Biblioteka <ranges>

Paweł Skorupa

Jakub Ciołek

Czym jest biblioteka ranges?

- Nazywana jest często „STL 2.0” (nowe możliwości kompozycji, mniej podatna na błędy)
- Posiada znaczną część funkcjonalności biblioteki STL m.in. wszystkie funkcje z biblioteki <algorithm>
- Funkcje z biblioteki <numeric> nie zostały jeszcze wprowadzone (planowane wprowadzenie w C++23)
- Wprowadza nowe pojęcia takie jak range, view

Komunikacja o błędach

- Biblioteka <ranges> w czytelniejszy sposób informuje programistę o błędach. Przykład:

```
std::list<int> dt = { 1, 4, 3, 2 };
std::sort(dt.begin(), dt.end());
```

- Błąd wykrywany w czasie kompilacji
- Nieczytelny komunikat – kilkadziesiąt linijek

```
std::list<int> dt = { 1, 4, 3, 2 };
std::ranges::sort(dt.begin(), dt.end());
```

- Błąd wykrywany jeszcze przed kompilacją
- Jasny komunikat – w funkcji sort wymagany random_access_iterator (list_iterator nim nie jest)

Czym jest range?

- W uproszczeniu, wszystko co można przedstawić za pomocą .begin() oraz .end()
- Czyli kontenery vector, deque, list itd.
- Algorytmy przyjmujące jako argumenty pary iteratorów posiadają teraz przeciążenia, w których wystarczy podać kontener:

sort(vec.begin(), vec.end()) = sort(vec)

Czym jest range?

- Ponadto:
- Ciągi <i, n), fragmenty kontenerów
- Ciągi z warunkiem <i, warunek), elementy od i do pierwszego, który nie spełnia warunku
- Ciągi nieskończone <i…)

Czym jest view?

- Lekki obiekt reprezentujący iterowany ciąg (range). Czas kopiowania, przenoszenia oraz przypisania jest stały – złożoność obliczeniowa O(1)
- Obiekty view nie są właścicielami swoich elementów
- Mają leniwą naturę, tzn. nie wykonują obliczeń dopóki nie jest to konieczne

Zastosowania <ranges>

- Konstrukcje typu pipeline

```
auto is_odd = [](int a) {return a % 2 == 1; };

std::vector<int> vec{ 1,2,3,4,5,6,7 };

namespace rv = std::ranges::views;

for (int v : rv::reverse(rv::take(rv::filter(rv::reverse(vec), is_odd), 2)))
{
    std::cout << v << " ";
}
```

Chcemy wypisać ostatnie dwie nieparzyste liczby z wektora.
Tradycyjny zapis z wykorzystaniem nawiasów jest dosyć nieczytelny

Zastosowania <ranges>

- Konstrukcje typu pipeline

```
auto is_odd = [](int a) {return a % 2 == 1; };

std::vector<int> vec{ 1,2,3,4,5,6,7 };

namespace rv = std::ranges::views;

for (int v : rv::reverse(vec) | rv::filter(is_odd) | rv::take(2) | rv::reverse)
{
    std::cout << v << " ";
}
```

Dużo czytelniej – łatwo odczytać kolejne operacje

Zastosowania <ranges>

- unreachable_sentinel

```
std::vector<int> dt(100000000);
std::iota(std::begin(dt), std::end(dt), 0);
std::ranges::shuffle(dt, std::mt19937(std::random_device() ()));

auto pos2 = std::find(dt.begin(), dt.end(), 543210);
```

- Przy każdej iteracji sprawdzany jest warunek (it != end)
- Wiemy, że szukany element na pewno znajduje się w wektorze, sprawdzanie czy wyszliśmy poza zakres jest zbędne

Zastosowania <ranges>

- unreachable_sentinel

```
std::vector<int> dt(100000000);
std::iota(std::begin(dt), std::end(dt), 0);
std::ranges::shuffle(dt, std::mt19937(std::random_device() ()));

auto pos2 = std::ranges::find(dt.begin(), std::unreachable_sentinel, 543210);
```

Warunek (it != end) zostaje na stałe zastąpiony wartością true przy pierwszej iteracji

Zastosowania <ranges>

- Lazy evaluation

```
for (int i : std::views::iota(0, 10)) vec.push_back(i);

for (int i : std::views::iota(0) | std::views::take(10)) vec.push_back(i);
```

- Subtelna różnica – w pierwszym przypadku funkcja iota po prostu generuje kolejne liczby od 0 do 9, w drugim przypadku mogłaby generować w nieskończoność, ale my prosimy o wygenerowanie 10 liczb
- W przypadku tego kodu bez różnicy, staje się bardzo przydatne w sytuacji kiedy nie wiemy ile razy musimy inkrementować, aby osiągnąć pożądany wynik

Dziękujemy za uwagę

- Źródła:
  - https://en.cppreference.com
  - https://hannes.hauswedell.net/post/2019/11/30/range_intro/
  - https://itnext.io/c-20-ranges-complete-guide-4d26e3511db0
  - https://www.modernescpp.com/index.php/c-20-functional-patterns-with-the-ranges-library
  - https://www.youtube.com/watch?v=d_E-VLyUnzc
  - https://www.youtube.com/watch?v=SYLgG7Q5Zws