第七周上机

窦闻

2025-4-3

1. 完成以下程序中声明的所有未实现的函数. 代码保存为 lab6-1.cpp. 提示: 先注释掉主函数中所有未实现的函数调用, 逐个函数实现后再在主函数中进行测试.

#include <iostream>  
using namespace std;  
   
// 定义链表结点类型  
struct Node  
{  
 double value;  
 Node \*next{nullptr};  
};  
   
void printList(Node \*head); // 打印链表  
void dropList(Node \*head); // 清除链表  
int get\_len(Node \*head); // 获取链表长度  
void concatList(Node \*head1, Node \*head2); // 将第二个链表连接到第一个链表尾部  
void shiftList(Node \*&head, int n); // 循环左移 n 位  
void printNode(Node \*pNode); // 打印结点的数据  
void rm\_node(Node \*&head, int idx); // 根据索引删除结点  
void push(Node \*&head, Node \*pNode); // 将结点压入链表  
Node \*pop(Node \*&head); // 弹出头结点  
Node \*pop(Node \*&head, int idx); // 弹出 idx 位置的结点  
Node \*fetch(Node \*head, int idx); // 获取 idx 位置的结点  
Node \*fetch\_prev(Node \*head, int idx); // 获取 idx 位置结点的前驱结点  
Node \*fetch\_last(Node \*head); // 获取尾结点  
Node \*search(Node \*head, double x); // 根据值查找结点  
   
int main()  
{  
 // 创建空链表  
 Node \*head = nullptr;  
   
 Node \*aNode;  
   
 // 添加结点  
 for (int i = 10; i >= 0; --i)  
 {  
 aNode = new Node; // 创建新结点  
 aNode->value = i; // 存储值  
 push(head, aNode);  
 }  
 printList(head); // 输出: 0 1 2 3 4 5 6 7 8 9 10  
 cout << "Length: " << get\_len(head) << endl; // 输出：Length: 11  
   
 // 弹出指定位置结点  
 printNode(pop(head, 7)); // 输出: 7  
 printList(head); // 输出: 0 1 2 3 4 5 6 8 9 10  
   
 // 输出值为6的结点后继的值  
 cout << search(head, 6)->next->value << endl; // 输出: 8  
   
 // 弹出结点  
 printNode(pop(head)); // 输出: 0  
 printList(head); // 输出: 1 2 3 4 5 6 8 9 10  
   
 // 移除结点  
 rm\_node(head, 7);  
 printList(head); // 输出: 1 2 3 4 5 6 8 10  
   
 // 获取指定位置的结点  
 printNode(fetch(head, 2)); // 输出: 3  
 printNode(fetch(head, 6)); // 输出: 8  
 printNode(fetch(head, 10)); // 输出: Invalid node  
   
 // 获取尾结点  
 printNode(fetch\_last(head)); // 输出: 10  
   
 // 新建列表  
 Node \*head2 = nullptr;  
   
 // 添加结点  
 for (int i = 3; i >= 0; --i)  
 {  
 aNode = new Node; // 创建新结点  
 aNode->value = i \* 10; // 存储值  
 push(head2, aNode);  
 }  
 printList(head2); // 输出: 0 10 20 30  
   
 // 链表连接  
 concatList(head, head2);  
 printList(head); // 输出: 1 2 3 4 5 6 8 10 0 10 20 30  
   
 // 链表左移  
 shiftList(head, 3);  
 printList(head); // 输出: 4 5 6 8 10 0 10 20 30 1 2 3  
   
 // 链表右移  
 shiftList(head, -4);  
 printList(head); // 输出: 30 1 2 3 4 5 6 8 10 0 10 20  
   
 // 删除列表  
 dropList(head); // 输出: List has been DROPPED!  
   
 return 0;  
}  
   
   
void printList(Node \*head)  
{  
 Node \*pCur = head;  
 while (pCur)  
 {  
 cout << pCur->value << ' ';  
 pCur = pCur->next;  
 }  
 cout << endl;  
 return;  
}  
   
int get\_len(Node \*head)  
{  
 Node \*pCur = head;  
 if (!head)  
 return 0;  
 int len = 1;  
 while (pCur->next)  
 {  
 pCur = pCur->next;  
 ++len;  
 }  
 return len;  
}  
   
void printNode(Node \*pNode)  
{  
 if (pNode)  
 cout << pNode->value << endl;  
 else  
 cout << "Invalid node" << endl;  
}  
   
Node \*fetch\_prev(Node \*head, int idx)  
{  
 if (idx <= 0)  
 return nullptr;  
 return fetch(head, idx - 1);  
}  
   
void rm\_node(Node \*&head, int idx)  
{  
 delete pop(head, idx);  
 return;  
}  
   
void concatList(Node \*head1, Node \*head2)  
{  
 fetch\_last(head1)->next = head2;  
 return;  
}