

(Standard Template Library) STL

הנובע מכך ש-STL הוא אוסף של כלים לניהול נתונים.

- Container: אוסף נתונים (vector, list, deque, set, multiset, map, multimap, bitset, valarray).
- Adapters: מודולים המאפשרים שימוש בContainerים באמצעותם של מנגנונים מותאמים.
- Utility: מנגנונים עיקריים (copy, swap, compare, etc.) ופונקציות מומלצות (find, find_if, transform, etc.).

: containers (ב-vector, list, deque)

(Containers, probably). vector, list, deque, set, multiset, map, multimap - Sequence Containers -

: SC - S allways

(vector, list, deque). vector מוגדר כ-vector (vector = Vector -

array). list, deque הם vector's аналогים - deque -

רשימה (list) או רשימה דו-방ת (deque). list מוגדר כ-list (list = list -
(prev = i, next = i+1))

רשותן של אובייקטים - Associative Containers -

: AC - S always

sett (sett) מוגדר כsett (sett = set - set -
sett מוגדר כsett (sett = set - set - set))

sett מוגדר כsett (sett = multi-set - multi-set -

sett מוגדר כsett (sett = set - set - map -

sett מוגדר כsett (sett = multi-map -

Adapter (Adapter). מוגדר כsett (sett = stack - priority queue, queue)

STL מוגדר כsett (sett = stack, queue, priority queue, etc.)

(vector, list, deque, set, multiset, map, multimap, bitset) - bitset

(vector, list, deque, set, multiset, map, multimap, bitset, valarray) - valarray

string - string

Hashtable, slist, svector, etc. מוגדרים כsett (sett = set, multiset, map, multimap, bitset, valarray, string) ומשתמשים בContainerים.

T (template), STL (standard template library), copy (copy), swap (swap), compare (compare), transform (transform), find (find), find_if (find_if), reverse (reverse), sort (sort), etc. (etc.).

copy (copy), copy_if (copy_if), swap (swap), compare (compare), transform (transform), find (find), find_if (find_if), reverse (reverse), sort (sort), etc. (etc.).

= 16/20

לעומת נספחים מחרוזת

reference, iterator (ws) of length 6 of `string::Properties`. to call it's 23-24 methods.

`Container` of `C` is `STL` or `vector` (array). `Container` ≥ 18 elements \Rightarrow `vector` (array). `Container` ≤ 6 elements \Rightarrow `string` (array). `Container` ≥ 18 elements \Rightarrow `vector` (array).

Type Check: `if` (`"s" is <6`) \Rightarrow `s` is `string` (array). `STL` \Rightarrow `s` is `string` (array).
`(else)` `find` \Rightarrow `0.0` ref to `string` (array). `find` \Rightarrow `0.0` ref to `string` (array).
`18` elements \Rightarrow `vector` (array). `18` elements \Rightarrow `vector` (array).
`Container` ≥ 18 elements \Rightarrow `vector` (array). `Container` ≤ 18 elements \Rightarrow `string` (array).
`Container` ≥ 18 elements \Rightarrow `vector` (array). `Container` ≤ 18 elements \Rightarrow `string` (array).
`Container` ≥ 18 elements \Rightarrow `vector` (array). `Container` ≤ 18 elements \Rightarrow `string` (array).
`Container` ≥ 18 elements \Rightarrow `vector` (array). `Container` ≤ 18 elements \Rightarrow `string` (array).

do not know how many bytes (6) to write to `cout` because `cout` is `<operator>`.
`cout` (6) bytes. `cout` is `<operator>`. `cout` is `<operator>`. `cout` is `<operator>`.
`cout` is `<operator>`. `cout` is `<operator>`. `cout` is `<operator>`.

Up to date with `list`: `list::iterator` \Rightarrow `list::iterator` \Rightarrow `list::iterator`.
`list::iterator` \Rightarrow `list::iterator`. `list::iterator` \Rightarrow `list::iterator`.
`list::iterator` \Rightarrow `list::iterator`. `list::iterator` \Rightarrow `list::iterator`.

Forward - backward - middle - begin - end

middle \Rightarrow begin (begin) - middle (middle) - end (end).

vector \Rightarrow begin (begin) - end (end).

vector \Rightarrow begin (begin) - end (end). vector \Rightarrow begin (begin) - end (end).
`begin` \Rightarrow `vector::iterator`. `begin` \Rightarrow `vector::iterator`. `begin` \Rightarrow `vector::iterator`.
`end` \Rightarrow `vector::iterator`. `end` \Rightarrow `vector::iterator`. `end` \Rightarrow `vector::iterator`.

`v.end() - 1` \Rightarrow `v.begin()`. `v.begin() - 1` \Rightarrow `v.end()`.
`v.begin() - 1` \Rightarrow `v.end()`. `v.begin() - 1` \Rightarrow `v.end()`.
`v.end() - 1` \Rightarrow `v.begin()`. `v.end() - 1` \Rightarrow `v.begin()`.
`v.end() - 1` \Rightarrow `v.begin()`. `v.end() - 1` \Rightarrow `v.begin()`.
`v.end() - 1` \Rightarrow `v.begin()`. `v.end() - 1` \Rightarrow `v.begin()`.
`v.end() - 1` \Rightarrow `v.begin()`. `v.end() - 1` \Rightarrow `v.begin()`.
`v.end() - 1` \Rightarrow `v.begin()`. `v.end() - 1` \Rightarrow `v.begin()`.

swap \Rightarrow begin (begin) - end (end).

deque

`deque` \Rightarrow `deque::iterator`. `deque` \Rightarrow `deque::iterator`. `deque` \Rightarrow `deque::iterator`.
`deque` \Rightarrow `deque::iterator`. `deque` \Rightarrow `deque::iterator`. `deque` \Rightarrow `deque::iterator`.

list

bidirectional iterator (begin, end), random access, pointer
 (no random access) (no begin, end, size(), etc.)

insert

Array

push_back, push_front
 \rightarrow insert at positionerasepop_back, pop_front
 \rightarrow erase at positionfind

find first element matching

O(n) time complexity
 O(1) space complexitylist

| | | |
|---|--|--|
| push_back, push_front \rightarrow insert at position | pop_back, pop_front \rightarrow erase at position | find first element matching \rightarrow O(n) time complexity O(1) space complexity |
|---|--|--|

push merge = 1, push = 2, splice = 3, remove = 4, sort = 5, copy = 6, reverse = 7, sort = 8, size = 9, etc.

array

sort examples (vector, algorithm), vector, array, copy, array

array (vector, algorithm, copy, swap, reverse, copy, copy)

copy (vector, algorithm, copy, swap, reverse, copy, copy)

Safe STL

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)

safe STL (vector, algorithm, copy, swap, reverse, copy, copy)