std::array library

#include <array>

(there exists also std::vector)

https://en.cppreference.com/w/cpp/container/array

<https://www.learncpp.com/cpp-tutorial/6-15-an-introduction-to-stdarray/>

**Member functions**

|  |  |
| --- | --- |
| **Implicitly-defined member functions** | |
| (constructor)  (implicitly declared) | initializes the array following the rules of [aggregate initialization](https://en.cppreference.com/w/cpp/language/aggregate_initialization) (note that default initialization may result in indeterminate values for non-class T)  (public member function) |
| (destructor)  (implicitly declared) | destroys every element of the array  (public member function) |
| **operator=**  (implicitly declared) | overwrites every element of the array with the corresponding element of another array  (public member function) |
| **Element access** | |
| [**at**](https://en.cppreference.com/w/cpp/container/array/at) | access specified element with bounds checking  (public member function) |
| [**operator[]**](https://en.cppreference.com/w/cpp/container/array/operator_at) | access specified element  (public member function) |
| [**front**](https://en.cppreference.com/w/cpp/container/array/front) | access the first element  (public member function) |
| [**back**](https://en.cppreference.com/w/cpp/container/array/back) | access the last element  (public member function) |
| [**data**](https://en.cppreference.com/w/cpp/container/array/data) | direct access to the underlying array  (public member function) |
| **Iterators** | |
| [**begin cbegin**](https://en.cppreference.com/w/cpp/container/array/begin) | returns an iterator to the beginning  (public member function) |
| [**end cend**](https://en.cppreference.com/w/cpp/container/array/end) | returns an iterator to the end  (public member function) |
| [**rbegin crbegin**](https://en.cppreference.com/w/cpp/container/array/rbegin) | returns a reverse iterator to the beginning  (public member function) |
| [**rend crend**](https://en.cppreference.com/w/cpp/container/array/rend) | returns a reverse iterator to the end  (public member function) |
| **Capacity** | |
| [**empty**](https://en.cppreference.com/w/cpp/container/array/empty) | checks whether the container is empty  (public member function) |
| [**size**](https://en.cppreference.com/w/cpp/container/array/size) | returns the number of elements  (public member function) |
| [**max\_size**](https://en.cppreference.com/w/cpp/container/array/max_size) | returns the maximum possible number of elements  (public member function) |
| **Operations** | |
| [**fill**](https://en.cppreference.com/w/cpp/container/array/fill) | fill the container with specified value  (public member function) |
| [**swap**](https://en.cppreference.com/w/cpp/container/array/swap) | swaps the contents |

------------------------------------------------------------------

include <array>

#include <iostream>

int main()

{

    std::array myArray { 7, 3, 1, 9, 5 };

    // std::array<int, 5>::size\_type is the return type of size()!

    for (std::array<int, 5>::size\_type i{ 0 }; i < myArray.size(); ++i)

        std::cout << myArray[i] << ' ';

    std::cout << '\n';

    return 0;

}

---------------------------------------------------------------------------------------

#include <array>

#include <iostream>

 struct House

{

    int number{};

    int stories{};

    int roomsPerStory{};

};

 int main()

{

    std::array<House, 3> houses{};

    houses[0] = { 13, 4, 30 };

    houses[1] = { 14, 3, 10 };

    houses[2] = { 15, 3, 40 };

    for (const auto& house : houses)

    {

        std::cout << "House number " << house.number

                  << " has " << (house.stories \* house.roomsPerStory)

                  << " rooms\n";

    }

    return 0;

Output

House number 13 has 120 rooms

House number 14 has 30 rooms

House number 15 has 120 rooms