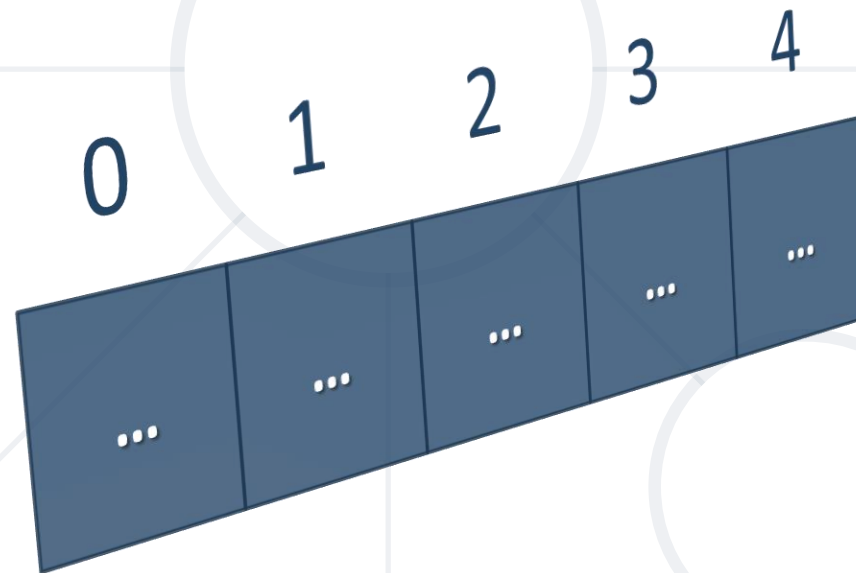


Stacks and Queues

Processing Sequences of Elements



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>



sli.do

#cpp-advanced

1. `std::stack<T>`

- LIFO - last in, first out

2. `std::queue<T>`

- FIFO - first in, first out



- Wrap a container (e. g. **vector**) with a different interface
- Allow you to express intentions better
- Make code more abstract and focused on the task, not the code

STL Adapters for common Computer Science data structures:

`std::stack`

last-in, first-out
(LIFO) data
structure

`std::queue`

first-in, first-out
(FIFO) data
structure

`std::priority_queue`

data structure that gives quick
access to the "highest priority
item"



Overview and Working with Stack

`std::stack<T>`

- Represents a container (a **deque** by default) working like a stack
- A stack is a "last-in, first-out structure" (LIFO)
 - Example: *Pile of dishes: the last dish you put is the first you can remove*
- Access to elements other than **top()** is not provided

top() gets the top

pop() removes top

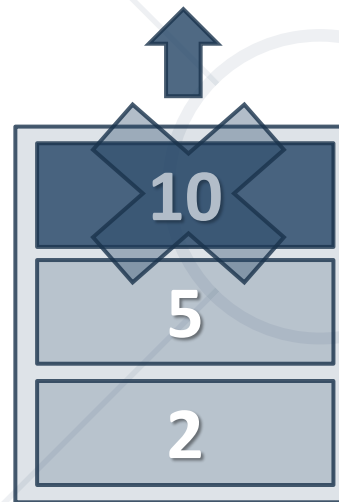
push(T) adds to top

Stack

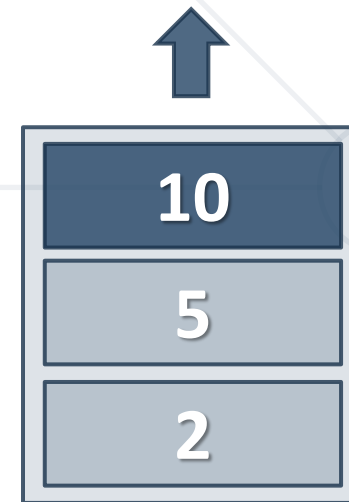
- Stacks provide the following functionality
 - Pushing** an element at the **top of the stack**
 - Popping** element from the **top of the stack**
 - Getting** the topmost element **without removing it**



Push



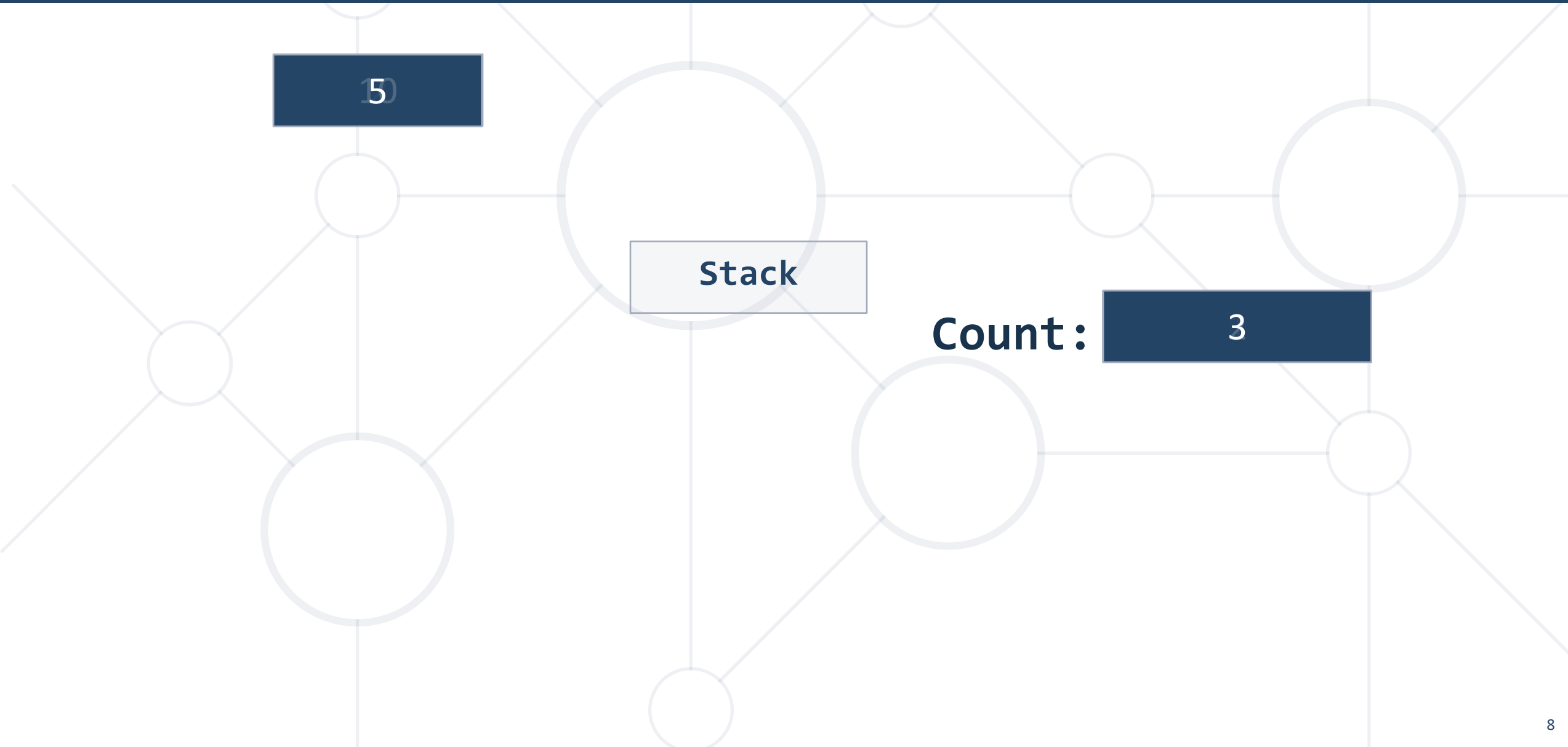
Pop



Top



push() – Adds an Element On Top of the Stack



pop() – Returns and Removes the Last Element

Stack

2

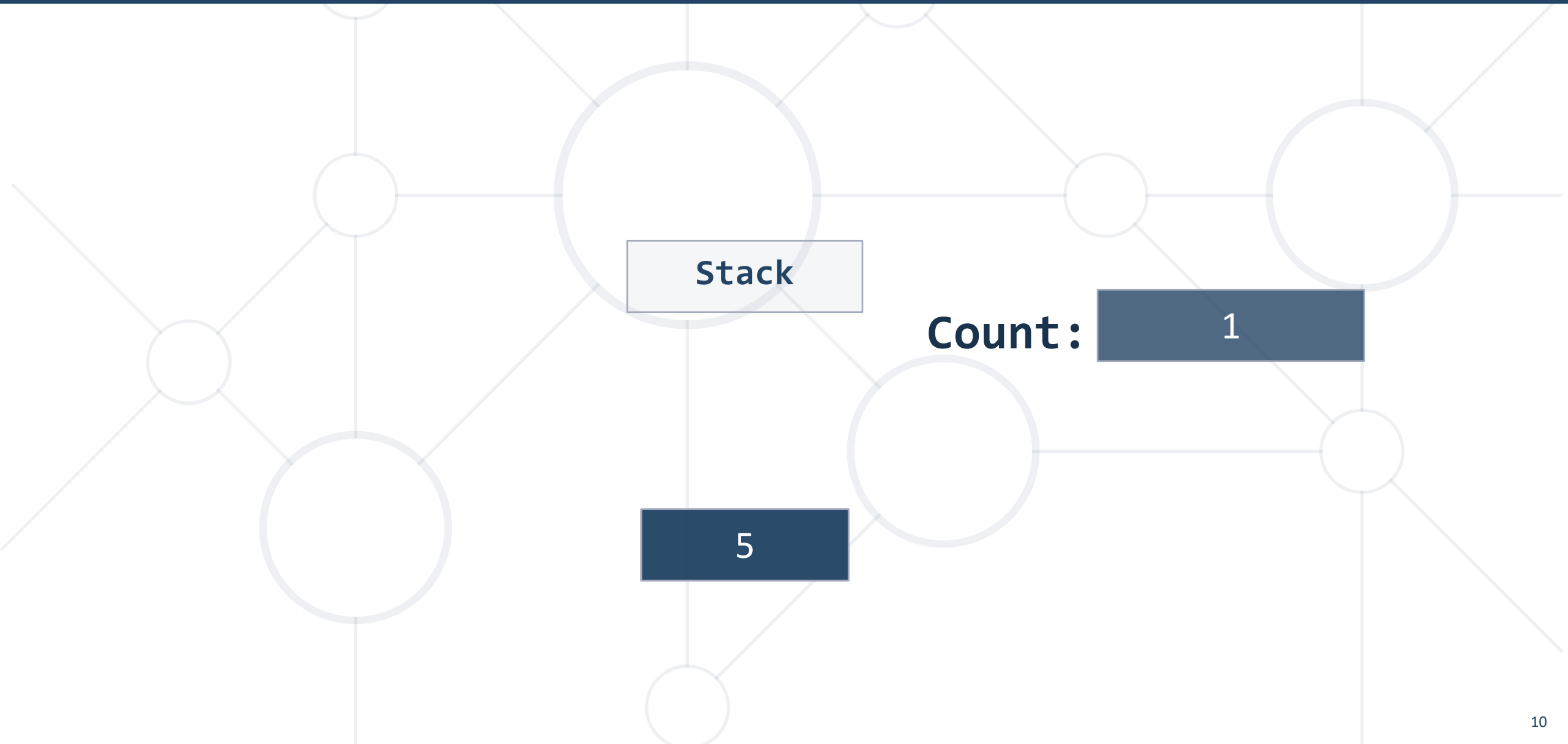
10

5

Count:

2

top() - Returns the Last Element



Stack Example: "Browser History"

- Write a program which takes 2 types of browser instructions:
 - Normal navigation: a URL is set, given by a string
 - The string **/back** that sets the current URL to the last set URL
- After each instruction the program should print the current URL
- If the **/back** instruction can't be executed, print **"no previous URLs"**





Overview and Working with Queue

`std::queue<T>`

- Represents a container (**deque** by default) working like a queue
- A queue is a "first-in, first-out" structure (FIFO)
 - Example: *Line at a store – first person in the line is the first to get out*
- Access to elements other than **front()** is not provided

front() gets first

pop() removes first

push(T) adds to back

Queue Example: "Browser History"

- Extend "Browser History" with a **/forward** instruction
 - Visits URLs that were navigated away from by **/back**
 - Each **/forward** instruction visits the next most-recent such URL
 - If a normal navigation happens, all potential **/forward** URLs are removed until a new **/back** instruction is given
- If the **/forward** instruction can't be executed, print **"no next URLs"**





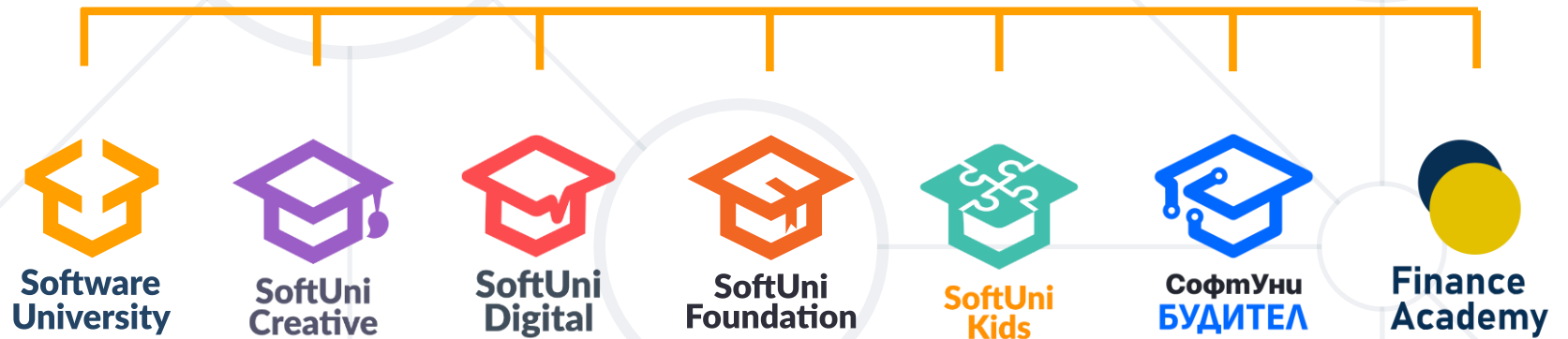
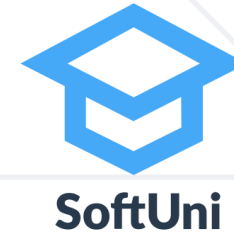
`std::priority_queue`

- Represents a queue, but elements are **ordered by priority**
 - By default, "**larger**" elements have higher priority
 - **Example**: *Queue at a hospital's emergency room – patients with more serious cases treated BEFORE those with less serious ones*
- Higher priority elements **move in front of lower priority ones**
 - Getting top-priority element is **$O(1)$** , insertion is **$\log(N)$**
 - To get top-priority element use **`top()`** (instead of **`front()`**)

- `stack<T>`
 - **LIFO**
- `queue<T>`
 - **FIFO**
- Container Adaptors
 - Each is efficient for certain use-cases



Questions?



SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg, about.softuni.bg
- Software University Foundation
 - softuni.foundation
- Software University @ Facebook
 - facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

