

原创 不良 于 2023-07-10 20:57:08 发布 阅读量448 收藏 32 点赞数 33 版权

分类专栏: C++ 文章标签: c++ 开发语言 后端



C++ 专栏收录该内容

2 订阅 21 篇文章

订阅专栏

目录

文章目录

- 认识deque
- stack简介
- stack常用接口
- stack模拟实现
- queue简介
- queue常用接口
- queue模拟实现

文

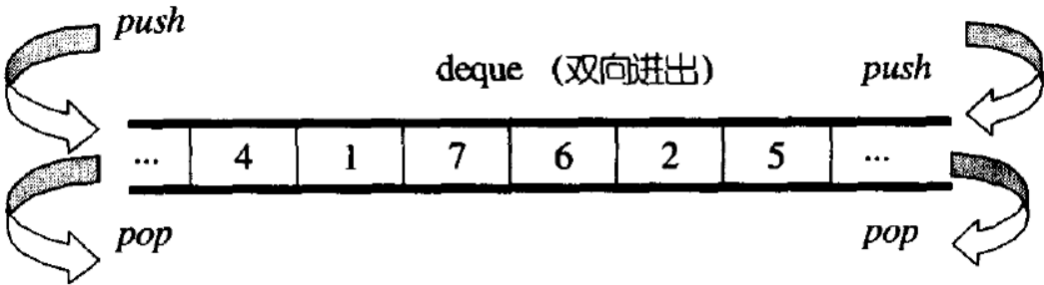
🚀 作者简介：一名在后端领域学习，并渴望能够学有所成的追梦人。
🌐 个人主页：不良
🔥 系列专栏：C++ Linux
📌 学习格言：博观而约取，厚积而薄发
🌹 欢迎进来的小伙伴，如果小伙伴们在学习的过程中，发现有需要纠正的地方，烦请指正，希望能够与诸君一同成长！🌹

文章目录

- 认识deque
- stack简介
- stack常用接口
- stack模拟实现
- queue简介
- queue常用接口
- queue模拟实现

认识 deque

deque(双端队列)：是一种双开口的"连续"空间的数据结构，双开口的含义是：可以在头尾两端进行插入和删除操作，且时间复杂度为O(1)，与vector比较，头插效率高，不需要搬移元素；与list比较，空间利用率比较高。



deque是双端队列，对标的是vector加list的合体。

```
1 | template < class T, class Alloc = allocator<T> > class deque;
```



不良

关注

33



32



27



专栏目录

vector的功能	
front	Access first element (public member function)
back	Access last element (public member function)

Modifiers:

assign	Assign container content (public member function)
push_back	Add element at the end (public member function)
push_front	Insert element at beginning (public member function)
pop_back	Delete last element (public member function)
pop_front	Delete first element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace	Construct and insert element (public member function)
emplace_front	Construct and insert element at beginning (public member function)
emplace_back	Construct and insert element at the end (public member function)

目录

文章目录

- 认识deque
- stack简介
- stack常用接口
- stack模拟实现
- queue简介
- queue常用接口
- queue模拟实现

vector的优缺点：

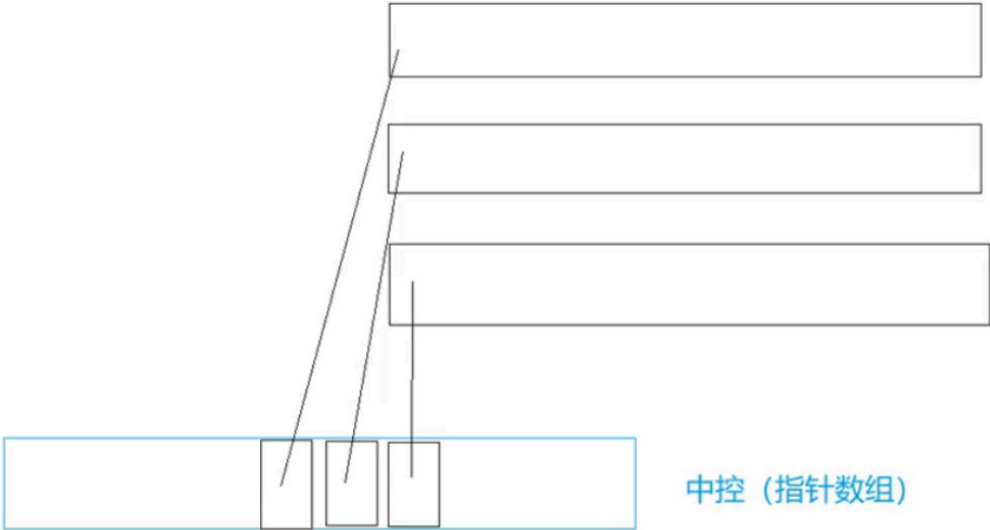
- 优点：支持下标随机访问；
- 缺点：头部或者中部插入删除效率低下；扩容

list的优缺点：

- 优点：任意位置插入删除效率高；
- 缺点：不能支持下标随机访问

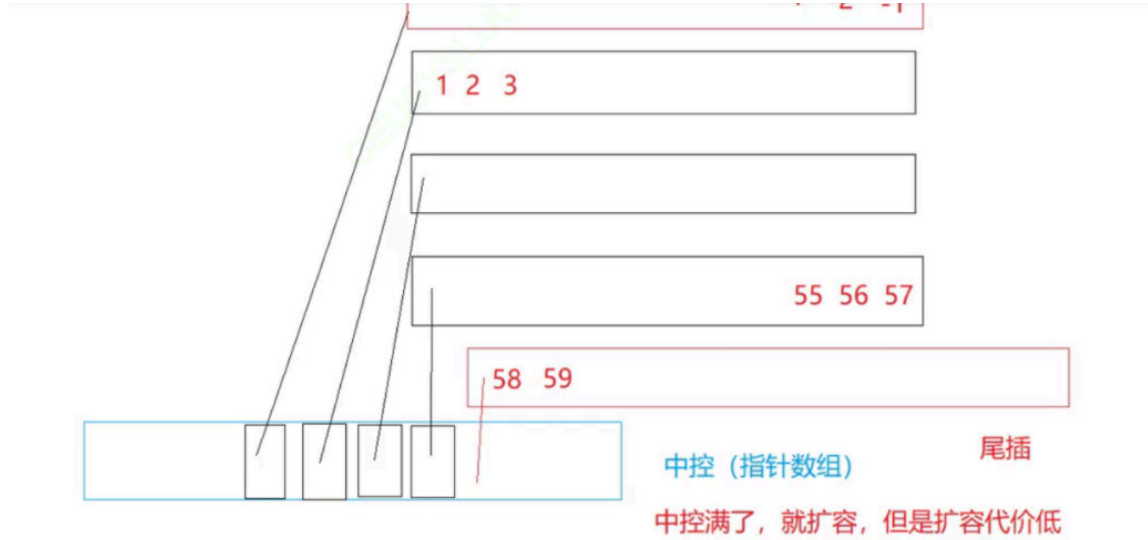
deque并不是真正连续的空间，而是由一段段连续的小空间拼接而成的，实际deque类似于一个动态的二维数组。

deque容器是开一个一个的小数组，当数组满了之后再开一个小数组；将这些小数组管理起来的数组是指针数组（中控（指针数组）），最开始使用的指针是中控指针数组中间位置的指针，当进行头插、尾插的时候，就可以直接使用前一个、后一个指针指向新开辟的空间。



中控满了之后就扩容，而且deque扩容给代价低，一个小数组对应一个中控数组中的指针，扩容代价低是因为只需要拷贝 **指针数组** 中的指针，然后再将原来的空间释放。

头插入数组尾部，尾插在数组头部：



目录

文章目录

认识deque

stack简介

stack常用接口

stack模拟实现

queue简介

queue常用接口

queue模拟实现

deque支持下标的随机访问，但是效率没有vector高。

deque的下标访问（每个数组大小固定）：假设每个小数组的容量为 10，我们想要找到下标为 25 的元素，用下标减去第一个数组内元素的个数，再除以每个小数组的容量就能找到其所在哪一个小数组。比如上图中就用 $(25 - 3) / 10$ 。找到对应元素存在于第 2 个数组后，再用 $(25 - 3) \% 10$ 就可以知道对应元素是在该小数组中的第几个。

deque容器中间插入删除时候很难搞，可以用size和capacity记录每个数组，可以每个数组不一样大，但是此时随机访问就麻烦了。

与vector比较，deque的优势是：头部插入和删除时，不需要移动元素，效率特别高，而且在扩容时，也不需要移动大量的元素，因此其效率是比vector高的。

与list比较，其底层是连续空间，空间利用率比较高，不需要存储额外字段。

所以deque优点有：

- 1. 相比vector，deque扩容代价低；
- 2. 头插头删、尾插尾删效率高；
- 3. 支持下标随机访问

缺点：

- 1.中间插入删除后很难搞（当每个数组不一样大时，中间插入删除的效率会变高但是随机访问的效率变低；当每个数组大小固定时，牺牲中间插入删除的效率，随机访问的效率就变高了）；
- 2.没有vector和list优点极致

但是，deque有一个致命缺陷：不适合遍历，因为在遍历时，deque的迭代器要频繁的去检测其是否移动到某段小空间的边界，导致效率低下，而序列式场景中，可能需要经常遍历，因此在实际中，需要线性结构时，大多数情况下优先考虑vector和list，deque的应用并不多，而目前能看到的一个应用就是，STL用其作为stack和queue的底层数据结构。

deque适用于什么情况？

如果头插头删多，尾插尾删多可以使用deque容器，所以很适合作为栈和队列的默认底层容器。所以默认作为栈和队列的底层适配容器。

stack是一种后进先出的特殊线性数据结构，因此只要具有push_back()和pop_back()操作的线性结构，都可以作为stack的底层容器，比如vector和list都可以；queue是先进先出的特殊线性数据结构，只要具有push_back和pop_front操作的线性结构，都可以作为queue的底层容器，比如list。但是STL中对stack和queue默认选择deque作为其底层容器，主要是因为：

stack和queue不需要遍历(因此stack和queue没有迭代器)，只需要在固定的一端或者两端进行操作。

在stack中元素增长时，deque比vector的效率(扩容时不需要搬移大量数据)；

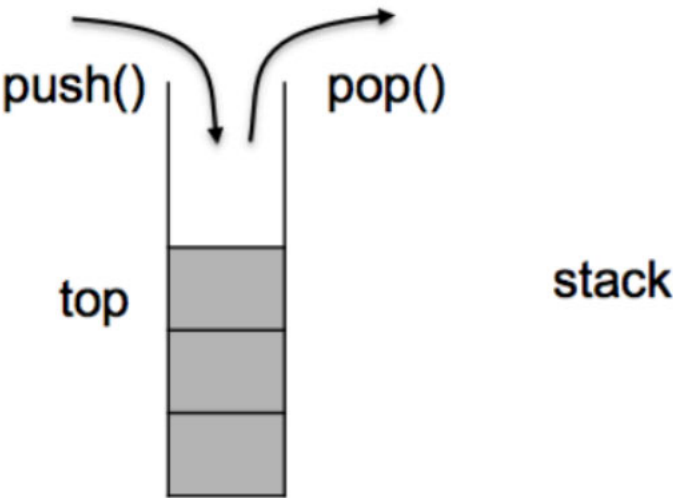
queue中的元素增长时，deque不仅效率高，而且内存使用率高。结合了deque的优点，而完美的避开了其缺陷

stack是作为容器适配器被实现的，容器适配器即是对特定类封装作为其底层的容器，并提供一组特定的成员函数来访问其元素，将特定类作为其底层的，元素特定容器的尾部(即栈顶)被压入和弹出。

stack的底层容器可以是任何标准的容器类模板或者一些其他特定的容器类，这些容器类应该支持以下操作：

- empty：判空操作
- back：获取尾部元素操作
- push_back：尾部插入元素操作
- pop_back：尾部删除元素操作

标准容器vector、deque、list均符合这些需求，默认情况下，如果没有为stack指定特定的底层容器，默认情况下使用deque。



stack常用接口

函数说明	接口说明
stack()	构造空的栈
empty()	检测stack是否为空
size()	返回stack中元素的个数
top()	返回栈顶元素的引用
push()	将元素val压入stack中
pop()	将stack中尾部的元素弹出

库中stack定义方式接口：

```
1 | template <class T, class Container = deque<T> > class stack;
```

缺省值为deque容器，说明底层容器为deque，我们也可以根据情况定义，所以定义方式有以下两种：

1.使用默认的适配器定义栈。

```
1 | stack<int> st; //定义一个存储int类型的栈
```

2.: 使用特定的适配器定义栈。

```
1 | stack<int, vector<int>>
2 | stack<int, list<int>> st
```

目录

文章目录

认识deque

stack简介

stack常用接口

stack模拟实现

queue简介

queue常用接口

queue模拟实现



不良

关注

33



32



27



专栏目录



```
1 #include <iostream>
2 #include <stack>
3 using namespace std;
4 int main()
5 {
6     stack<int> st;//构造
7     //判断是不是空
8     cout << st.empty() << endl;//0
9     //插入元素
10    for (int i = 0; i < 4; i++)
11    {
12        st.push(i);
13    }
14    //判断是不是空
15    cout << st.empty() << endl;//1
16    //打印栈中元素个数
17    cout << st.size() << endl;
18    //打印, stack不支持迭代器
19    for (int i = 0; i < 4; i++)
20    {
21        cout << st.top() << " "; //打印栈顶元素
22        st.pop();//删除
23    }
24    //输出3 2 1 0
25 }
```

目录

文章目录

认识deque

stack简介

stack常用接口

stack模拟实现

queue简介

queue常用接口

queue模拟实现

A

stack没有迭代器，所以遍历的时候不能使用范围for。

stack模拟实现

我们用适配器模式/配接器模式，本质是转换也就是把已有的东西进行转换，就好比我们的手机充电器并不能直接使用220v电压，所以提供了一个转换器，将电压转换为适合我们使用的。

设计模式就是指把常见的设计方法进行总结，适配器也是一种设计模式。

我们用已有的容器封装：可以这样定义类模板 `template<class T,class Container>`，Container就是符合我们要求的一个容器。

stack模拟实现主要依赖底层容器中的函数，所以stack模拟实现比较简单，将类模板中的第二个参数默认值为deque容器，目的是让deque作为默认底层容器。

```
1 namespace Niu {
2     //给类模板中的第二个参数默认值为deque容器
3     template<class T,class Container = deque<T>>
4     class stack {
5     public:
6         //实现push
7         void push(const T& val)
8         {
9             //调用容器的push_back函数
10            _con.push_back(val);
11        }
12        //实现pop
13        void pop()
14        {
15            //调用容器的pop_back函数
16            _con.pop_back();
17        }
18
19        const T& top()
20        {
21            //调用容器的back函数返回容器最后一个数据
22            return _con.back();
23        }
24        size_t size()
25        {
26            //调用容器的size函数
27            return _con.size();
28        }
29    };
30 }
```



不良

关注

33

32

27

27

27

27

27

专栏目录

```
30     {
31         //调用容器的empty函数
32         return _con.empty();
33     }
34     private:
35         Container _con;
36     };
37 }
```

目录

文章目录

认识deque

stack简介

stack常用接口

stack模拟实现

queue简介

queue常用接口

queue模拟实现



queue简介

队列是一种容器适配器，专门用于在FIFO上下文(先进先出)的环境中操作，其中从容器一端插入元素，另一端提取元素。

队列作为容器适配器实现，容器适配器即将特定容器类封装作为其底层容器类，queue提供一组特定的成员函数来访问其元素。元素从队尾入队列，从队头出队列。

底层容器可以是标准容器类模板之一，也可以是其他专门设计的容器类。该底层容器应至少支持以下操作：

- empty：检测队列是否为空
- size：返回队列中有效元素的个数
- front：返回队头元素的引用
- back：返回队尾元素的引用
- push_back：在队列尾部入队列
- pop_front：在队列头部出队列

标准容器类deque和list满足了这些要求。默认情况下，如果没有为queue实例化指定容器类，则使用标准容器deque。

queue常用接口

函数声明	接口说明
queue()	构造空的队列
empty()	检测队列是否为空，是返回true，否则返回false
size()	返回队列中有效元素的个数
front()	返回队头元素的引用
back()	返回队尾元素的引用
push()	在队尾将元素val入队列
pop()	将队头元素出队列

库中queue定义方式接口：

```
1 | template <class T, class Container = deque<T> > class queue;
```

缺省值为deque容器，说明底层容器为deque，我们也可以根据情况定义，所以定义方式有以下两种：

1.使用默认的适配器定义栈。

```
1 | queue<int> st;//定义一个存储int类型的栈
```

2.：使用特定的适配器定义栈。

注意queue中底层容器仅可使用deque和list容器，其他容器没有符合的对应操作。

```
1 | queue<int, list<int>> st;//定义一个用list<int>作为底层容器存储int类型的栈
```

队列能不能用vector适配？不能，v



不良

关注

33



32



27



专栏目录

```
1 #include <iostream>
2 #include <queue>
3 using namespace std;
4 int main()
5 {
6     queue<int> q;
7     //判断q队列中是不是为空
8     cout << q.empty() << endl;//0
9     // 插入元素
10    for (int i = 0; i < 5; i++)
11    {
12        q.push(i);
13    }
14    //获取队头元素
15    cout << q.front() << endl; //0
16    //获取队尾元素
17    cout << q.back() << endl; //4
18
19    //判断队列是否为空
20    cout << q.empty() << endl; //1
21    //队列中元素个数
22    cout << q.size() << endl; //5
23    //删除元素
24    for (int i = 0; i < 5; i++)
25    {
26        cout << q.front() << " ";
27        q.pop();//从队头开始删除
28    }
29    //输出0 1 2 3 4
30
31    return 0;
32 }
```

目录

文章目录

认识deque

stack简介

stack常用接口

stack模拟实现

queue简介

queue常用接口

queue模拟实现

queue模拟实现

queue模拟实现和stack模拟实现差不多，都是通过调用容器的函数来完成对应的功能。

模板声明和定义分离，在不同的文件中，要单独加模板参数，即便加了之后也可能会出错。

模板声明和定义分离会有很多的问题，会产生链接错误。所以模板都定义在.h中就可以，不会产生未知错误。

```
1 namespace Niu {
2     template<class T , class Container = deque<T>>
3     class queue {
4     public:
5         //判空
6         bool empty()
7         {
8             return _con.empty();
9         }
10        //在队尾插入元素
11        void push(const T& val)
12        {
13            _con.push_back(val);
14        }
15        //删除队头元素
16        void pop()
17        {
18            _con.pop_front();
19        }
20        //返回size大小
21        size_t size()
22        {
23            return _con.size();
24        }
25        //返回队头元素
26        T& front()
27        {
28
```



不良

关注

👍 33



🌟 32



💬 27



专栏目录


```
31 //返回队头元素
32 const T& front() const
33 {
34     return _con.front();
35 }
36 //返回队尾元素
37
38 const T& back() const
39 {
40     return _con.back();
41 }
42 //返回队尾元素
43 T& back()
44 {
45     return _con.back();
46 }
47 private:
48     Container _con;
49 };
}
```

目录

文章目录

- 认识deque
- stack简介
- stack常用接口
- stack模拟实现
- queue简介
- queue常用接口
- queue模拟实现



C++ STL容器stack和queue详解 09-01

主要介绍了C++ STL容器stack和queue详解的相关资料,需要的朋友可以参考下

C++提高编程——STL：deque容器、stack容器和queue容器 weixin_63866037的博客 901

deque容器和vector容器的构造方式几乎一致，灵活使用即可。

27 条评论 rygttm 热评 支持我佬 写评论

【C++】STL——容器适配器 stack和queue 深度剖析及模拟实现 & 适配器... 5-31

但是STL中对stack和queue默认选择deque作为其底层容器,主要是因为: stack和queue不需要遍历(因此stack和queue没有迭代器),...

C++ STL——栈和队列(stack & queue)_c++ 队列实现 5-29

1.stack()是构造一个空的stack 2.empty()是用于判断stack是否为空 3.size()是用来返回stack中元素的个数 4.top()适用于返回栈顶元...

【C++】STL之适配器---用deque实现栈和队列 m0_63198468的博客 2422

本介绍了什么是适配器， deque，并且使用queue去实现了stack和queue，重点介绍了他们函数接口的使用，以及后面的stack和qu...

C++STL库的 deque、stack、queue、list、set/multiset、map/multimap weixin_50963877的博客 1633

Deque 最大的工作就是维护这些分段连续的内存空间的整体性的假象，并提供随机内存访问的接口，避免了重新配置空间，复制，释放...

【C++进阶】STL容器--stack和queue深度剖析&&优先队列&&适配器原理... 5-21

6. priority_queue的实现 前言 栈和队列在C语言中大家都有所了解,C语言的栈和队列都是我们手动去实现,而C++中的栈和队列不同,...

C++——STL标准模板库——容器详解——stack+queue_stl顺序栈类模板定 ... 5-24

stack<T>::stack(stack<T>&&s); 移动构造 stack<T,TypeContainer<T>>::stack(TypeContainer&c); 利用基础容器构造堆栈(二)queue...

STL——Stack和Queue 冯同学的博客 475

理解什么是适配器，学会使用stack和queue，以及实现两者模拟实现

STL——stack和queue TheBao0107的博客 140

1. stack是一种容器适配器，专门用在具有后进先出操作的上下文环境中，其删除只能从容器的一端进行元素的插入与提取操作。2. ...

【c++】STL之stack和queue详解_cplusplus stack 5-26

stack - C++ Reference (cplusplus.com) queue - C++ Reference (cplusplus.com) ★主体 在数据结构初阶中,我们模拟实现了stack...

C++STL的stack和queue(超详解) 5-12

queue的模拟实现 前言 栈和队列这一块其实有数据结构的基础,学起来非常简单。 stack 栈的成员函数就这么写,除了emplace其他都...

【STL】stack和queue的实现原理 qq_43142509的博客 861


文章目录






STL的stack和queue 汐风的博客 275

STL的stack和queue的模拟实现，以及deque的简单介绍和它与vector，list的对比

C++【STL】之stack和queue学习_标准库队列 5-15

1.2 STL标准库中stack和queue的底层结

 不良 关注

 33   32  27 

专栏目录