CAPHYON ⚡ LIGHTNING TALKS

The beginning of the end for [begin, end)

July, 2019
Craiova


@ciura_victor

Victor Ciura
Technical Lead, Caphyon
www.caphyon.ro

A functional language is one that supports and encourages the `functional style`

What do you mean ?

True Story

1986:

Donald Knuth was asked to implement a program for the *"Programming pearls"* column in the **Communications of ACM** journal.

The task:

Read a file of text, determine the n most frequently used words, and print out a sorted list of those words along with their frequencies.

His solution written in `Pascal` was **10 pages** long.

True Story

Doug McIlroy

His response was a 6-line shell script that did the same:

```
tr -cs A-Za-z '\n' |
    tr A-Z a-z |
    sort |
    uniq -c |
    sort -rn |
    sed ${1}q
```

It's all about pipelines

Taking inspiration from Doug McIlroy's UNIX shell script,

write an algorithm in `your favorite programming language`,

that solves the same problem: word frequencies

C++ 20 Ranges

```
[ iterator, sentinel )

Iterator:
*i  // access the value
++i // move to the next element

Sentinel:
i == s  // has iterator reached the end
```

C++ 20 Ranges

New algorithms

Many iterator adaptors

Views

Pipelines

Actions

Lazy evaluation

Projections

Very efficient generated code

New namespace: `std::ranges`

C++ 20 Ranges

"The beginning of the end for [begin, end)"

Jeff Garland

C++ 20 Ranges

Until the new ISO standard lands in a compiler near you...

Eric Niebler's implementation of the Ranges library is available here:
https://github.com/ericniebler/range-v3

It works will **Clang** 3.6.2 or later, **gcc** 5.2 or later, and **MSVC** 15.9 or later.

```
#include<range/v3/all.hpp>
namespace rs = ranges::v3;
namespace rv = ranges::v3::view;
namespace ra = ranges::v3::action;
```

C++ 20 Ranges

Print only the **even** elements of a range in **reverse** order:

```
std::for_each(
   std::crbegin(v), std::crend(v),
   [](auto const i) {
      if(is_even(i))
         cout << i;
   });
```

```
for (auto const i : v
                  | rv::reverse
                  | rv::filter(is_even))
{
   cout << i;
}
```

C++ 20 Ranges

Skip the first **2** elements of the range and print only the **even** numbers of the **next 3** in the range:

```
auto it = std::cbegin(v);
std::advance(it, 2);
auto ix = 0;
while (it != cend(v) && ix++ < 3)
{
   if (is_even(*it))
      cout << (*it);
   it++;
}
```

```
for (auto const i : v
                  | rv::drop(2)
                  | rv::take(3)
                  | rv::filter(is_even))
{
   cout << i;
}
```

C++ 20 Ranges

Modify an *unsorted* range so that it retains only the **unique** values but in **reverse** order.

```
vector<int> v{ 21, 1, 3, 8, 13, 1, 5, 2 };

std::sort(std::begin(v), std::end(v));

v.erase(
   std::unique(std::begin(v), std::end(v)),
   std::end(v));

std::reverse(std::begin(v), std::end(v));
```

```
vector<int> v{ 21, 1, 3, 8, 13,
1, 5, 2 };

v = std::move(v) |
    ra::sort |
    ra::unique |
    ra::reverse;
```

C++ 20 Ranges

Create a range of ***strings*** containing the **last 3** numbers **divisible to 7** in the range **[101, 200]**, in **reverse** order.

```
vector<std::string> v;

for (int n = 200, count = 0;
     n >= 101 && count < 3; --n)
{
   if (n % 7 == 0)
   {
      v.push_back(to_string(n));
      count++;
   }
}
```

```
auto v = rs::iota_view(101, 201)
       | rv::reverse
       | rv::filter([](auto v) { return v%7==0; })
       | rv::transform(to_string)
       | rv::take(3)
       | rs::to_vector;
```

It's all about pipelines

Taking inspiration from Doug McIlroy's UNIX shell script,

write an algorithm in `your favorite programming language`,

that solves the same problem: word frequencies

Word Frequencies

```
const auto words =
      istream_range<std::string>(std::cin)
    | view::transform(string_to_lower)
    | view::transform(string_only_alnum)
    | view::remove_if(&std::string::empty)
    | ranges::to_vector | action::sort;
```

Word Frequencies

```
const auto results = words
  | view::group_by(std::equal_to())
  | view::transform([] (const auto & group) {
        const auto b = std::begin(group);
        const auto e = std::end(group);
        const auto size = std::distance(b, e);
        const std::string word = *b;
        return make_pair(size, word);
      })
  | ranges::to_vector | action::sort;
```

Word Frequencies

```
for (auto value : results | view::reverse
                          | view::take(n))
{
  std::cout << value.first << ": " << value.second << "\n";
}
```

CAPHYON ⚡ LIGHTNING TALKS

The beginning of the end for [begin, end)

July, 2019
Craiova


@ciura_victor

Victor Ciura
Technical Lead, Caphyon
www.caphyon.ro