Lako do N-arnih funkcija

Petar Trifunovic, 29.12.2022

Motivacija

```
bind_front(f, a1, a2, a3, a4)

bind_back(f, a9, a8, a7, a6)

compose(f, g, h, l, m, n)

first_of(f, g, h, l, m, n)
```

Trenutno stanje

```
template<typename Func, typename... FrontArgs>
inline auto bind_front(Func &&f, FrontArgs &&...args)
{
   return
       [HFTECH_CAPTURE(f), binds = hftech::makeTuple(HFTECH_FWD(args)...)]  //
       (auto &&...backArgs) mutable
       -> std::invoke_result_t<Func, FrontArgs..., decltype(backArgs)...> {
           return std::apply(
               HFTECH_FWD_CAPTURED(f),
               std::tuple_cat(
                   binds, hftech::makeTuple(HFTECH_FWD(backArgs)...)));
       };
}
```

Lenjost

```
template<typename Callable, typename First>
struct BindFirst
{
    [[no_unique_address]] Callable callable{};
    [[no_unique_address]] First first{};

    auto operator()(auto &&...args)
        HFTECH_RETURNS(callable(first, HFTECH_FWD(args)...));
};
```

Posledica

```
bind_first(f, a1)

bind_last(f, a9)

compose_two(f, g)

first_of_two(f, g)
```

Resenje

```
template<typename Callable, typename First>
struct BindFirst
{
    [[no_unique_address]] Callable callable{};
    [[no_unique_address]] First first{};

    auto operator()(auto &&...args)
        HFTECH_RETURNS(callable(first, HFTECH_FWD(args)...));
};

HFTECH_GEN_LEFT_N_ARY(BindFront, BindFirst);
```

Magician Implementacija

```
template<
    template<typename...>
    typename BinaryMetaOp,
    typename First,
    typename... Rest>
struct LeftReduce : public First
{};

template<
    template<typename...>
    typename BinaryMetaOp,
    typename First,
    typename Second,
    typename... Rest>
struct LeftReduce<BinaryMetaOp, First, Second, Rest...>
    : public LeftReduce<BinaryMetaOp, BinaryMetaOp<First, Second>, Rest...>
{};
```

```
#define HFTECH_GEN_LEFT_N_ARY(NAME, OP)                                  \
    template<typename First, typename... Rest>                           \
    struct NAME : public First                                           \
    {};                                                                  \
                                                                         \
    template<typename First, typename Second, typename... Rest>          \
    struct NAME<First, Second, Rest...>                                  \
        : public NAME<OP<First, Second>, Rest...>                        \
    {};                                                                  \
                                                                         \
    template<typename... F>                                              \
    NAME(F &&...) -> NAME<F...>;
```

https://godbolt.org/z/4WYYT53Tx

Hvajaa

Nadam se da cemo od sada svi ziveti lepo i srecno sa n-arnim funkcijamaa