

## Лабораторная работа №5

### Шаблонные классы

#### Контрольные вопросы:

1. Что такое множественное наследование?
2. Что такое ромбовидное наследование и какие проблемы с этим связаны?
3. Что такое виртуальное наследование и для чего оно применяется в языке C++?
4. Что такое шаблоны и для чего они применяются?
5. Как объявить шаблон функции и шаблон класса?
6. Что такое инстанцирование шаблона и в какой момент оно происходит?
7. Какие преимущества и недостатки есть у шаблонов?
8. Что такое non-type параметры шаблона, какого типа они могут быть?
9. Что такое явная специализация шаблона и каким образом она объявляется?
10. Что такое частичная специализация шаблона и каким образом она объявляется?
11. Какое ограничение существует у частичной специализации шаблона и каким образом его можно обходить?
12. Чем характеризуется приведение типов с помощью `static_cast`?
13. Чем характеризуется приведение типов с помощью `dynamic_cast`?
14. Чем характеризуется приведение типов с помощью `const_cast` и `reinterpret_cast`?
15. Что такое `enum class` и в чём его отличие от обычного `enum`?

#### Задание

В данной лабораторной работе необходимо разработать шаблонный класс в соответствии с вариантом задания. Любой класс должен иметь конструктор копии и перегруженный оператор `=`. Также класс должен содержать перегрузку оператора `<<` для класса `iostream`, чтобы его можно было выводить в консоль через стандартный поток вывода `cout` с указанием количества элементов и их значений. В главной функции `main` обеспечить консольный интерфейс для тестирования всех функций шаблонного класса с типами `int`, `float`, `char*`, `struct Vec2 {float x; float y;}` (аналогично лаб 1)

#### Варианты

1. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.

Необходимые методы:

**`reserve(n)`** – зарезервировать ещё `n` элементов

**`getCapacity()`** – получить текущую ёмкость

**`getLength()`** – получить текущую длину

**`insert(elem,n)`** – добавить элемент на позицию `n`

**`remove(elem,n)`** – удалить элемент на позиции `n`

Перегрузить следующие операции:

**`+`** – добавить элемент в конец массива

**`-`** – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

**`[]`** – доступ к элементу по индексу

**`==` и `!=`** – проверка на идентичность и не идентичность массивов (для типа `char*` должна быть

своя специализация шаблона)

2. **Динамически-инициализированный массив**, количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.

Необходимые методы:

**sort(bAscending)** – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа `char*` должна быть своя специализация шаблона, сортирующая по длине строк.

**getMaxSize()** – получить максимальный размер массива

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

**+** – добавить элемент в конец массива

**-** – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

**[]** – доступ к элементу по индексу

**< и >** – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа `char*` и `Vec2` должна быть своя специализация шаблона: `char*` исходя из длины строки, для `Vec2` – исходя из длины вектора).

3. **Множество**, динамическая структура данных, где через non-type параметр типа `enum` задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**getSizeInBytes()** – получить текущий размер всей структуры в памяти в байтах (для типа `char*` должна быть своя специализация шаблона исходя из длины строки)

**getRepetitions(elem)** – получить текущее кол-во повторений элемента для мультимножества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:

**+** – добавить элемент в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**-** – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**[]** – проверить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**> и <** – проверить на подмножество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

4. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**unite(set)** – объединить с другим множеством.

Перегрузить следующие операции:

+ – добавить элемент в множество

- – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

\* – получить пересечение двух множеств (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

`==` и `!=` – проверка множеств на тождественность \ не тождественность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

5. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов `int` и `float` это модуль значения, для `char*` - длина строки, для `Vec2` – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в список (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

+ – добавить элемент в начало списка.

- – удалить все вхождения элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

-- – удалить первый элемент из начала списка.

[] – доступ к элементу на заданной позиции.

6. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

+ – добавить элемент в конец списка.

+ – добавить все элементы из другого списка.

- – удалить первое и последнее вхождение элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

[] – доступ к элементу на заданной позиции.

`==` и `!=` – проверка списков на идентичность и не идентичность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

7. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.

Необходимые методы:

**reserve(n)** – зарезервировать ещё `n` элементов

**getCapacity()** – получить текущую ёмкость

**getLength()** – получить текущую длину

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

**+** – добавить элемент в конец массива

**-** – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

**[]** – доступ к элементу по индексу

**==** и **!=** – проверка на идентичность и не идентичность массивов (для типа `char*` должна быть своя специализация шаблона)

8. **Динамически-инициализированный массив**, количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.

Необходимые методы:

**sort(bAscending)** – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа `char*` должна быть своя специализация шаблона, сортирующая по длине строк.

**getMaxSize()** – получить максимальный размер массива

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

**+** – добавить элемент в конец массива

**-** – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

**[]** – доступ к элементу по индексу

**< и >** – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа `char*` и `Vec2` должна быть своя специализация шаблона: `char*` исходя из длины строки, для `Vec2` – исходя из длины вектора).

9. **Множество**, динамическая структура данных, где через non-type параметр типа `enum` задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**getSizelnBytes()** – получить текущий размер всей структуры в памяти в байтах (для типа `char*` должна быть своя специализация шаблона исходя из длины строки)

**getRepetitions(elem)** - получить текущее кол-во повторений элемента для мультимножества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:

**+** – добавить элемент в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**-** – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**[]** – проверить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**> и <** – проверить на подмножество (для типа `char*` должна быть своя специализация

шаблона исходя из содержимого строки).

10. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в множество (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**unite(set)** – объединить с другим множеством.

Перегрузить следующие операции:

+ – добавить элемент в множество

- – удалить элемент из множества (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

\* – получить пересечение двух множеств (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки).

== и != – проверка множеств на тождественность \ не тождественность (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки).

11. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для char\* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в список (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в начало списка.

- – удалить все вхождения элемента из списка (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

-- – удалить первый элемент из начала списка.

[] – доступ к элементу на заданной позиции.

12. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец списка.

+ – добавить все элементы из другого списка.

- – удалить первое и последнее вхождение элемента из списка (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**[]** – доступ к элементу на заданной позиции.

**==** и **!=** – проверка списков на идентичность и не идентичность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

13. **Динамический массив**, использующий избыточное резервирование памяти под элементы.

Увеличивает ёмкость на количество элементов, заданных в non-type параметре.

Необходимые методы:

**reserve(n)** – зарезервировать ещё `n` элементов

**getCapacity()** – получить текущую ёмкость

**getLength()** – получить текущую длину

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

**+** – добавить элемент в конец массива

**-** – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

**[]** – доступ к элементу по индексу

**==** и **!=** – проверка на идентичность и не идентичность массивов (для типа `char*` должна быть своя специализация шаблона)

14. **Динамически-инициализированный массив**, количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.

Необходимые методы:

**sort(bAscending)** – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа `char*` должна быть своя специализация шаблона, сортирующая по длине строк.

**getMaxSize()** – получить максимальный размер массива

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

**+** – добавить элемент в конец массива

**-** – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

**[]** – доступ к элементу по индексу

**< и >** – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа `char*` и `Vec2` должна быть своя специализация шаблона: `char*` исходя из длины строки, для `Vec2` – исходя из длины вектора).

15. **Множество**, динамическая структура данных, где через non-type параметр типа `enum` задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**getSizelnBytes()** – получить текущий размер всей структуры в памяти в байтах (для типа `char*` должна быть своя специализация шаблона исходя из длины строки)

**getRepetitions(elem)** – получить текущее кол-во повторений элемента для мультимножества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:

- + – добавить элемент в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)
- – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)
- [] – проверить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)
- > и < – проверить на подмножество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

16. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**unite(set)** – объединить с другим множеством.

Перегрузить следующие операции:

- + – добавить элемент в множество
- – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)
- \* – получить пересечение двух множеств (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).
- == и != – проверка множеств на тождественность \ не тождественность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

17. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов `int` и `float` это модуль значения, для `char*` - длина строки, для `Vec2` – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в список (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

- + – добавить элемент в начало списка.
- – удалить все вхождения элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)
- – удалить первый элемент из начала списка.
- [] – доступ к элементу на заданной позиции.

18. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов  
**insert(elem,n)** – добавить элемент на позицию n  
**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец списка.

+ – добавить все элементы из другого списка.

- – удалить первое и последнее вхождение элемента из списка (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

[] – доступ к элементу на заданной позиции.

== и != – проверка списков на идентичность и не идентичность (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки).

19. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.

Необходимые методы:

**reserve(n)** – зарезервировать ещё n элементов

**getCapacity()** – получить текущую ёмкость

**getLength()** – получить текущую длину

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец массива

- – удалить первое вхождение элемента в массиве (для типа char\* должна быть своя специализация шаблона)

[] – доступ к элементу по индексу

== и != – проверка на идентичность и не идентичность массивов (для типа char\* должна быть своя специализация шаблона)

20. **Динамически-инициализированный массив**, количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.

Необходимые методы:

**sort(bAscending)** – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа char\* должна быть своя специализация шаблона, сортирующая по длине строк.

**getMaxSize()** – получить максимальный размер массива

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец массива

- – удалить первое вхождение элемента в массиве (для типа char\* должна быть своя специализация шаблона)

[] – доступ к элементу по индексу

< и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа char\* и Vec2 должна быть своя специализация шаблона: char\* исходя из длины строки, для Vec2 – исходя из длины вектора).



21. **Множество**, динамическая структура данных, где через non-type параметр типа enum задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**getSizeInBytes()** – получить текущий размер всей структуры в памяти в байтах (для типа char\* должна быть своя специализация шаблона исходя из длины строки)

**getRepetitions(elem)** – получить текущее кол-во повторений элемента для мультимножества (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:

**+** – добавить элемент в множество (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**-** – удалить элемент из множества (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**[]** – проверить вхождение элемента в множество (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**> и <** – проверить на подмножество (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки).

22. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в множество (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**unite(set)** – объединить с другим множеством.

Перегрузить следующие операции:

**+** – добавить элемент в множество

**-** – удалить элемент из множества (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**\*** – получить пересечение двух множеств (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки).

**== и !=** – проверка множеств на тождественность \ не тождественность (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки).

23. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов int и float это модуль значения, для char\* - длина строки, для Vec2 – длина вектора). Т.е. элементы с характеристикой, превышающей заданное значение нельзя добавить в список.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в список (для типа char\* должна быть своя специализация шаблона исходя из содержимого строки)

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

- + – добавить элемент в начало списка.
- – удалить все вхождения элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)
- – удалить первый элемент из начала списка.
- [] – доступ к элементу на заданной позиции.

24. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимое кол-во элементов в списке.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец списка.

+ – добавить все элементы из другого списка.

- – удалить первое и последнее вхождение элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

[] – доступ к элементу на заданной позиции.

**==** и **!=** – проверка списков на идентичность и не идентичность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

25. **Динамический массив**, использующий избыточное резервирование памяти под элементы. Увеличивает ёмкость на количество элементов, заданных в non-type параметре.

Необходимые методы:

**reserve(n)** – зарезервировать ещё n элементов

**getCapacity()** – получить текущую ёмкость

**getLength()** – получить текущую длину

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец массива

- – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

[] – доступ к элементу по индексу

**==** и **!=** – проверка на идентичность и не идентичность массивов (для типа `char*` должна быть своя специализация шаблона)

26. **Динамически-инициализированный массив**, количество элементов задаётся через non-type параметр, выделение памяти происходит в конструкторе при создании объекта.

Необходимые методы:

**sort(bAscending)** – сортировать элементы по возрастанию (**bAscending=true**), либо по убыванию. Для типа `char*` должна быть своя специализация шаблона, сортирующая по длине строк.

**getMaxSize()** – получить максимальный размер массива

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию n

**remove(elem,n)** – удалить элемент на позиции n

Перегрузить следующие операции:

+ – добавить элемент в конец массива

- – удалить первое вхождение элемента в массиве (для типа `char*` должна быть своя специализация шаблона)

[] – доступ к элементу по индексу

< и > – проверка, что в двух одинакового размера массивов все элементы больше или меньше элементов другого массива (для типа `char*` и `Vec2` должна быть своя специализация шаблона: `char*` исходя из длины строки, для `Vec2` – исходя из длины вектора).

27. **Множество**, динамическая структура данных, где через non-type параметр типа `enum` задаётся будет ли использоваться мультимножество (множество с повторениями элементов), либо простое множество (все элементы уникальны).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**getSizeInBytes()** – получить текущий размер всей структуры в памяти в байтах (для типа `char*` должна быть своя специализация шаблона исходя из длины строки)

**getRepetitions(elem)** – получить текущее кол-во повторений элемента для мультимножества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

Перегрузить следующие операции:

+ – добавить элемент в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

- – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

[] – проверить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

> и < – проверить на подмножество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

28. **Множество**, динамическая структура данных, где через non-type параметр задаётся максимально допустимый размер множества (больше этого кол-ва нельзя увеличить множество).

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в множество (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**unite(set)** – объединить с другим множеством.

Перегрузить следующие операции:

+ – добавить элемент в множество

- – удалить элемент из множества (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

\* – получить пересечение двух множеств (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

== и != – проверка множеств на тождественность \ не тождественность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).

29. **Односвязный список**, динамическая структура данных, где через non-type параметр задаётся максимально допустимая характеристика добавляемого элемента (для типов `int` и `float` это модуль значения, для `char*` - длина строки, для `Vec2` – длина вектора). Т.е. элементы с

характеристикой, превышающей заданное значение нельзя добавить в список.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**isExist(elem)** – получить вхождение элемента в список (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

**+** – добавить элемент в начало списка.

**-** – удалить все вхождения элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**--** – удалить первый элемент из начала списка.

**[]** – доступ к элементу на заданной позиции.

30. **Односвязный список**, динамическая структура данных, где через `non-type` параметр задаётся максимально допустимое кол-во элементов в списке.

Необходимые методы:

**getLength()** – получить текущее кол-во элементов

**insert(elem,n)** – добавить элемент на позицию `n`

**remove(elem,n)** – удалить элемент на позиции `n`

Перегрузить следующие операции:

**+** – добавить элемент в конец списка.

**+** – добавить все элементы из другого списка.

**-** – удалить первое и последнее вхождение элемента из списка (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки)

**[]** – доступ к элементу на заданной позиции.

**== и !=** – проверка списков на идентичность и не идентичность (для типа `char*` должна быть своя специализация шаблона исходя из содержимого строки).