

## Лабораторная работа по теме «Управление ресурсами».

Рассмотрим шаблон класса **Grid**, который моделирует двумерную сетку из произвольных объектов, допускающих конструирование копированием (copy-constructible) и копирующее присваивание (copy-assignable):

```
1 template <typename T>
2 class Grid final {
3 public:
4     using value_type = T;
5     using size_type = unsigned;
6 private:
7     T * const data;
8     size_type const y_size, x_size;
9
10    Grid(T *data, size_type y_size, size_type x_size):
11        data(data), y_size(y_size), x_size(x_size) { }
12
13    Grid(Grid<T> const &) = delete;
14    Grid(Grid<T>&&) = delete;
15    Grid<T>& operator=(Grid<T>&) = delete;
16    Grid<T>& operator=(Grid<T>&&) = delete;
17
18    T operator()(size_type y_idx, size_type x_idx) const {
19        return data[y_idx * x_size + x_idx];
20    }
21
22    T& operator()(size_type y_idx, size_type x_idx) {
23        return data[y_idx * x_size + x_idx];
24    }
25
26    Grid<T>& operator=(T const &t) {
27        for (
28            auto it = data, end = data + x_size * y_size;
29            it != end; ++it
30        ) *it = t;
31        return *this;
32    }
33
34    size_type get_y_size() const { return y_size; }
35    size_type get_x_size() const { return x_size; }
36};
```

В существующем виде класс не управляет ресурсом памяти самостоя-

тельно. Память выделяется внешним кодом и освобождена должна быть внешним кодом.

## RAII

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  $1 \times 1$  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 конструктор (3).

## Оператор индексирования

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:

```
1 #include <cassert>
2 int main() {
3     Grid<float> g(3, 2, 0.0f);
4     assert(3 == g.get_y_size());
5     assert(2 == g.get_x_size());
6 }
```

```

7      using gsize_t = Grid<int>::size_type;
8
9      for (gsize_t y_idx = 0; y_idx != g.get_y_size(); ++y_idx)
10         for (gsize_t x_idx = 0; x_idx != g.get_x_size(); ++x_idx)
11             assert(0.0f == g[y_idx][x_idx]);
12
13     for (gsize_t y_idx = 0; y_idx != g.get_y_size(); ++y_idx)
14         for (gsize_t x_idx = 0; x_idx != g.get_x_size(); ++x_idx)
15             g[y_idx][x_idx] = 1.0f;
16
17     for (gsize_t y_idx = 0; y_idx != g.get_y_size(); ++y_idx)
18         for (gsize_t x_idx = 0; x_idx != g.get_x_size(); ++x_idx)
19             assert(1.0f == g(y_idx, x_idx));
20     return 0;
21 }

```

## Многомерные сетки

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 сетки. Хранящиеся элементы являются copy-constructible и copy-assignable. Следующий пример кода должен успешно срабатывать:

```

1  Grid<float,3> const g3(2, 3, 4, 1.0f);
2  assert(1.0f == g3(1, 1, 1));
3
4  Grid<float,2> g2(2, 5, 2.0f);
5  assert(2.0f == g2(1, 1));
6
7  g2 = g3[1];
8  assert(1.0f == g2(1, 1));

```

**Пояснение.** Как и в предыдущих задачах, оператор круглых скобок принимает ровно то количество аргументов, какова размерность сетки. 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 присваива-

ние копированием, но для самой сетки также должно быть допустимо перемещение.