10.10 Функция std::invoke, ее предназначение и идея реализации. Зачем нужна эта функция? Реализация метафункции std::invocable<F, Args...>

Invoke - вызвать функцию. Пример использования:

```
int main() {
    std::function<bool(int, int)> compare = [](int x, int y) { return x < y;
    };
    cout << std::invoke(compare, 1, 2);
    //output 1;
}
```

std::invoke нужен чтобы единым образом вызвать функторы (в т.ч. лямбды), указатели на функции и указатели на функции-члены классов. Последние имеют специфический синтаксис вызова не совпадающий с синтаксисом обычных функторов:

```
(obj->*mem_fn_ptr)( args... );
```

Если вам нужно реализовать поддержку всех callable объектов, то проще передать их одним пакетом в invoke, чем писать отдельную шаблонную реализацию для каждого случая. Такой синтаксический сахар.

Реализация метафункции std::invocable<F, Args...>

```
namespace detail {
template<class>
constexpr bool is_reference_wrapper_v = false;
template<class U>
constexpr bool is_reference_wrapper_v<std::reference_wrapper<U>> = true;

template<class C, class Pointed, class T1, class... Args>
constexpr decltype(auto) invoke_memptr(Pointed C::* f, T1&& t1, Args&&...
    args)
{
    if constexpr (std::is_function_v<Pointed>) {
        if constexpr (std::is_base_of_v<C, std::decay_t<T1>>)
            return (std::forward<T1>(t1).*f)(std::forward<Args>(args)...);
        else if constexpr (is_reference_wrapper_v<std::decay_t<T1>>)
            return (t1.get().*f)(std::forward<Args>(args)...);
        else
            return ((*std::forward<T1>(t1)).*f)(std::forward<Args>(args)...)
    ;
    } else {
        static_assert(std::is_object_v<Pointed> && sizeof...(args) == 0);
        if constexpr (std::is_base_of_v<C, std::decay_t<T1>>)
            return std::forward<T1>(t1).*f;
        else if constexpr (is_reference_wrapper_v<std::decay_t<T1>>)
            return t1.get().*f;
        else
            return (*std::forward<T1>(t1)).*f;
    }
}
} // namespace detail

template<class F, class... Args>
constexpr std::invoke_result_t<F, Args...> invoke(F&& f, Args&&... args)
    noexcept(std::is_nothrow_invocable_v<F, Args...>)
{
```

```
33      if constexpr (std::is_member_pointer_v<std::decay_t<F>>)
34          return detail::invoke_memptr(f, std::forward<Args>(args)...);
35      else
36          return std::forward<F>(f)(std::forward<Args>(args)...);
37  }
```

Идея реализаций: расписываем вручную все варианты, и в зависимости от этого выбираем нужную сигнатуру