

Write-only Algorithms

Solutions

- What does fill() do?
 - fill() assigns a given value to the elements in an iterator range
- What arguments does fill() take?
 - The iterator range and the value
- What is the return value of fill()?
 - void
- Write a simple program that uses fill() to populate a vector
- Write the equivalent code without using fill()

- What does `fill_n()` do?
 - `fill_n` assigns a given value to a fixed number `n` of elements
- What arguments does `fill_n()` take?
 - An iterator to the first element, the number `n` and the value
- What is the return value of `fill()`?
 - An iterator to the element after the last element assigned
- Write a simple program using `fill_n()` to populate a vector
- Write the equivalent code without using `fill_n()`

- What problems can occur when using `fill_n` with an uninitialized container?
 - The container may have less than 'n' elements
 - `fill_n` assigns without checking whether the element is valid
 - Undefined behaviour

- Write a program which executes the code below and runs it

```
vector<int> v;  
fill_n(v.begin(), 20, 6);
```

- What happens and why?
 - The program writes to “elements” which do not belong to the vector
 - The program will probably crash due to a memory access violation

- Write an equivalent of `fill_n()` which uses a loop and does not write past the end of the container

```
for (int i = 0; i < 20; ++i)  
    vec.push_back(6);
```

- How does this avoid the problem of writing to non-existent elements?
 - The call to `push_back()` creates a new element, which then has 6 assigned to it
 - The element always exists when it is written to

- Explain what `back_inserter()` does
 - `back_inserter()` creates an insert iterator
 - Every time we assign to the insert iterator, a new element is created
- How does `back_inserter()` solve the problem from the last slide?
 - The insert iterator calls `push_back()` internally
 - The element will always exist when it is written to

- Rewrite your program from the previous slide to use `back_inserter()` and run it. What happens?
 - The program runs as expected

- What does generate() do?
 - generate() assigns the elements in an iterator range to the return value from calling a function
- What arguments does generate() take?
 - The iterator range and the function to be called
- Write a simple program that uses generate() to populate a vector
- Write the equivalent code without using generate()

- What does `generate_n()` do?
 - `generate_n()` assigns a fixed number of elements to the return value from calling a function
- What arguments does `generate_n()` take?
 - An iterator to the first element, the number `n` and the function
- Write a simple program that uses `generate_n()` to populate a vector
- Write the equivalent code without using `generate_n()`