



**WIKIPEDIA**  
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

[Current events](#)

[Random article](#)

[Donate to Wikipedia](#)

[Wikipedia store](#)

Interaction

[Help](#)

[About Wikipedia](#)

[Community portal](#)

[Recent changes](#)

[Contact page](#)

Tools

[What links here](#)

[Related changes](#)

[Upload file](#)

[Special pages](#)

[Permanent link](#)

[Page information](#)

[Wikidata item](#)

[Cite this page](#)

Print/export

[Create a book](#)

[Download as PDF](#)

[Printable version](#)

Languages

[Català](#)

[Deutsch](#)

[Esperanto](#)

[فارسی](#)

[Français](#)

[Italiano](#)

[Nederlands](#)

[日本語](#)

[Polski](#)

[Português](#)

[Runa Simi](#)

[Русский](#)

[ᲣᲚᲠ](#)

[Українська](#)

[中文](#)

[Edit links](#)

Article [Talk](#)

[Read](#)

[Edit](#)

[View history](#)



# Container (abstract data type)

From Wikipedia, the free encyclopedia

(Redirected from [Container \(data structure\)](#))

*"Container (computer science)" redirects here. For the abstract notion of containers in [type theory](#), see [Container \(type theory\)](#).*



This article **may require [cleanup](#) to meet Wikipedia's [quality standards](#)**. The specific problem is: **text is clunky**. Please help [improve this article](#) if you can. *(March 2012)*

In [computer science](#), a **container** is a [class](#), a [data structure](#),<sup>[1][2]</sup> or an [abstract data type](#) (ADT) whose instances are collections of other objects. In other words, they store objects in an organized way that follows specific access rules. The size of the container depends on the number of objects (elements) it contains. Underlying implementation of various container types may vary in space and time complexity, which provides flexibility in choosing the right implementation for a given scenario.

## Contents [hide]

- 1 Overview
- 2 Types
  - 2.1 Value based containers
  - 2.2 Reference based containers
- 3 Single or associative
  - 3.1 Single value containers
  - 3.2 Associative containers
- 4 Examples of containers
- 5 Graphic containers
- 6 Implementations
- 7 See also
- 8 References
- 9 External links

## Overview [edit]

Containers can be looked at in three ways:

- Access** : Accessing the container elements. In the case of arrays, accessing is done with the array index. For stacks, access of elements is done using [LIFO](#) (Last In First Out)<sup>[3]</sup> (alternative name **FILO** (First In Last Out) and in queues it is done using [FIFO](#) (First In First Out)).<sup>[3][4]</sup>
- Storage** : Storage includes storing items in containers. Some containers are finite and some are infinite.
- Traversal** : This describes how the item can be [traversed](#).

Container classes are expected to implement methods to do the following:

- create a new empty container,
- report the number of objects it stores (size),
- delete all the objects in the container (clear),
- insert new objects into the container,
- remove objects from it,
- provide access to the stored objects.

Containers are sometimes implemented in conjunction with [iterators](#).

## Types [edit]

Containers can be divided into two groups:

- Value based containers*
- Reference based containers*

## Value based containers [edit]

Value based containers store copies of objects. If we access an object, the object returns a copy of it. If an external object changes after it has been inserted in the container, it does not affect the container's content.

## Reference based containers [edit]

Reference based containers store [pointers](#) or [references](#) to the object. If we access an object, the object returns a reference to it. If an external object is changed after it has been inserted in the container, it affects the content of the container.

## Single or associative [edit]

A container may be:

1. *Single value*
2. *Associative*

## Single value containers [edit]

Each object is stored independently in the container and it is accessed directly or with an [iterator](#).

## Associative containers [edit]

An [associative array](#), map, or dictionary is a container composed of (key,value) pairs, such that each key appears at most once in the container. The key is used to find the value, the object, if it is stored in the container.

## Examples of containers [edit]

Containers are divided in the [Standard Template Library](#) into [associative containers](#) and standard [sequence containers](#). Besides these two types, so-called [container adaptors](#) exist. Data structures that are implemented by containers include [arrays](#), [lists](#), [maps](#), [queues](#), [sets](#), [stacks](#), [tables](#), [trees](#), and [vectors](#).

## Graphic containers [edit]

[Widget toolkits](#) use special [widgets](#) also called *Containers* to group the other widgets together ([windows](#), [panels](#), ...). Apart from their graphical properties, they have the same type of behavior as **container classes**, as they keep a list of their child [widgets](#), and allow to add, remove, or retrieve [widgets](#) amongst their children.

## Implementations [edit]

- .NET: [System.Collections](#) (MSDN)
- ActionScript3: [AS3Commons Collections Framework](#)
- C++: [C++ Standard Library](#) (SC++L) or the obsolete [Standard Template Library](#) (STL)
- Java: [Java collections framework](#) (JCF)
- Objective-C: part of the [Foundation Kit](#)
- PL/SQL [Collections](#)<sup>[5]</sup>
- Python: [collections](#)  module
- [Scala](#) Mutable and Immutable Collections in the packages `scala.collection.mutable` and `scala.collection.immutable`

## See also [edit]

- [List of data structures](#)
- [Standard Template Library#Containers](#)
- [Collection \(abstract data type\)](#)
- [Stack data structure](#)

## References [edit]

- ↑ Paul E. Black (ed.), entry for *data structure* in *Dictionary of Algorithms and Data Structures*. US *National Institute of Standards and Technology*. 15 December 2004. Accessed on Oct 04, 2011.
- ↑ Entry *data structure* in the *Encyclopædia Britannica* (2009) [Online entry](#)  Accessed on Oct 04, 2011.
- ↑ <sup>a</sup> <sup>b</sup> [LIFO](#)(investopedia.com)

4. <sup>^</sup> [FIFO\(businessdictionary.com\)](#)<sup>↗</sup>
5. <sup>^</sup> ["PL/SQL Collections and Records"](#)<sup>↗</sup>. Retrieved 2013-04-20.

External links

[\[edit\]](#)

- [Container Data Structure Declaration and Initialization](#)<sup>↗</sup>
- [Container data structures](#)<sup>↗</sup>
- [Python SortedContainers module](#)<sup>↗</sup> A fast implementation of sorted list, sorted dict, and sorted set data types in Python.

<div><div>v · t · e</div></div>	<div><div><b>Data structures</b></div></div>	<div><div><a href="#">[show]</a></div></div>
<div><div>v · t · e</div></div>	<div><div><b>Data types</b></div></div>	<div><div><a href="#">[show]</a></div></div>

Categories:

[Abstract data types](#)

[Object-oriented programming](#)

[Data structures](#)

This page was last modified on 3 September 2015, at 03:00.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

