком, либо отрезок объединяет два или более отрезка в один), удаление отрезка из списка (в этом случае либо со списком ничего не происходит, либо удаляется/обрезается один из отрезков списка, либо один из отрезков списка делится надвое). Реализация списка – классическая ссылочная, с использованием «умных» указателей.