1 Transform

1.1 Aufgabe

In dieser Aufgabe lernen Sie, wie Sie eigene Algorithmen schreiben, welche dieselben Mechanismen anwenden, die auch in der Standard-Library üblich sind.

a) Erstellen Sie eine Template-Funktion `PrintContainer`, welche alle Elemente eines beliebigen Containers auf die Konsole ausdruckt.

b) Erstellen Sie analog zum bereits existierenden `std::transform` eine eigene Implementation `Transform`. Diese soll Neben den `begin`- und `end`-Iteratoren einen weiteren Iterator entgegennehmen. Zudem eine unäre-Funktion, welche ein Element des Containers beliebig manipuliert und den geänderten Wert zurückgibt. Die Funktion `Transform` soll nun jedes Element im Interval $[begin, end)$ mittels der unären Funktion manipulieren und an die nächste Position des Output-Iterators schreiben.

c) Um die beiden Funktionen zu testen, schreiben Sie ein Programm, welches einen `std::vector` mit Integer-Werten füllt und diese mittels der `PrintContainer`-Funktion ausgibt. Nun verwenden Sie `Transform` um jeden Wert im Vektor mit 2 zu multiplizieren. Geben Sie den Vektor erneut aus.

1.2 Lösung

a)

```
template<typename InputIterator>
void PrintContainer(InputIterator begin, InputIterator end) {
  for (auto it = begin; it != end; ++it) {
    std::cout << *it << ", ";
  }
  std::cout << std::endl;
}
```

b)

```
template<typename InputIterator, typename OutputIterator, typename
    UnaryFunction>
void Transform(InputIterator begin, InputIterator end, OutputIterator out,
    const UnaryFunction& function) {
  for (auto it = begin; it != end; ++it) {
    *out = function(*it);
    ++out;
  }
```

```
7 }
```

Anstatt einen weiteren Template-Parameter `UnaryFunction` hätte man auch `std::function<int(int)>` verwenden können.

c)

```
1  std::vector<int> data = { 1, 3, 6, 7, 8 };
2  PrintContainer(data.cbegin(), data.cend());
3
4  const auto manipulator = [](const int& value) {
5    return value * 2;
6  };
7  Transform(data.begin(), data.end(), data.begin(), manipulator);
8
9  PrintContainer(data.cbegin(), data.cend());
```