Семинар #3: Класс `std::string`. Инициализация. Оператор `new`. Домашнее задание.

Задача 1. Измени регистр первой буквы

Напишите функцию, которая будет принимать на вход строку типа `std::string` и возвращать строку с изменённым регистром первой буквы. Например, если на вход пришла строка `"Cat"`, то функция должна вернуть строку `"cat"`. Если на вход пришла строка `"dog"`, то функция должна вернуть строку `"Dog"`. Если на вход пришла пустая строка, то функция должна вернуть простую строку.

Задача 2. Удвоение

Напишите функции, которые будут принимать на вход строку и возвращать строку, повторённую два раза. То есть, если на вход этой функции приходит строка `"Cat"`, то функция должна вернуть `"CatCat"`. При этом нужно написать несколько функций, которые должны делать одно и то же, но возвращать результат разными методами.

- `std::string repeat1(const std::string& s)` Должна принимать на вход строку и возвращать результат.
- `void repeat2(std::string& s)` Должна принимать на вход строку по ссылке и изменять эту строку.
- `void repeat3(std::string* ps)` Должна принимать на вход указатель на строку и изменять строку, чей адрес хранит этот указатель.
- `std::string* repeat4(const std::string& s)` Эта функция должна создавать удвоенную строку в куче с помощью оператора `new` и возвращать указатель на неё. После вызова функции `repeat4` программист, который будет использовать эту функцию, сам должен позаботиться об её удалении.

Протестируйте эти функции в `main`.

Задача 3. Умножение строки

Напишите перегруженный оператор умножения, которая будет принимать на вход строку `std::string` и некоторое целое число n и возвращать строку, повторённую n раз. Протестируйте эту функцию в функции `main`.

Задача 4. Усечение

Напишите функцию `void truncateToDot(std::string& s)`, которая будет принимать строку по ссылке и усекать её до первого символа точки. Размер и вместимость строки должны стать как можно более маленькими, для этого используйте метод `shrink_to_fit`. Функция не должна ничего возвращать. Протестируйте функцию в `main`.

до	после
`"cat.dog.mouse.elephant.tiger.lion"`	`"cat"`
`"wikipedia.org"`	`"wikipedia"`
`".com"`	`""`

Задача 5. Сумма из строки

Напишите функцию, которая принимает на вход строку в следующем формате: `"[num1, num2, ... numN]"`. Функция должна возвращать целое число типа `int` – сумму всех чисел в квадратных скобках. В случае, если на вход приходит некорректная строка, то функция должна бросать исключение `std::invalid_argument`. Протестируйте эту функцию в `main`.

аргумент	возвращаемое значение
`"[10, 20, 30, 40, 50]"`	150
`"[4, 8, 15, 16, 23, 42]"`	108
`"[20]"`	20
`"[]"`	0

Задача 6. `new`

Используйте операторы `new` или `new[]`, чтобы создать в куче и сразу инициализировать следующие объекты:

- Один объект типа `int`, равный `123`.
- Один объект типа `std::string`, равный `"Cats and Dogs"`.
- Массив объектов типа `int`, равный `{10, 20, 30, 40, 50}`.
- Массив объектов типа `std::string`, равный `{"Cat", "Dog", "Mouse"}`.
- Массив из 3-х объектов типа `std::string_view`, указывающих на строки из предыдущего пункта.

Все эти объекты обязательно должны быть созданы в куче, а не на стеке. Напечатайте все созданные объекты на экран. Удалите все созданные объекты с помощью операторов `delete` и `delete[]`.

Задача 7. Кошки-мышки

Есть некоторое количество групп кошек, которые соревнуются в том, кто поймает больше мышек. Соревнование командное, побеждает та группа, которая суммарно поймала больше мышек.

Ваша задача - написать класс `Cat`, который оказался бы полезным для автоматизации подсчёта пойманных мышек. Сумарное количество пойманых мышек для каждой группы будет хранится вне класса `Cat`, например в локальной переменной (так как кошки из других групп не должны иметь доступ к этой переменной). В самом же классе должна храниться *ссылка* на эту переменную. В классе вам нужно написать следующие методы:

- Конструктор. Должен конструироваться от переменной, в которой будет храниться суммарное количество мышек.
- Метод `catchMice` - поймать мышек. Принимает число и увеличивает суммарно количество пойманых мышек данной группы на это число.

Пример использования такого класса (код ниже должен работать и с вашим классом):

```
int miceCaughtA = 0;
int miceCaughtB = 0;

Cat alice(miceCaughtA), alex(miceCaughtA), anna(miceCaughtA);
Cat bob(miceCaughtB), bella(miceCaughtB);

alice.catchMice(2);
alex.catchMice(1);
bella.catchMice(4);
bob.catchMice(2);
anna.catchMice(1);
bella.catchMice(1);
alex.catchMice(4);
bella.catchMice(5);
alice.catchMice(2);

cout << miceCaughtA << endl; // Должно напечатать 10
cout << miceCaughtB << endl; // Должно напечатать 12
```

Задача 8. Создание `mipt::String`

В файле `miptstring.hpp` содержится класс `mipt::String`. Создайте объекты этого класса в следующим образом:

- Создайте объект класса `mipt::String` на стеке обычным образом и инициализируйте его строкой `Cat`.
- Создайте объект класса `mipt::String` в куче, используя оператор `new`, и инициализируйте его строкой `Dog`. Напечатайте этот объект на экран. Удалите этот объект, с помощью оператора `delete`.

- Пусть у нас есть массив:

  ```
  char x[sizeof(mipt::String)];
  ```

  Создайте объект класса `mipt::String` в этом массиве с помощью оператора `placement new` и инициализируйте объект строкой `Elephant`. Напечатайте этот объект на экран. Удалите этот объект.

Необязательные задачи

Задача 1. `StringView`

Создайте свой класс `mipt::StringView`, аналог класса `std::string_view` для класса `mipt::String`. Этот класс должен содержать 2 поля указатель `mpData` (тип `const char*`) и размер `mSize` (тип `size_t`). Класс `mipt::String` можно найти в файле `miptstring.hpp`.

Методы, которые нужно реализовать:

- Конструктор по умолчанию. Должен устанавливать указатель в `nullptr`, а размер в 0.
- Конструктор копирования.
- Конструктор от `mipt::String`.
- Конструктор от `const char*`
- Перегруженный `operator[]`
- Метод `at`, аналог `operator[]`, но если индекс выходит за границы, то данный метод должен бросать исключение `std::out_of_range`
- Перегруженный `operator<`
- Перегруженный `operator<<` с объектом `std::ostream`.
- Метод `size`.
- Метод `substr`, должен возвращать объект типа `std::string_view`.
- Методы `remove_prefix` и `remove_suffix`.

Также придётся изменить класс `mipt::String`. Нужно будет добавить конструктор от `mipt::StringView`. Протестируйте этот класс.