Maximum N elements

Pieter P

C++ 20

C++ 20 Ranges Implementation

```cpp
#include <algorithm> // ranges::partial_sort_copy
#include <ranges>    // views::iota, ranges::forward_range, ranges::borrowed_iterator_t
#include <vector>    // vector

/**
 * @brief   Return a vector of iterators to the n largest elements of the given
 *          range.
 *
 * @tparam  R
 *          The type of range.
 * @param   range
 *          The input range to find the largest elements in.
 * @param   n
 *          The number of largest elements to return.
 *
 * @return  Vector containing iterators to the n largest elements of the range.
 */
template <std::ranges::forward_range R>
constexpr std::vector<std::ranges::borrowed_iterator_t<R>>
max_n_elements(R &&range, size_t n) {
    // The iterator type of the input range
    using iter_t = std::ranges::borrowed_iterator_t<R>;
    // The range of iterators over the input range
    auto iterators = std::views::iota(std::ranges::begin(range),
                                      std::ranges::end(range));
    // Vector of iterators to the largest n elements
    std::vector<iter_t> result(n);
    // Lambda function to compare two iterators: dereference them and compare
    // their values
    auto compare = [](iter_t it_a, iter_t it_b) { return *it_a > *it_b; };
    // Sort the largest n elements of the input range, and store iterators to
    // these elements to the result vector
    std::ranges::partial_sort_copy(iterators, result, compare);
    return result;
}
```

C++ 20 Driver

```cpp
#include <iostream>

int main() {
    std::vector<int> v = {10, 12, 8, -5, 8, 3, -2, 1, 9, 12};
    size_t n = 4;
    auto max_elems = max_n_elements(v, n);
    std::cout << "Largest elements:" << std::endl;
    for (auto it : max_elems) {
        auto idx = it - v.begin();
        std::cout << "    [" << idx << "]:\t" << *it << std::endl;
    }
}
```

```
Largest elements:
    [1]: 12
    [9]: 12
    [0]: 10
    [8]: 9
```

C++ 11

C++ 11 Algorithms Implementation

```cpp
#include <algorithm> // partial_sort_copy
#include <iterator>  // forward_iterator_tag
#include <vector>    // vector

/// Range iterator type, to iterate over ranges of iterators
template <class Iterator>
struct RangeIterator {
    using value_type = Iterator;
    using reference = const Iterator &;
    using pointer = void;
    using difference_type = ptrdiff_t;
    using iterator_category = std::forward_iterator_tag;

    value_type it;

    RangeIterator(Iterator it) : it(it) {}
    reference operator*() const { return it; };
    RangeIterator &operator++() { return ++it, *this; }
    RangeIterator operator++(int) { RangeIterator t = *this; ++it; return t; }
    bool operator!=(RangeIterator other) const { return it != other.it; }
    bool operator==(RangeIterator other) const { return it == other.it; }
};

/**
 * @brief   Return a vector of iterators to the n largest elements of the given
 *          range.
 *
 * @tparam  InputIt
 *          The type of iterator over the input range.
 * @param   first
 *          The iterator to the beginning of the input range.
 * @param   last
 *          The iterator to the end of the input range.
 * @param   n
 *          The number of largest elements to return.
 *
 * @return  Vector containing iterators to the n largest elements of the range.
 */
template <class InputIt>
constexpr std::vector<InputIt>
max_n_elements(InputIt first, InputIt last, size_t n) {
    // An iterator over the iterators over the input range
    using iter_iter_t = RangeIterator<InputIt>;
    // Vector of iterators to the largest n elements
    std::vector<InputIt> result(n);
    // Lambda function to compare two iterators: dereference them and compare
    // their values
    auto compare = [](InputIt it_a, InputIt it_b) { return *it_a > *it_b; };
    // Sort the largest n elements of the input range, and store iterators to
    // these elements to the result vector
    std::partial_sort_copy(iter_iter_t(first), iter_iter_t(last),
                           std::begin(result), std::end(result),
                           compare);
    return result;
}
```

C++ 11 Driver

```cpp
#include <iostream>

int main() {
    std::vector<int> v = {10, 12, 8, -5, 8, 3, -2, 1, 9, 12};
    size_t n = 4;
    auto max_elems = max_n_elements(v.begin(), v.end(), n);
    std::cout << "Largest elements:" << std::endl;
    for (auto it : max_elems) {
        auto idx = it - v.begin();
        std::cout << "    [" << idx << "]:\t" << *it << std::endl;
    }
}
```

```
Largest elements:
    [1]: 12
    [9]: 12
    [0]: 10
    [8]: 9
```