華東師範大學 | 数学科学学院
School of Mathematical Sciences, East China Normal University

第十六讲

标准模板库

STL：Standard Template Library

标准模板库 STL

C++ STL 是一套功能强大的 C++ 模板库，提供了大量的通用模板类和模板函数，这些模板类和模板函数可以实现多种流行和常用的算法和数据结构，如向量、链表、队列、栈，等等，大大提升软件开发的效率。

STL 三大核心组件

- 容器 Containers

  用来存储数据的一种数据结构（模板类），如向量，链表

- 算法 Algorithms

  对数据的各种操作（模板函数），如插入，排序，搜索等

- 迭代器 iterators

  访问容器中的数据的方法，作用类似于指针

1 容器 Containers

- 顺序容器/序列容器
- 关联容器
- 容器适配器

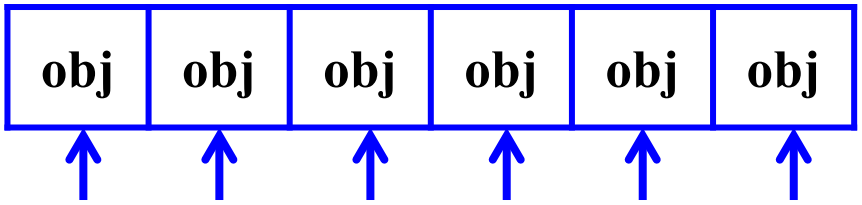
容器：顺序容器

顺序容器/序列容器（Sequential Containers）

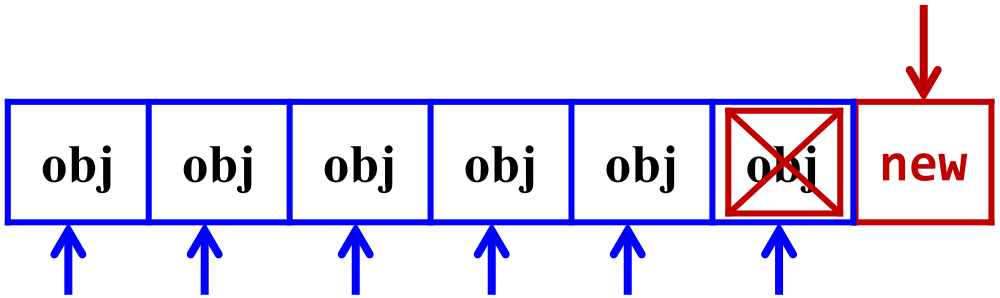
按顺序（物理/逻辑）存储数据，如数组，链表等 → 数据结构

`array`	数组，长度不能改变
`vector`	只能在最后面插入或删除数据
`deque`	与 `vector` 类似，但允许在最前面插入或删除数据
`list`	双向链表，可在任意位置插入或删除数据
`forward_list`	与 `list` 类似，但是单向的，只能沿一个方向访问
`string`	字符串，与 `vector` 类似，但存储的是字符

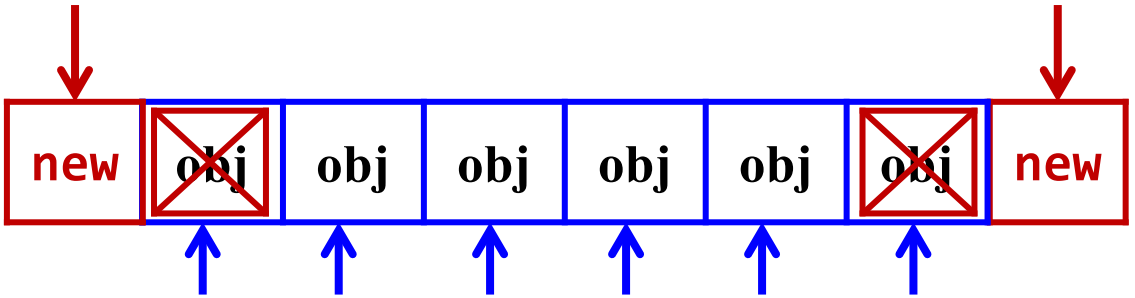
array：长度固定，可随意访问其中的任何元素



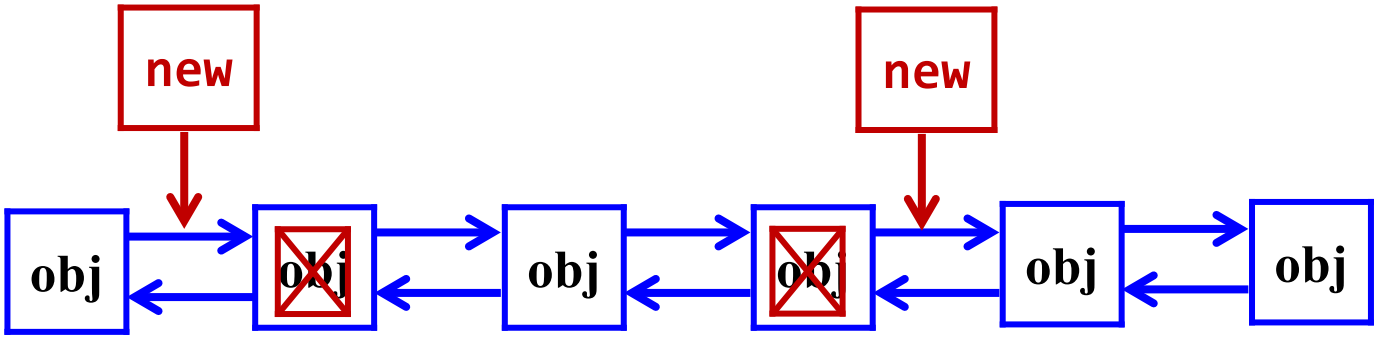
vector：与数组类似，且长度可变，但只能在最后面添加或删除数据



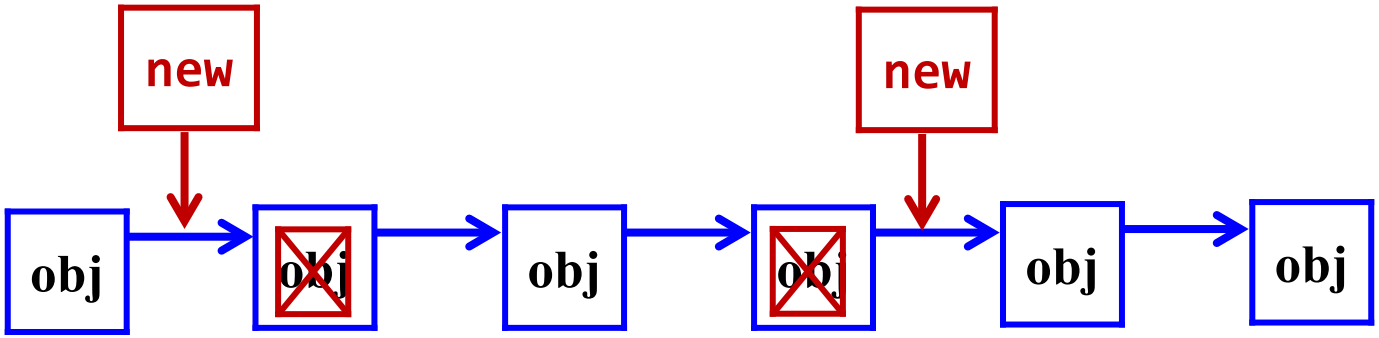
deque：与 vector 类似，但可在两头添加或删除数据



`list`：双向链表，可在任意位置添加或删除数据，但只能从第一个元素或最后一个元素开始访问



`forward_list`：单向链表，与 `list` 类似，但只能单向访问



容器：关联容器

关联容器（Associative Containers）

按排序方式存储数据，就像词典一样 → 方便搜索

`set`	存储互不相同的数据，插入数据时进行排列
`unordered_set`	与 `set` 类似，但按 Hash 值排序
`map`	存储“键-值”对，按唯一的键排序
`unordered_map`	与 `map` 类似，但按“键”的 Hash 值排序
`multiset`	与 `set` 类似，但允许有相同的数据
`unordered_multiset`	与 `unordered_set` 类似，但允许有相同的数据
`multimap`	与 `map` 类似，但不要求“键”唯一
`unordered_multimap`	与 `unordered_map` 类似，但不要求“键”唯一

容器适配器

- 容器适配器（Associative adapters）

顺序适配器和关联适配器的变种，增加一些特殊功能

stack	栈，按后进先出（LIFO）方式存储数据
queue	队列，按先进先出（FIFO）方式存储数据
priority_queue	队列，但能保证最大元素总在最前面

2 算法 Algorithms

对容器进行的各种操作，如排序，查找，反转等等，也可以理解为容器所具有的功能（成员函数）。

```
#include <algorithm>
```

find	查找指定的值
find_if	根据条件查找
reverse	反转
remove_if	根据条件删除相应的数据
transform	根据用户给定的方法对数据进行交换

容器常见成员函数

begin()	返回开始迭代器
end()	返回结束迭代器
size()	返回实际元素个数
capacity()	返回当前容量
empty()	判断是否为空
max_size()	返回元素个数的最大值
front()	返回第一个元素的引用
back()	返回最后一个元素的引用
push_back()	在序列的尾部添加一个元素
pop_back()	移出序列尾部的元素
clear()	移出所苻的元素，容器大小变为 0
resize()	改变实际元素的个数

容器常见成员函数（续）

at()	使用索引访问元素，会进行边界检查
assign()	用新元素替换原有内容
insert()	在指定的位置插入一个或多个元素
erase()	移出一个元素或一段元素
swap()	交换两个容器的所有元素
data()	返回包含元素的内部数组的指针
sort()	对元素进行排序

- 这里仅列出部分成员函数。
- 并非所有容器都具有这些成员函数。
- 容器不仅使用方便，而且效率也非常高，可代替数组。
- 优先使用 vector 和 string

3 迭代器 Iterators

访问容器中数据的方法，如指针。

要访问容器的数据，需要通过迭代器。

迭代器是算法与容器之间的桥梁。

- 实际上，迭代器就是一个泛型指针。
- 算法是作用在迭代器上，而不是容器上。
- 可以根据需要自己定义迭代器。

迭代器举例

Example

```
vector<int> x;

x.push_back(2);
x.push_back(0);
x.push_back(1);
x.push_back(9);
x.push_back(6);
x.push_back(13);

cout << "x: " << endl;

vector<int>::iterator idx = x.begin (); // 迭代器，类似于指针
while (idx != x.end ())
{   cout << *idx << endl;
    ++ idx;
}
```

迭代器举例（续）

ex16_iterator.cpp

```
vector<int>::iterator fidx;
fidx = find (x.begin (), x.end (), 9);
    // 寻找满足条件的数据，返回迭代器（类似于地址）
if (fidx != x.end ())
{
    int shift = distance(x.begin (), fidx);
      // 计算与开头元素的间距，即在向量中的位置
    cout << "要找的值为："<< *fidx << endl;
    cout << "位置:" << shift << endl;
}
```

谢 谢

THANK YOU