Shared Pointers and Arrays

C++ 17 has come up with a new method of handling arrays.

WE'LL COVER THE FOLLOWING

Before C++17

Only `unique_ptr` was able to handle arrays out of the box (without the need to define a custom deleter).

After C++17

Now it's also possible with `shared_ptr` as shown below:

```
std::shared_ptr<int[]> ptr(new int[10]);
```

> Please note that `std::make_shared` doesn't support arrays in C++17. However, this will be fixed in C++20 (see P0674 which is already merged into C++20)

Another important remark is that raw arrays should be avoided. It's usually better to use standard containers. However, sometimes, you don't have the luxury to use vectors or lists - for example:

- in an embedded environment
- or when you work with third-party API

In that situation, you might end up with a raw pointer to an array. With C++17, you'll be able to wrap those pointers into *smart pointers* (`std::unique_ptr` or

`std::shared_ptr`) and be sure the memory is deleted correctly.

> **Extra Info:** See the initial proposal: P0414R2.

Next up, we'll tackle three new generic functions which have been introduced for data structures.