Bridging the Language Gap
Classes for C++/Fortran Interoperability

Ivan Pribec, ivan.pribec@lrz.de

Enhanced C interoperability with **Fortran 2018**

```
interface
  subroutine process_array(a) bind(c)
    real(c_double), intent(out) :: a(:)
  end subroutine
end interface
```

```
#include <ISO_Fortran_binding.h>
void process_array(CFI_cdesc_t *a)
{
  double *a_ptr = a->base_addr;
  for (int i = 0; i < a->dim[0].extent; i++){
    *a_ptr = (double) i;
    a_ptr += a->dim[0].sm;
  }
}
```

C++ Adaptor

```
template<typename T, int rank = 1>
class cdesc_ptr {
public:
  cdesc_ptr(CFI_cdesc_t *a) : a_(a) { ... }

        T& operator[](size_t idx) { ... }
  const T& operator[](size_t idx) const { ... }

  T* begin() { ... }
  T*   end() { ... }

private:
  CFI_cdesc_t *a_{nullptr};
};

extern "C"
void process_array(CFI_cdesc_t *a_f) {
  cdesc_ptr<double> a(a_f);
  std::iota(a.begin(),a.end());
}
```

Run-time type and rank checking

Element access

Iterator access, range-based for loops

Reduced boiler-plate
Use of STL algorithms