

Arrays

Solutions

- Describe what is meant by a `std::array` in C++
 - `std::array` represents a fixed-size block of memory. Elements can be accessed using indexes or iterators
- Give one example of how `std::array` differs from `std::vector`
 - Number of elements is fixed and must be known in advance

- Convert the code below into a full working program. At the end of the program, print out the elements of a and b
- Experiment with printing and assigning different elements until you are confident with it

```
std::array<int, 5> a {1, 2, 3, 4, 5};  
cout << a[3];  
a[2] = 6;  
std::array<int, 5> b;  
b = a;
```

- Are the following statements legal C++? If they are legal, what will happen when they are executed?
 - All are legal C++, but some will cause undefined behaviour

```
std::array<int, 5> a {1, 2, 3, 4, 5}; // OK
cout << a[-2];                       // Undefined behaviour (not element in array)
a[20] = 0;                           // Undefined behaviour (not element in array)
array<int, 5> c;                      // OK
cout << c[3];                        // Undefined behaviour (element not initialized)
```
- Write an equivalent program which uses `std::vector`. Does it behave differently from the `std::array` program?

```
int main() {  
    vector<int> a {1, 2, 3, 4, 5};  
    cout << a[-2] << endl;  
    a[20] = 0;  
    vector<int> c(5);  
    cout << c[3] << endl;  
}
```

- Both versions have arbitrary behaviour
- With `std::vector`, we can call the `at()` member function to do a range check on the index

```
cout << a.at(-2) << endl;           // Throws an exception if the index is out of range
```

- Give some advantages of `std::array` compared to `std::vector`
 - Lower overhead
 - Does not perform internal memory allocations on the heap
 - No exceptions
 - Allocated on the stack

- Give some disadvantages of `std::array` compared to `std::vector`
 - Number of elements must be known at compile time
 - `std::array` instances with different number and/or type of elements are not compatible with each other
 - No provision for range checking
 - Allocating a very large array on the stack causes the stack to overflow