Podstawy Programowania
Semestr letni 2022/23
Materiały z laboratorium i zadania domowe

Przemysław Olbratowski

3 marca 2023

Slajdy z wykładu są dostępne w serwisie UBI. Informacje organizacyjne oraz formularz do uploadu prac domowych znajdują się na stronie `info.wsisiz.edu.pl/~olbratow`. Przy zadaniach domowych w nawiasach są podane terminy sprawdzeń.

13.2 Zadania domowe z działu Lambdy (21, 28 czerwca, 5 lipca)

13.2.1 All Of: Czy wszystkie elementy spełniają warunek

Napisz funkcję `all_of`, która przyjmuje niemodyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych oraz predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja zwraca prawdę, jeżeli wszystkie elementy wycinka spełniają ten predykat, albo fałsz w przeciwnym razie. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {-7, 3, 1, 5, 11};
    bool result = all_of(vector.cbegin(), vector.cend(),
                         [](int element) {return element % 2; });
    std::cout << std::boolalpha << result << std::endl; }
```

Wykonanie

```
Out: true
```

13.2.2 Any Of: Czy którykolwiek element spełnia warunek

Napisz funkcję `any_of`, która przyjmuje niemodyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych oraz funktor przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja zwraca prawdę, jeżeli przynajmniej jeden element wycinka spełnia ten predykat, albo fałsz w przeciwnym razie. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {-7, 3, 1, 5, 11};
    bool result = any_of(vector.cbegin(), vector.cend(),
                         [](int element) {return element % 3 == 0; });
    std::cout << std::boolalpha << result << std::endl; }
```

Wykonanie

```
Out: true
```

13.2.3 Copy If: Kopiowanie warunkowe

Napisz funkcję `copy_if`, która przyjmuje niemodyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych, modyfikujący iterator początkowy innego wycinka oraz predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja kopiuje do drugiego wycinka te elementy pierwszego, które spełniają ten predykat. Funkcja zwraca modyfikujący iterator końcowy wycinka powstałego ze skopiowanych elementów. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector1 {-7, 0, 1, 0, 11};
    std::vector<int> vector2(5);
    auto result = copy_if(vector1.cbegin(), vector1.cend(), vector2.begin(),
                          [](int element) {return element; });
```

```
    for (auto iterator = vector2.cbegin(); iterator < result;) {
        std::cout << *iterator++ << " "; }
    std::cout << std::endl; }
```

Wykonanie

```
Out: -7 1 11
```

13.2.4 Count If: Zliczanie warunkowe

Napisz funkcję `count_if`, która przyjmuje niemodyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych oraz predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja zwraca liczbę elementów wycinka, które spełniają ten predykat. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {5, 17, 8, 3, 36, 7};
    int result = count_if(vector.cbegin(), vector.cend(),
                          [](int element) {return element % 2; });
    std::cout << result << std::endl; }
```

Wykonanie

```
Out: 4
```

13.2.5 Find If: Wyszukiwanie warunkowe

Napisz funkcję `find_if`, która przyjmuje modyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych oraz predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja zwraca modyfikujący iterator pierwszego elementu wycinka, który spełnia ten predykat albo końcowy iterator wycinka, jeżeli taki element w nim nie występuje. Napisz analogiczną funkcję `find_if` przyjmującą i zwracającą iteratory niemodyfikujące. Funkcje powinny być przystosowane do użycia w przykładowym programie poniżej. Funkcje korzystają tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {3, -1, 7, 12, -5, 7, 10};
    auto result1 = find_if(vector.begin(), vector.end(),
                           [](int element) {return element > 10; });
    auto result2 = find_if(vector.cbegin(), vector.cend(),
                           [](int element) {return element > 10; });
    std::cout << result1 - vector.begin() << " "
              << result2 - vector.cbegin() << std::endl; }
```

Wykonanie

```
Out: 3 3
```

13.2.6 Min: Pierwsza z dwóch liczb

Napisz funkcję `min`, która przyjmuje dwie liczby całkowite oraz predykat przyjmujący dwie takie liczby i zwracający wartość logiczną. Predykat ten określa pewien porządek sortowania i zwraca prawdę, gdy jego pierwszy argument wypada w tym porządku przed drugim, albo fałsz w przeciwnym razie. Funkcja zwraca tę z przekazanych liczb, która wypada w tym porządku nie później niż druga. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z pliku nagłówkowego `functional`.

Przykładowy program

```
int main() {
    int result = min(1, 2, [](int a, int b) {return a > b; });
    std::cout << result << std::endl; }
```

Wykonanie

```
Out: 2
```

13.2.7 Partition: Partycjonowanie

Napisz funkcję `partition`, która przyjmuje modyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych oraz predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja przenosi wszystkie elementy spełniające ten predykat przed wszystkie, które go nie spełniają. Początkowa kolejność elementów w każdej grupie nie musi być zachowana. Funkcja zwraca modyfikujący iterator końcowy wycinka powstałego z przeniesionych elementów spełniających predykat. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {2, 3, 8, 1, 6, 5, 0, 7, 4, 9};
    auto result = partition(vector.begin(), vector.end(),
                            [](int element) {return element % 2; });
    std::cout << result - vector.begin() << std::endl;
    for (auto iterator = vector.cbegin(); iterator < vector.cend();) {
        std::cout << *iterator++ << " "; }
    std::cout << std::endl; }
```

Przykładowe wykonanie

```
Out: 5
Out: 9 3 7 1 5 6 0 8 4 2
```

13.2.8 Remove If: Usuwanie warunkowe - indywidualnie

Napisz funkcję `remove_if`, która przyjmuje modyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych oraz predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną. Funkcja kopiuje kolejne elementy niespełniające tego predykatu na kolejne pozycje na początku wycinka i zwraca modyfikujący iterator końcowy wycinka powstałego ze skopiowanych elementów. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {-7, 5, 2, 2, 11, 2, 3};
    auto result = remove_if(vector.begin(), vector.end(),
                            [](int element) {return element < 3; });
    for (auto iterator = vector.cbegin(); iterator < result;) {
        std::cout << *iterator++ << " "; }
    std::cout << std::endl; }
```

Wykonanie

```
Out: 5 11 3
```

13.2.9 Replace If: Warunkowa zamiana

Napisz funkcję `replace_if`, która przyjmuje modyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych, predykat przyjmujący liczbę całkowitą i zwracający wartość logiczną, oraz dowolną wartość całkowitą. Funkcja zastępuje tą wartością wszystkie elementy wycinka spełniające przekazany predykat. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector {2, 3, 8, 1, 6, 5, 0, 7, 4, 9};
    replace_if(vector.begin(), vector.end(),
               [](int element) {return element % 3; }, 0);
    for (auto iterator = vector.cbegin(); iterator < vector.cend();) {
        std::cout << *iterator++ << " "; }
    std::cout << std::endl; }
```

Wykonanie

```
Out: 0 3 0 0 6 0 0 0 0 9
```

13.2.10 Transform: Transformacja elementów - grupowo

Napisz funkcję `transform`, która przyjmuje niemodyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych, modyfikujący iterator początkowy innego wycinka, oraz operację przyjmującą i zwracającą liczbę całkowitą. Funkcja wpisuje do kolejnych komórek drugiego wycinka wyniki wywołania tej operacji na kolejnych elementach pierwszego i zwraca modyfikujący iterator końcowy drugiego wycinka. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja korzysta tylko z plików nagłówkowych `functional` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector1 {-7, 5, 1, 2, 11};
    std::vector<int> vector2(5);
    auto result = transform(vector1.cbegin(), vector1.cend(), vector2.begin(),
                            [](int element) {return element * element; });
    for (auto iterator = vector2.cbegin(); iterator < result;) {
        std::cout << *iterator++ << " "; }
    std::cout << std::endl; }
```

Wykonanie

```
Out: 49 25 1 4 121
```