C++ Core Guidelines

Quick 10 minutes talk

By Shalom Kramer @kramerpeace

Why?

- Less arguments
- Safety
- Use new features
- Show how to use new features
- Introduces new libraries

- CppCon 2017: Kate Gregory “10 Core Guidelines You Need to Start Using Now”

Structure

- I is for interface, F is for functions
- Rule, reason, suggestions, how to check it

Where can I find it?

- Official docs - Git Hub – REALLY not user friendly
- Microsoft Checker - unusable
- Microsoft GSL – needs compilation
- Gsl-lite – one file header.

span<T>

- Pointer and length
- Random Access
- Iterators
- Represents a *view*
- Non owner (as opposed to vector)

I.24: Avoid adjacent unrelated parameters of the same type

```
void copy_n(T* p, T* q, int n);  // we can easily swap p and q

// better

void copy_n(const T* p, T* q, int n);

// nirvana

void copy_n(span<const T> p, span<T> q);
```

I.13: Do not pass an array as a single pointer

```
// lots of problems
void copy_n(const T* p, T* q, int n); // copy from [p:p+n) to [q:q+n)
// solves most issues since span has a size()
void copy(span<const T> r, span<T> r2); // copy r to r2
```

I.23: Keep the number of function arguments low

```
void draw(Shape* p, int n);  // poor interface; poor code

Circle arr[10];

// there can be quite a few lines here..

draw(arr, 10);

// less error prone

void draw2(span<Circle>);

Circle arr[10];

draw2(arr);
```

P.5: Prefer compile-time checking to run-time checking

```
void read(int* p, int n);   // read max n integers into *p

int a[100];

read(a, 1000);    // will fall on run time


// alternative

void read(span<int> r); // read into the range of integers r

int a[100];

read(a);        // better: let the compiler figure out the number of elements
```

Thank You!

Questions?