C4GC: gsl::span

YAIR FRIEDMAN

What we will see

- My №1disliked C++ feature
- Guideline F.24
- Caller view
  - Live example
- Calee view
  - Live example
- Summary

My № 1disliked C++ feature

T*

- Who owns the memory? (gsl::owner<T> and gsl::observer<T>)
- Can it be null? (gsl::not_null<T> and std::optional<T>)
- Is it just a pointer or a decayed array?
- If it is an array:
  - `(T* vp, size_t size)`
  - Size errors cannot be checked at compile time, or at runtime

F.24: Use a span<T> or a span_p<T> to designate a half-open sequence


```
void f(int* p, size_t n);
int a[100];
// ...
f(a, 100);
f(a, 1000); // BAD
```

F.24: Use a span<T> to designate a half-open sequence


```
#include <gsl/gsl>
void f(gsl::span<int>arr);
int a[100];
// ...
f(a);
// OR
f(gsl::span<int>{a});
// OR EVEN
f(gsl::span<int, 100>{a});
```

span_p<T> // {p, predicate} [p:q) where q is the first element for which predicate(*p) is true

From the calee side

```
void f(int* p, size_t n) {
    // P can be null?
    //if (!p) return;
   for (auto i=0u ; i<n ;i++) {
      std::cout << *(p+i) << ' ';
   }
}

int a[100];
// ...
f(a, 100); // OK
f(NULL, 10); // BAD
f(NULL, 0); // OK???
```

At the callee side


```
#include <gsl/gsl>
void f(gsl::span<int>arr) {
    for (const auto x : arr) { // RANGE
        std::cout << x << ' ';
    }
    for (auto i=gsl::index{0}; i<arr.size(); i++) {// C-Style and [] access
        std::cout << i << ':' << arr[i] << ' ';
    }
    for (auto it = std::begin(arr); it != std::end(arr); it++) {// Iterators
        std::cout << *it << ' ';
    }
    // Every loop is an algorithm
    std::for_each(std::begin(arr), std::end(arr), [](const int x) {
        std::cout << x << ' ';
    });
}
```

Summary

- Basically just a `struct {T* const ptr; size_t size;}`

	std::vector	**std::array**	**gsl::span**
Heap allocation	Yes	No	No
Ownership Semantics	Owns its contents	Owns its contents	Non-owning
Cost of Copying	Expensive*	Expensive	Cheap
Passing Style	By Reference	By Reference	By Value
Element Mutability	Yes	Yes	Yes

Questions?

Sources

(Kate Gregory)

https://engineering.tamu.edu/media/3627488/no-littering-tamu.pdf

https://github.com/isocpp/CppCoreGuidelines/blob/master/docs/gsl-intro.md

https://www.youtube.com/watch?v=qjxBKINAWk0