Tabelle 1 – Wichtige Range Adapters des C++20-Standards
R, V sollen für Eingangs-Range und Ausgang-View stehen

Adapter	Argument	Beschreibung
`all`		Erzeugt View V aus allen Elementen der Range R
`counted`*	$(begin, n \in \mathbb{N})$	$V := R[begin,\ begin{+}n)$
`filter`	$P : T \to \{0,1\}$	$V := (r_0, \ldots, r_{n-1} \mid 0 \leqslant i < n = \lvert R\rvert : r_i = R[i] \land P(r_i) = 1)$
`transform`	$\pi : T \to T'$	$V := (r'_0, \ldots, r'_{n-1} \mid 0 \leqslant i < n = \lvert R\rvert : r_i = R[i] \land \pi(r_i) = r'_i)$
`take`	$i \in \mathbb{N}$	$V := (r_0, \ldots, r_{i-1} \mid 0 \leqslant j \leqslant i < \lvert R\rvert : r_j = R[j])$
`take_while`	$P : T \to \{0,1\}$	Wie `take` mit $i := \min(\{j \mid P(R[j]) = 0\})$
`drop`	$i \in \mathbb{N}$	$V := (r_i, \ldots, r_{n-1} \mid i \leqslant j \leqslant n = \lvert R\rvert : r_j = R[j])$
`drop_while`	$P : T \to \{0,1\}$	Wie `drop` mit $i := \min(\{j \mid P(R[j]) = 0\})$
`reverse`		$V := (r_0, \ldots, r_{n-1} \mid 0 \leqslant i < n = \lvert R\rvert : r_i = R[\lvert R\rvert - i - 1])$
`elements`	$\underbrace{T_1 \times \cdots \times T_n}_{R} \to \underbrace{T_i}_{V}$	Erstellt View durch Abbildung der Eingangstupel auf das i. Element
`keys`	$\underbrace{T_1 \times T_2}_{R} \to \underbrace{T_1}_{V}$	Erstellt View aus dem ersten Elementen der Eingangsdupel
`values`	$\underbrace{T_1 \times T_2}_{R} \to \underbrace{T_2}_{V}$	Erstellt View aus dem zweiten Elementen der Eingangsdupel
`join`		Erstellt eine „glatte“ View aus einer Range, die aus Ranges besteht
`split`		Erstellt eine View aus einer an einem Trennzeichen gestückelten Range
`common`		Erstellt Views mit gleich-typisierten `begin` und Sentinel aus Ranges, für welche das nicht zutrifft

$*$: ist nicht unter `ranges::counted_view` verfügbar

Tabelle 2 – Wichtige Range Adapters des C++23-Standards - Abkürzungen wie 1

Adapter	Beschreibung
`zip`$\langle$`_transform`$\rangle$	$\underbrace{\overbrace{T_1}^{R_1} \times \cdots \times \overbrace{T_n}^{R_n}}_{R} \to \underbrace{T_1 \times \cdots \times T_n}_{V}$
`adjacent`$\langle$`_transform`$\rangle^*$	Erstellt V aus Ranges von Teilsequenzen der Länge n von R
`join_with`	Wie `join` 1 mit Einfügen von Trennzeichen zwischen die geglätteten Strukturen
`chunk`$\langle$`_by`$\rangle$	Erstellt V aus Subranges der Länge n von R
`slide`	Wie `adjacent`, nur dass Tupel anstatt von Ranges gebildet werden
`stride`	Erstellt V aus jedem i. Element von R
`enumerate`	$T \to \mathbb{N}_{\nu} \times T : t_i \mapsto (i, t_i)$
`cartesian_product`	

$*$: n muss zur Übersetzungszeit bestimmt sein