Introduction to modern C++

Olve Maudal

C++ has evolved a lot since it was first introduced as "C with classes" with primitive support for object-oriented programming. In particular during the last 10-15 years the common use of the language has changed "dramatically" and the language itself has evolved accordingly. Modern C++ (C++11/14) is still very suitable for object-oriented programming, but now the language also provides good support for generic programming and functional programming. All of this while C++ is still a low-level language that can be used to create programs that compete with programs written in assembler both in terms of speed and size.

We start with a brief history of C++ before focusing on new features in C++11/14 and a demonstration of some typical modern programming techniques.

a 30 minute presentation for Finn
Oslo, Mar 18, 2015

- Brief History of C and C++
- New features in C++11/14
- Generic Programming
- The future of C++

Brief History of C and C++

http://www.aftenposten.no/fakta/innsikt/Da-skiene-ble-vinger-7459947.html

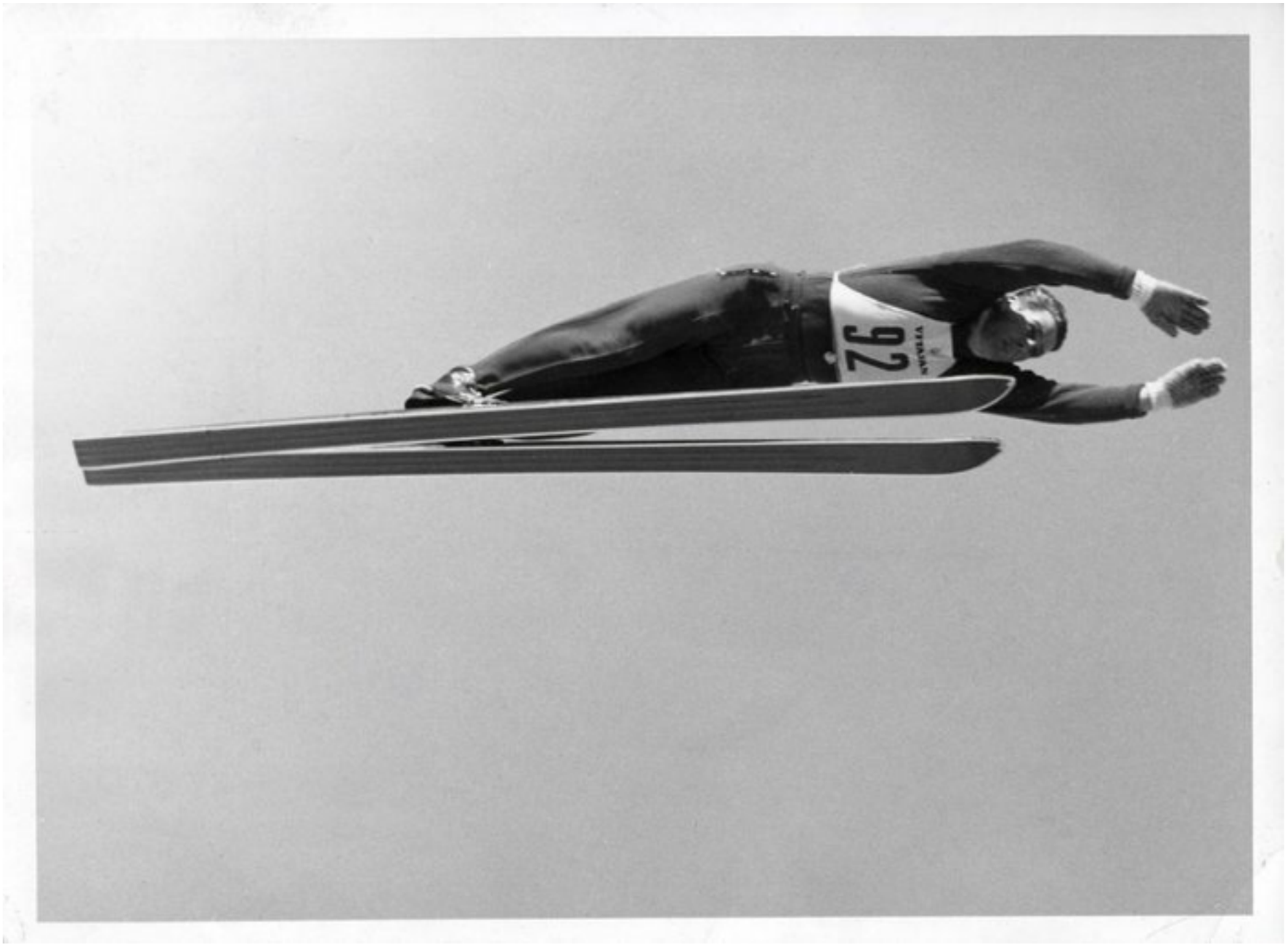
40's
Machine code, symbol tables and Assembler

50's
Fortran, Lisp, Cobol, Algol

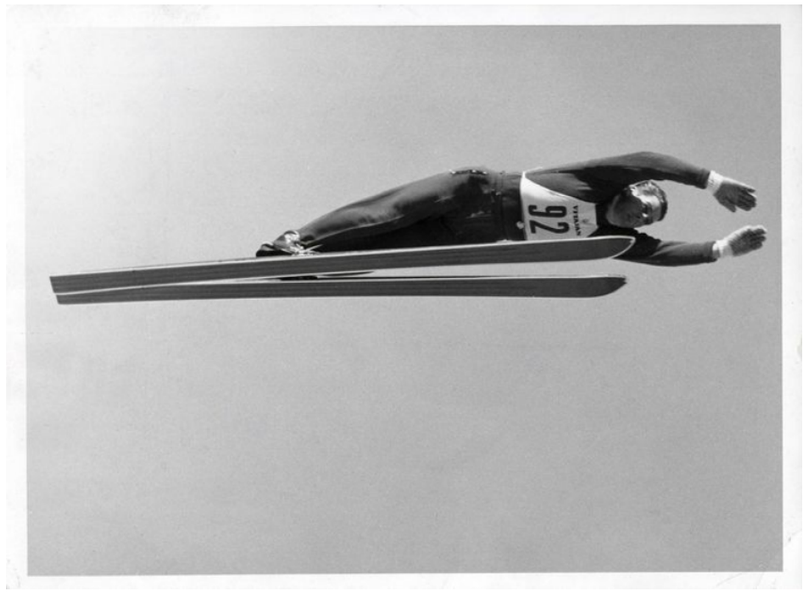
60's

many, many languages appeared in this period. In particular...

CPL



60's
... and Simula



Both CPL and Simula were examples of **very** elegant languages...

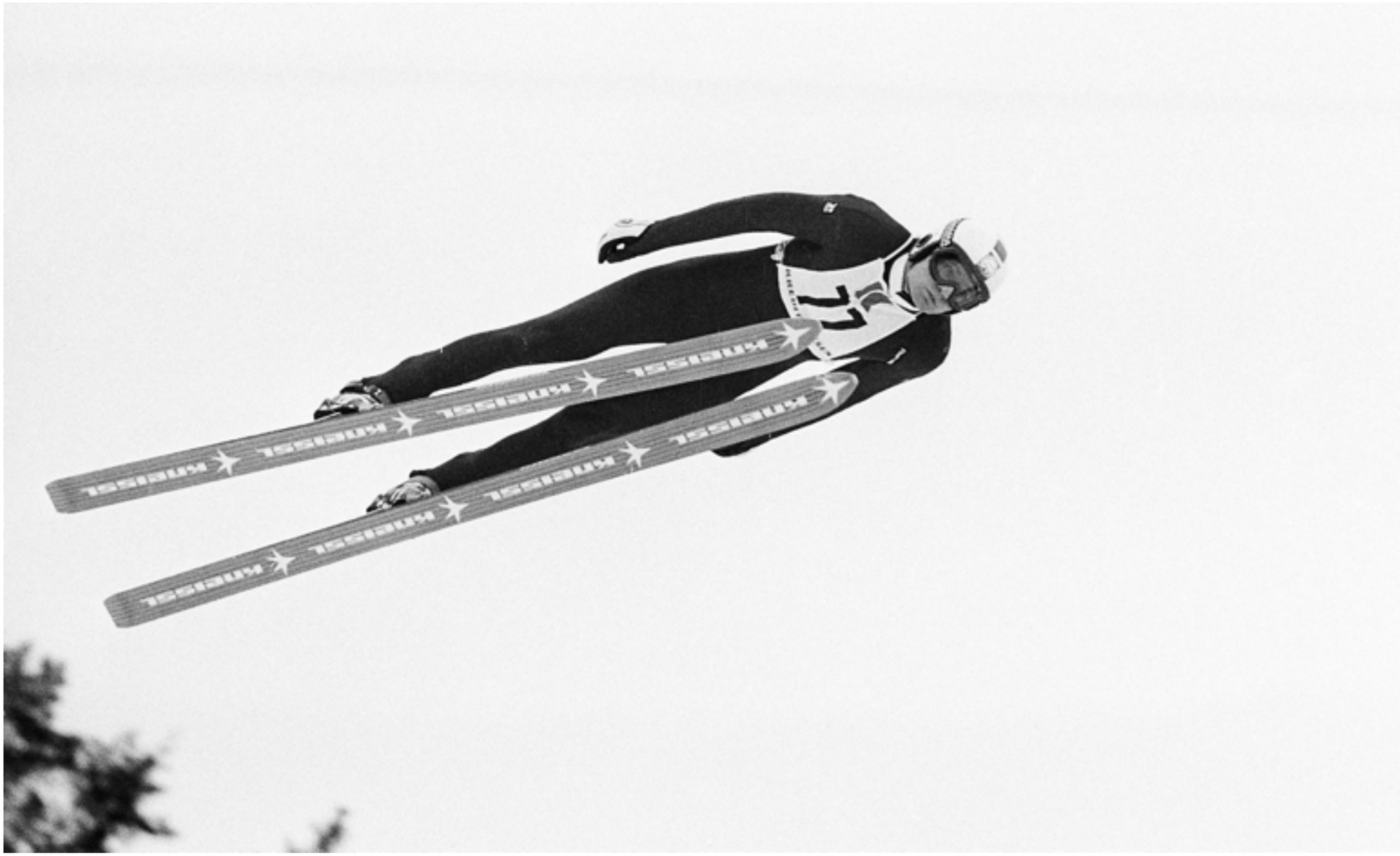
... but there was also a need for brutally efficient languages

... but there was also a need for brutally efficient languages

70's
BCPL, B, C

Bjarne Stroustrup combined the efficiency of C with some of the elegance from Simula...

80's
C with classes, C++/CFront, ARM



C++ was improved and became standardized

90's
X3J16, C++arm, WG21, C++98, STL

=

+

+ $\hbar$

C++ BCPL Simula

Ouch... Template Metaprogramming

C++03, TR1, Boost and other external libraries

While the language itself saw some minor improvements after C++98, Boost and other external libraries acted like laboratories for experimenting with potential new C++ features. Resulting in...

C++11/C++14

With the latest version C++ feels like a new language

The future of C++?

Modern C++ by Example

unless specified otherwise, all these code snippets should compile cleanly with a modern C++ compiler

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (std::vector<int>::const_iterator it = log.cbegin();
         it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (std::vector<int>::const_iterator it = log.cbegin();
         it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
$ g++-4.9 -std=c++1y -Wall -Wextra -pedantic -Werror foo.cpp && ./a.out
20
24
37
42
23
45
37
$
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (std::vector<int>::const_iterator it = log.cbegin();
         it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (decltype(log.cbegin()) it = log.cbegin();
         it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (auto it = log.cbegin();
         it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (auto it = log.cbegin();
                                  it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (auto it = log.cbegin(); it != log.cend(); ++it)

        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (auto it = log.cbegin(); it != log.cend(); ++it)
        transmit_item(*it);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (auto i : log)
        transmit_item(i);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>


static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    for (auto i : log)
        transmit_item(i);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);

}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{

    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(const std::vector<int> & log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> & log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(log);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}




static void transmit_log(std::vector<int> && log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static bool mycomp(int lhs, int rhs)
{
    return lhs > rhs;
}

static void transmit_log(std::vector<int> && log)
{
    std::sort(std::begin(log), std::end(log), mycomp);
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static bool mycomp(int lhs, int rhs)
{
    return lhs > rhs;
}

static void transmit_log(std::vector<int> && log, bool comp(int, int))
{
    std::sort(std::begin(log), std::end(log), comp);
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), mycomp);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static bool mycomp(int lhs, int rhs)
{
    return lhs > rhs;
}

static void transmit_log(std::vector<int> && log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log)
{
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>


static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log)
{



    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log));
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log, int limit)
{
    std::function<bool (int)> myfilter = [limit](int i) { return i <= limit; };
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), 23);
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log, std::function<bool (int)> myfilter)
{

    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static void transmit_log(std::vector<int> && log, std::function<bool (int)> myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Filt>
static void transmit_log(std::vector<int> && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>


static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log), transmit_item<int>);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<int>);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{

    std::vector<int> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](int i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::deque<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

template <typename T>
static void transmit_item(T i)
{
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>


template <typename T>
static void transmit_item(T i)
{

    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <type_traits>

template <typename T>
static void transmit_item(T i)
{
    static_assert(std::is_integral<T>::value, "integral type expected");
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <type_traits>


template <typename T>
static void transmit_item(T i)
{
    static_assert(std::is_integral<T>::value, "integral type expected");
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <type_traits>
#include "mystuff"

template <typename T>
static void transmit_item(T i)
{
    static_assert(my::is_transmittable<T>::value, "transmittable type expected");
    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <type_traits>
#include "mystuff"

template <typename T> require Transmittable<T>
static void transmit_item(T i)
{

    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <type_traits>
#include "mystuff"

template <Transmittable T>
static void transmit_item(T i)
{

    std::cout << i << std::endl;
    // ...
}

template <typename Log, typename Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>
#include <type_traits>
#include "mystuff"

template <Transmittable T>
static void transmit_item(T i)
{

    std::cout << i << std::endl;
    // ...
}

template <Iterable Log, UnaryFunctionPredicate Filt>
static void transmit_log(Log && log, Filt myfilter)
{
    log.erase(std::remove_if(std::begin(log), std::end(log), myfilter),
              std::end(log));
    std::sort(std::begin(log), std::end(log));
    std::for_each(std::begin(log), std::end(log),
                  transmit_item<typename Log::value_type>);
}

int main()
{
    using log_item_type = long;
    std::vector<log_item_type> log{20,24,37,42,23,45,37};
    transmit_log(std::move(log), [](log_item_type i) { return i <= 23; });
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    size_t items = transmit_log(log);
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>



static void transmit_item(int i)
{
    std::cout << i << std::endl;

    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    size_t items = transmit_log(log);
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    size_t items = transmit_log(log);
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>


static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    size_t items = transmit_log(log);

    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);
    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);
    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

```
$ g++-4.9 -std=c++1y -Wall -Wextra -pedantic -Werror -pthread foo.cpp
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);
    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);




    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);
    for (int i=0; i<5; i++) {
        std::this_thread::sleep_for(std::chrono::milliseconds(77));
        std::cout << "do something else..." << std::endl;
    }
    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);
    for (int i=0; i<5; i++) {
        std::this_thread::sleep_for(std::chrono::milliseconds(123));
        std::cout << "do something else..." << std::endl;
    }
    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

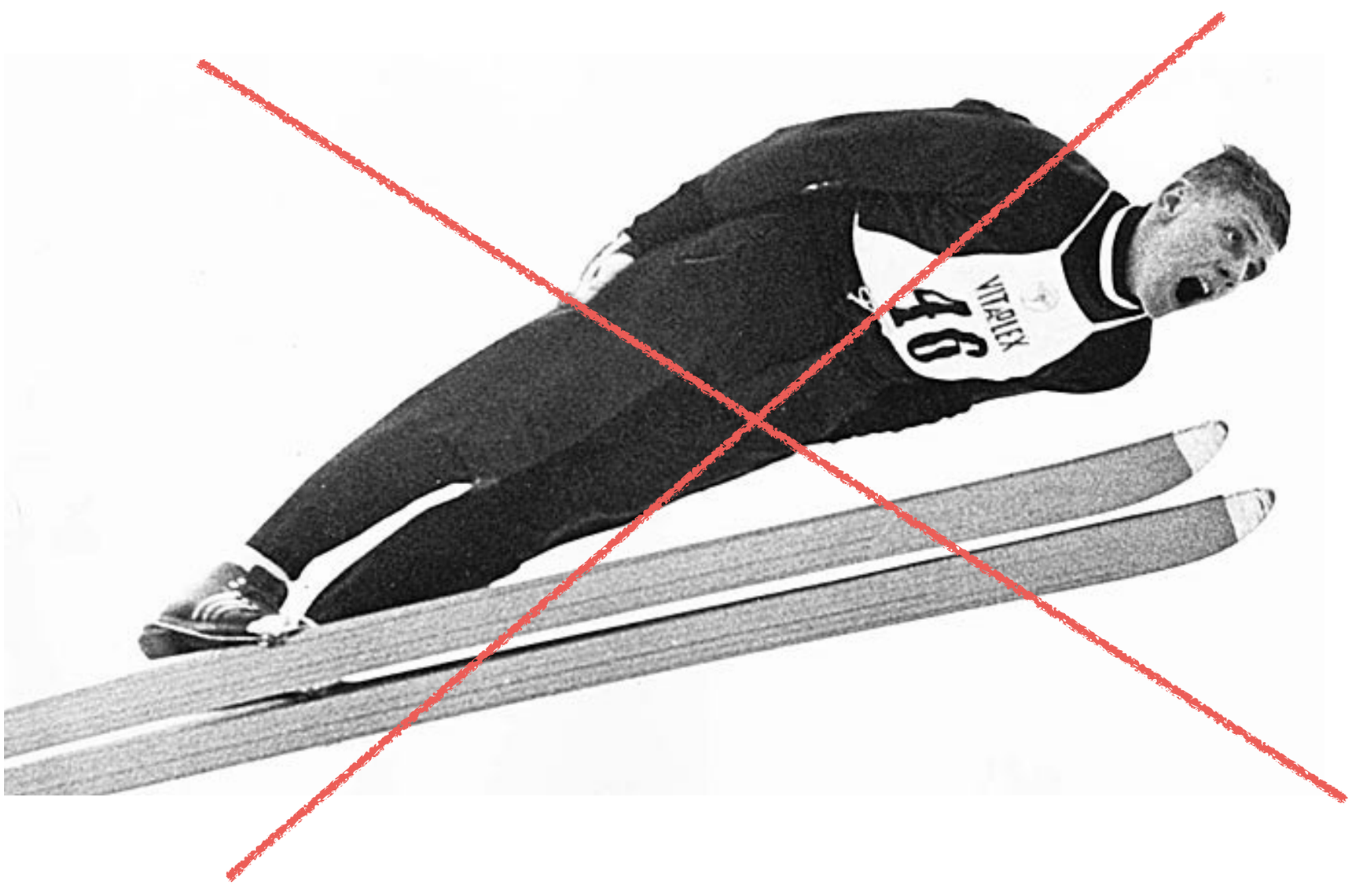
```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>
#include <thread>
#include <future>

static void transmit_item(int i)
{
    std::cout << i << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(200));
    // ...
}

static size_t transmit_log(const std::vector<int> & log)
{
    std::for_each(std::begin(log), std::end(log), transmit_item);
    return log.size();
}

int main()
{
    std::vector<int> log{20,24,37,42,23,45,37};
    auto res = std::async(std::launch::async, transmit_log, log);
    for (int i=0; i<5; i++) {
        std::this_thread::sleep_for(std::chrono::milliseconds(123));
        std::cout << "do something else..." << std::endl;
    }
    size_t items = res.get();
    std::cout << "# " << items << std::endl;
}
```

```
20
do something else...
24
do something else...
do something else...
37
do something else...
42
do something else...
23
45
37
# 7
```

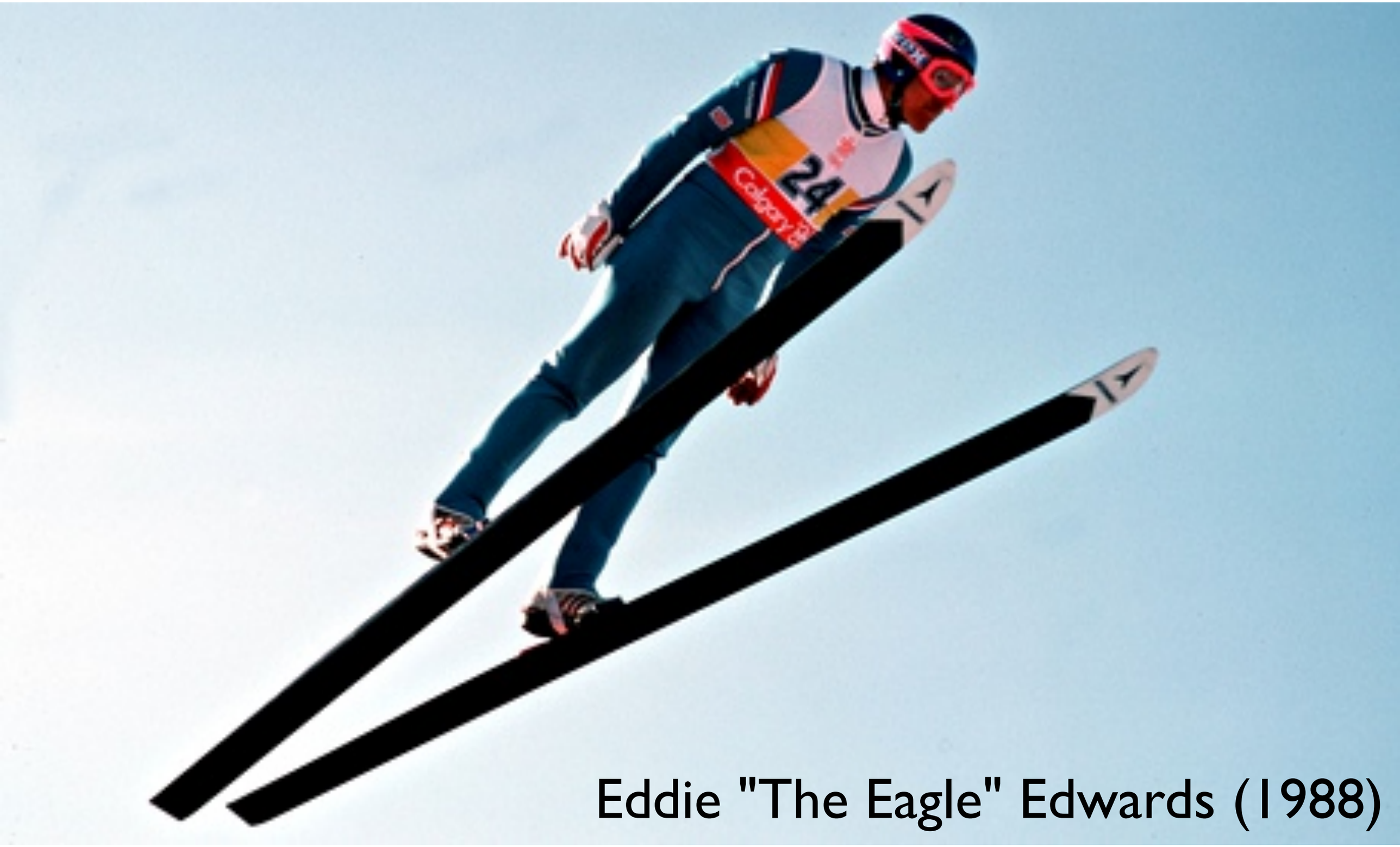
Modern C++
- move semantics (rvalue references, value semantics)
- type deduction (decltype, auto)
- better support for OOP (attributes, member initialization, delegation)
- compile time computation (templates, static_assert, constexpr)
- template metaprogramming (traits, constraints, concepts)
- robust resource management (RAII, unique, shared)
- high-order parallelism (atomic, mutex, async, promises and futures)
- functional programming (algorithms, lamdas, closures, lazy evaluation)
- misc (chrono, user-defined literals, regex, uniform initialization)

The success of C++ can also be meassured in that it has inspired a lot of other programming languages. Java, C#, Scala, D, Go, Swift just to mention a few

The success of C++ can also be meassured in that it has inspired a lot of other programming languages. Java, C#, Scala, D, Go, Swift just to mention a few



The success of C++ can also be meassured in that it has inspired a lot of other programming languages. Java, C#, Scala, D, Go, Swift just to mention a few



Eddie "The Eagle" Edwards (1988)

!