Overloading

Core Guidelines

Having different names for logically equivalent operations on different argument types is confusing, leads to encoding type information in function names, and inhibits generic programming.

C.162: Overload operations that are roughly equivalent

Mismatch with C

```
void pericombobulate_i16(int16_t x);
void pericombobulate_i32(int32_t x);
void pericombobulate_f32(float x);
void pericombobulate_f64(double x);
```

```
template<typename T>
void pericombobulate(const T& x)
{
    ?
}
```

Goal

```
const auto pericombobulate = overloads(
    pericombobulate_i16,
    pericombobulate_i32,
    pericombobulate_f32,
    pericombobulate_f64);

pericombobulate(42.0f);
```

overloads

```
template<typename... Funcs>
auto overloads(Funcs&&... funcs)
{


}
```

overload_set

```
template<typename... Funcs>
struct overload_set : Funcs...
{
    explicit overload_set(Funcs&&... funcs)
        : Funcs{std::forward<Funcs>(funcs)}... {}

    using Funcs::operator()...;
};

template<class... Funcs> overload_set(Funcs...) -> overload_set<Funcs...>;
```

https://godbolt.org/z/YynxhF

std::function_ref

```
template<typename... Funcs>
auto overloads(Funcs&&... funcs)
{
    return overload_set(tl::function_ref{std::forward<Funcs>(funcs)}...);
}
```

https://github.com/TartanLlama/function_ref

https://godbolt.org/z/tQQcgF


SCRAP GP