6.9 迭代器_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 6.9 迭代器涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

大白老师 · 更新于 2022-11-28

上一节

6.8 容器设配器之 stack、priority_queue

6.10 函数对象

下一节

9. 迭代器

一、介绍

迭代器用于遍历对象集合的元素，从实现角度来看，** 迭代器是一种将 ****operator*、operator->、operator++、operator—** 等指针操作予以重载的 class template

二、内部实现分析

```
#ifndef _LIST_HEAD_H
#define _LIST_HEAD_H
#include <iostream>

using namespace std;

template <typename T>
class ListNode{

public:
    ListNode(T value,ListNode *node):value(value),next(node){}
    T value;
    ListNode *next;
};
```

```
template <typename T>
class List{
public:
    List();
    void pushFront(const T &value);
    void pushBack(const T &value);
    void printList(void);

    class iterator{
    public:
       iterator(ListNode<T> *ptr);
       iterator &operator++();
       iterator  operator++(int);
       ListNode<T> *operator->()const;
       T operator*()const;
       bool operator==(const iterator &other)const;
       bool operator!=(const iterator &other)const;

    private:
            ListNode<T> *ptr;
    };

    iterator begin(void)const;
    iterator end(void)const;

private:
    ListNode<T> *m_head;
    ListNode<T> *m_tail;
};



template <typename T>
List<T>::List()
{
    m_head = NULL;
    m_tail = NULL;
}

template <typename T>
void List<T>::pushBack(const T &value)
{
    if(m_tail == NULL){
        m_tail = new ListNode<T>(value,NULL);
```

```
        m_head = m_tail;
    }else{
        ListNode<T> *ptr = new ListNode<T>(value,NULL);
        m_tail->next = ptr;
        m_tail = ptr;
    }

    return;
}

template <typename T>
void List<T>::pushFront(const T &value)
{
    if(m_head == NULL){
        m_head = new ListNode<T>(value,NULL);
        m_tail = m_head;
    }else{
        ListNode<T> *ptr = m_head;
        m_head = new ListNode<T>(value,ptr);
    }

    return;
}

template <typename T>
void List<T>::printList(void){
    for(ListNode<T> *ptr = m_head;ptr != NULL;ptr = ptr->next){
        cout << ptr->value << " " ;
    }
    cout << endl;
}

template<typename T>
typename  List<T>::iterator List<T>::begin(void) const
{
    return List<T>::iterator(m_head);
}

template<typename T>
typename List<T>::iterator List<T>::end(void) const
{
    return List<T>::iterator(m_tail->next);
}
```

```
template<typename T>
List<T>::iterator::iterator(ListNode<T> * ptr):ptr(ptr)
{

}

template<typename T>
typename List<T>::iterator& List<T>::iterator::operator++()
{
    ptr = ptr->next;
    return *this;
}

template<typename T>
typename List<T>::iterator List<T>::iterator::operator++(int)
{
    List<T>::iterator old(*this);

    ptr =ptr->next;
    return old;
}

template<typename T>
ListNode<T> * List<T>::iterator::operator->()const
{
    return ptr;
}

template<typename T>
T List<T>::iterator::operator*()const
{
    return ptr->value;
}

template<typename T>
bool List<T>::iterator::operator==(const List<T>::iterator &other)const
{
    return ptr == other.ptr;
}

template<typename T>
bool List<T>::iterator::operator!=(const List<T>::iterator &other)const
{
```

```
    return ptr != other.ptr;

}

#endif
#include "list.hpp"


int main(void)

{

    List<int> list;


    list.pushBack(1);

    list.pushBack(2);

    list.pushBack(3);

    list.pushFront(4);

    list.printList();


    for(List<int>::iterator it = list.begin();it != list.end();it ++){

       cout << *it << endl;

    }


    return 0;

}
```

上一节

6.8 容器设配器之 stack、priority_queue

下一节

6.10 函数对象

我要提出意见反馈