Q*. Array

Ограничение времени	1 секунда
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод
Вывод	стандартный вывод

https://gitlab.com/ibr11/cpp_psami_base/-/tree/main/array

C-style массивы эффективней в использовании чем массивы в динамической памяти, из-за известного на этапе компиляции фиксированного размера и более эффективного расположения в памяти (стековая/автоматическая память). Однако едва ли С-массивы можно назвать удобными в использовании - они спонтанно приводятся к указателям, их не так просто передать в функцию, не работает привычная операция присваивания и т.д. В этой задаче предлагается написать ООП обертку над C-style массивом, который обладает всеми его преимуществами, нивелируя при этом описанные недостатки.

std::array (C++11)
Начиная с C++11 в стандартной библиотеке (заголовочный файл `<array>`) появился шаблонный класс `std::array<T, N>`, параметризованный типом хранимых элементов `T` и размером `N`. Этот класс содержит единственное поле типа "массив `T` из `N` элементов" и предоставляет методы для удобного доступа к информации (размер, элементы и т.д), а также изменения элементов массива. Класс `std::array` в полной мере обеспечивает функционал обычных массивов, не теряя при этом эффективности. Поэтому в современном C++ принято использовать именно `std::array` вместо C-style массивов. Подробнее.

Детали реализации
Вам необходимо реализовать шаблонный класс `Array` - упрощенный аналог `std::array`. Ваш класс должен быть параметризован типом хранимых элементов, а также размером массива. Как было сказано ранее, этот класс должен иметь ровно одно поле - C-style массив требуемого типа и размера. Важным моментом является то, что это поле должно быть публичным! В этом случае, как известно, становится доступна агрегатная инициализация вашего массива:

```
int arr[4]{1, 2, 3};  // массив {1, 2, 3, 0}
Array<int, 4> my_arr{1, 2, 3};  // если поле публично, то {1, 2, 3, 0}, иначе - CE
```

Для корректного прохождения тестов ваш класс должен реализовывать следующие методы (подумайте, какие методы должны быть константными, какие - неконстантными, а какие должны иметь обе версии):

- Операция `[]` для доступа к элементу массива по индексу;
- Метод `At(size_t idx)`, который обеспечивает безопасный (с проверкой границ) доступ к элементу по индексу. В случае выхода за границы необходимо бросать исключение типа `ArrayOutOfRange`, которое уже написано за вас (см. замечания);
- Методы `Front()` и `Back()` для доступа к первому и последнему элементу соответственно;
- Метод `Data()`, возвращающий указатель на начало массива;
- Методы `Size()` и `Empty()`;
- Метод `Fill(const T& value)`, который заполняет массив значениями `value`;
- Метод `Swap(Array<T, N>& other)`, обменивающий содержимое массивов одинакового размера.

Замечания.

1. Проверка устроена таким образом, что требует от вас жесткого следования принятым (выше) сигнатурам и именованиям сущностей (то есть никакие `MyArray`, `__array_`, `back`, `superSolver3000` не пройдут). Если вы реализовали требуемый функционал не полностью или интерфейс отличается от заявленного, в ответ вы получите ошибку компиляции.
2. Решение должно состоять из одного файла `array.h` с определением класса.
3. В задаче нет скрытых тестов - все тесты публичные (`array_public_test.cpp`).
4. "Бросить исключение типа `E`" значит - написать строку `throw E{};`. В этот момент выполнение функции прекращается и, если исключение не будет обработано, программа завершится аварийно. Тестирующий код корректно обработает эту ошибку, вам этого делать не нужно.

Дополнительная часть

С помощью механизма перегрузки шаблонов реализуйте функции для извлечения свойств C-style массивов:

- `GetSize(array)` должна возвращать число элементов в массиве `array` и 0, если `array` не является C-style массивом
- `GetRank(array)` должна возвращать число координат многомерного массива `array`
- `GetNumElements(array)` должна возвращать *общее* число элементов в многомерном массиве `array`

Пример

```
int x;
int a[3];
int b[3][2][1];

std::cout << GetSize(x) << ' ' << GetSize(a) << ' ' << GetSize(b) << '\n';  // 0 3 3
std::cout << GetRank(x) << ' ' << GetRank(a) << ' ' << GetRank(b) << '\n';  // 0 1 3
std::cout << GetNumElements(x) << ' ' << GetNumElements(a) << ' ' << GetNumElements(b) << '\n';  // 1 3 6
```

Важно: если вы выполнили это задание, добавьте в файл `array.h` следующую строку, чтобы в тесты была включена проверка этого задания.

```
#define ARRAY_TRAITS_IMPLEMENTED
```