**array**

Моей задачей является реализации аналога std::array – массива статической длины (в отличие от vector)

1. lab1.hpp file contains MyArray class implementation
2. lab1\_test.cpp file contains unit tests usint GoogleTest for MyArray
3. wrapper.cpp file contains Wrapper for MyArray
4. Makefile: make test - run Google Test for MyArray
5. To start wrapper: make make wrap

ДОКУМЕНТАЦИЯ array Моей задачей является реализации аналога std::array – массива статической длины (в отличие от vector) Исходя из задания, я буду реализовывать свой класс MyArray. Класс шаблонный, как и стандартный массив. Методы существующего массива взяты с cppreference.com. Логично предположить, что методы аналога должны быть аналогичны. (<https://en.cppreference.com/w/cpp/container/array>) Класс MyArray template <class T, size\_t N> Методы, которые будут реализованы:

1. Constructor – конструктор MyArray(){}; 1.2 Constructor, принимающий список инициализации MyArray (std::initializer\_list list){};
2. Operator[] - обращение к элементу в указанной позиции, возвращаем data\_[i]; Value\_type& operator[](size\_t i) {};
3. Operator at() - обращение к элементу в указанной позиции, возвращаем data\_[i]; Value\_type& at(int i) {};
4. Empty – проверка, что size\_ == 0 bool empty(){};
5. Size – размер (количество элементов) (size\_) size\_t size() {};
6. Begin – начало итератора +const iterator iterator begin() {};
7. End - конец итератора +const iterator iterator end() {};
8. rBegin – наоборот +const iterator reverseIterator rbegin() {};
9. rEnd - наоборот +const iterator reverseIterator rend() {};
10. Вывод в стандартный поток (friend функция) (пользуемся итератором и выводим каждый элемент) friend std::ostream& operator<< (std::ostream& out, const MyArray& arr);
11. Operator== - оператор равенства (по элементам) (проходим итератором, сравниваем элементы) bool operator==(const MyArray& other){};
12. Operator!= - оператор неравенства (по элементам) (аналогично п.10) bool operator!=(const MyArray& other){};
13. Operator> - оператор больше (по сумме элементов) bool operator>(const MyArray& other){};
14. Operator< - оператор меньше (по сумме элементов) bool operator<(const MyArray& other){};
15. Operator>= - оператор больше либо равно (по сумме элементов) bool operator>=(const MyArray& other){};
16. Operator<= - оператор меньше либо равно (по сумме элементов) bool operator<=(const MyArray& other){};
17. front() – указатель на первый элемент +const value\_type& front() {};
18. back() – указатель на последний элемент +const value\_type& back() {};
19. получение размера массива на этапе компиляции constexpr size\_t size() {}

Поля класса:

1. data\_ - статический массив типа value\_type

Обертка: Изначально будет создано 5 пустых массивов: MyArray<int, MAX> arr = {0, 0, 0, 0, 0, 0}; //пример создания массива - пусть работаем с типом int

Вспомогательные классы:

1. iterator – итератор прямого порядка с поддержанием возможности обратного порядка и рандомного индекса
   1. iterator(pointer ptr)- конструктор, принимающий указатель
   2. reference operator\*() const – оператор непрямого обращения
   3. pointer operator->()- оператор для получения члена структуры
   4. iterator& operator++() – префиксный инкремент для прямого порядка
   5. iterator operator++(int) – постфиксный инкремент для прямого порядка
   6. iterator& operator--() – префиксный декремент для обратного порядка
   7. iterator operator--(int) – постфиксный декремент для обратного порядка
   8. iterator operator[](int n) const – метод для рандомного доступа
   9. friend bool operator== (const iterator& a, const iterator& b) – оператор равенства
   10. friend bool operator!= (const iterator& a, const iterator& b) – оператор неравенства
2. reverseIterator – итератор обратного порядка с поддержанием возможности прямого порядка. Ниже прямым порядком называется обратный и наоборот
   1. reverseIterator(pointer ptr)- конструктор, принимающий указатель
   2. reference operator\*() const – оператор непрямого обращения
   3. pointer operator->()- оператор для получения члена структуры
   4. reverseIterator& operator++() – префиксный инкремент для прямого порядка
   5. reverseIterator operator++(int) – постфиксный инкремент для прямого порядка
   6. reverseIterator & operator--() – префиксный декремент для обратного порядка
   7. reverseIterator operator--(int) – постфиксный декремент для обратного порядка
   8. reverseIterator operator[](int n) const – метод для рандомного доступа
   9. friend bool operator== (const reverseIterator & a, const reverseIterator & b) – оператор равенства
   10. friend bool operator!= (const reverseIterator & a, const reverseIterator & b) – оператор неравенства
3. Также поддерживается опция константных итераторов
   1. typedef const iterator constIterator
   2. typedef const reverseIterator constReverseIterator

Размер определен макросом #define MAX 6

Будет реализована в виде цикла while с switch case – цифрами, идентифицирующими операцию:

1. Изменить элемент (ввести номер массива, ввести позицию, ввести новое значение)
2. Сравнить 2 массива == (ввести номер первого массива, ввести номер второго массива)
3. Сравнить 2 массива != (ввести номер первого массива, ввести номер второго массива)
4. Сравнить 2 массива >= (ввести номер первого массива, ввести номер второго массива)
5. Сравнить 2 массива <= (ввести номер первого массива, ввести номер второго массива)
6. Сравнить 2 массива >(ввести номер первого массива, ввести номер второго массива)
7. Сравнить 2 массива < (ввести номер первого массива, ввести номер второго массива)
8. Распечатать элемент массива (ввести номер массива, ввести позицию)
9. Проверить массив на пустоту (ввести номер массива)
10. Распечатать размер массива (ввести номер массива)
11. Распечатать первый элемент массива front() (ввести номер массива)
12. Распечатать первый элемент массива back() (ввести номер массива)
13. Распечатать массив (ввести номер массива)
14. Создать массив (ввести размер) массивы будут создаваться с именами arr+N, где N – порядковый номер, создать можно не более 5 массивов //!не поддерживается
15. Выйти из программы