Лабораторная работа II. Сетка.

Задача: создать шаблон класса для моделирования двумерной сетки из произвольных объектов. Сетка должна автоматически управлять памятью.

Основная идея.

В некоторой системе для моделирования сетки произвольных величин задан следующий шаблон класса:

```
template <typename T>
class Grid final {
public:
    using value_type = T;
    using size_type = unsigned;
private:
    T * const data;
    size_type const y_size, x_size;

    Grid(T *data, size_type y_size, size_type x_size):
        data(data), y_size(y_size), x_size(x_size) {  }

    Grid(Grid<T> const &) = delete;
    Grid(Grid<T>&&) = delete;
    Grid<T>& operator=(Grid<T>&) = delete;
    Grid<T>& operator=(Grid<T>&&) = delete;

    T operator()(size_type y_idx, size_type x_idx) const {
        return data[y_idx * x_size + x_idx];
    }

    T& operator()(size_type y_idx, size_type x_idx) {
        return data[y_idx * x_size + x_idx];
    }

    Grid<T>& operator=(T const &t) {
        for (
            auto it = data, end = data + x_size * y_size;
            it != end; ++it
         ) *it = t;
     return *this;
    }

```

```
34      size_type get_y_size() const { return y_size; }
35      size_type get_x_size() const { return x_size; }
36  };
```

Шаблон `Grid<typename T>` моделирует двумерную сетку состоящую из `y_size` рядов по `x_size` ячеек в ряду, в каждой ячейке содержится элемент типа `T`. Обратите внимание, что все методы копирования и перемещения явно удалены. Добавлен только один специальный метод, который копирует переданный по ссылке объект типа `T` в каждую ячейку сетки.

Задание 1. RAII. (4 балла)

Модифицируйте класс `Grid` таким образом, чтобы он мог самостоятельно управлять собственным ресурсом памяти, а именно: реализуйте необходимые методы согласно «правилу пяти». Дополните шаблон класса следующими конструкторами:

1. конструктор `Grid(T const &t)` с одним параметром для неявного преобразования типов (из `T` в `Grid<T>`) создаёт новую сетку размером 1×1 с единственным элементом - копией `t`;

2. конструктор с двумя параметрами `Grid(size_type y_size, size_type x_size)` создаёт сетку размером $y\_size \times x\_size$, заполненную элементами типа `T`, созданными конструктором по умолчанию (default initialized);

3. конструктор с тремя параметрами `Grid(size_type y_size, size_type x_size, T const &t)` создаёт сетку размером $x\_size \times y\_size$, заполненную копиями объекта `t`;

Обратите внимание, что сетка должна успешно работать с объектами, которые не имеют конструктора без параметров. В этом случае мы ожидаем, что код, использующий наш класс (клиентский код) будет вызывать конструкторы 1 или 3. Для объектов `T` предполагается `copy-constructible` и `copy-assignable`.

Задание 2. Оператор индексирования. (3 балла)

Добавьте в класс сетки оператор индексирования таким образом, чтобы при обращении к сетке с помощью двойного применения оператора квадратных скобок мы могли получить элемент, хранящийся в сетке. Причём следующий код должен успешно выполняться:

```
#include <cassert>
int main() {
    Grid<float> g(3, 2, 0.0f);
    assert(3 == g.get_y_size());
    assert(2 == g.get_x_size());

    using  gsize_t = Grid<float>::size_type;

     for (gsize_t y_idx = 0; y_idx != g.get_y_size(); ++y_idx)
        for (gsize_t x_idx = 0; x_idx != g.get_x_size(); ++x_idx)
            assert(0.0f == g[y_idx][x_idx]);

    for (gsize_t y_idx = 0; y_idx != g.get_y_size(); ++y_idx)
        for (gsize_t x_idx = 0; x_idx != g.get_x_size(); ++x_idx)
            g[y_idx][x_idx] = 1.0f;

    for (gsize_t y_idx = 0; y_idx != g.get_y_size(); ++y_idx)
        for (gsize_t x_idx = 0; x_idx != g.get_x_size(); ++x_idx)
            assert(1.0f == g(y_idx, x_idx));
    return 0;
}
```

Задание 3. Многомерные сетки. (3 балла)

С развитием системы возникла необходимость оперировать не только двумерными, но и сетками более высоких размерностей. Реализуйте шаблон класса для работы с сетками произвольных размерностей. Шаблон класса должен зависеть от двух параметров: типа хранящихся элементов и размерности пространства сетки. Для хранящихся в сетке элементов можно предполагать `copy-constructible` и `copy-assignable`. Следующий пример кода должен успешно срабатывать:

```
Grid<float,3> const g3(2, 3, 4, 1.0f);
assert(1.0f == g3(1, 1, 1));

Grid<float,2> g2(2, 5, 2.0f);
assert(2.0f == g2(1, 1));

g2 = g3[1];
assert(1.0f == g2(1, 1));
```

Пояснение. Конструктор на первой строке принимает не два, а три целых параметра, так как размерность пространства сетки равна 3. Оператор круглых скобок также принимает три параметра, так как `g3` имеет размерность 3. Оператор индексирования (квадратные скобки), возвращает ссылку или копию (в зависимости от константности) сетки меньшей размерности. Оператор индексирования для сетки размерности 1 (линия) возвращает ссылку или копию хранящегося элемента. Элементы, хранящиеся, в сетке, должны допускать конструирование копированием и присваивание копированием и не обязаны быть перемещаемыми, для самой же сетки также должно быть допустимо перемещение.