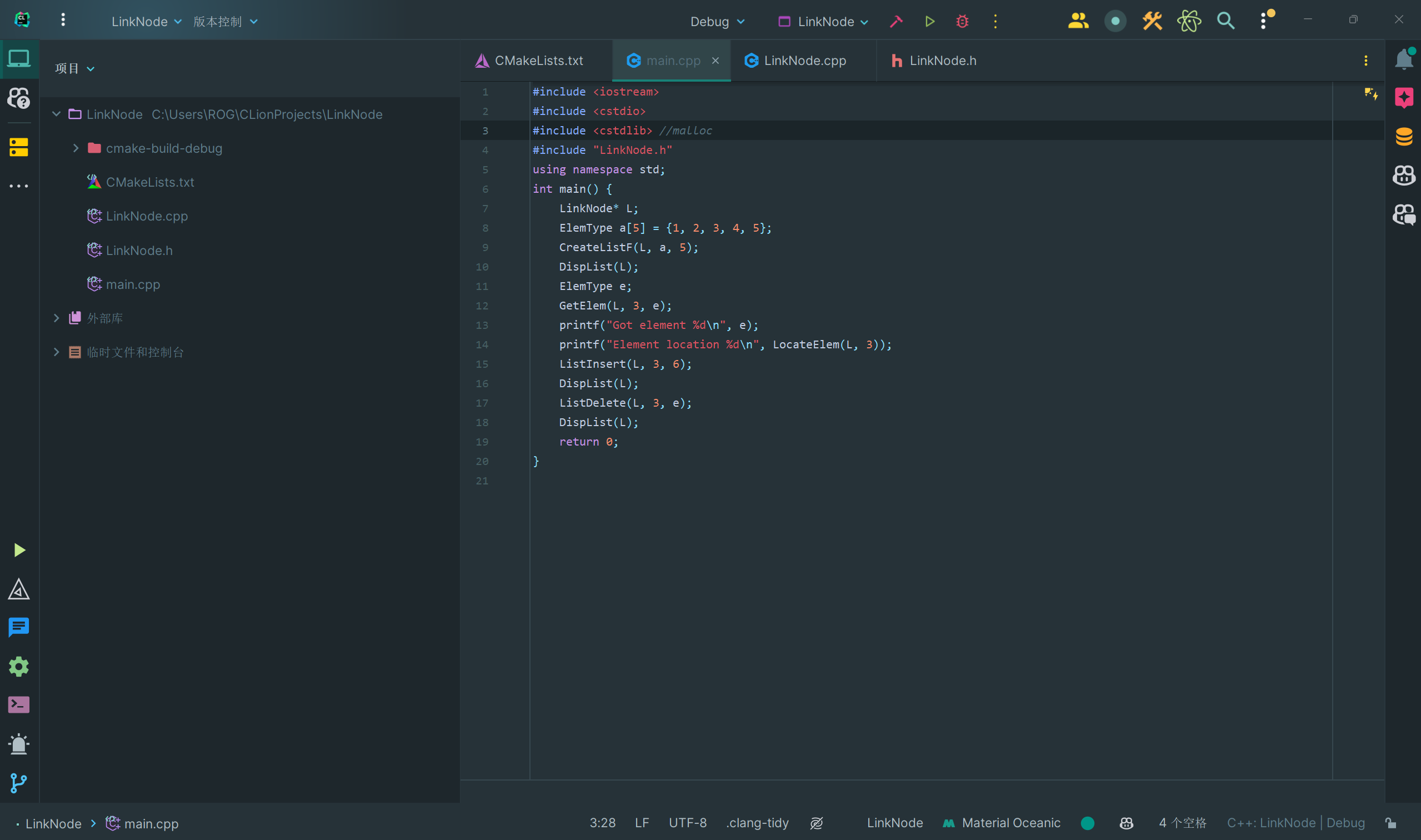
单链表的创建和基本操作

数据结构作业 2024.3.16

以下是截图

电脑萤幕的截图

描述已自动生成电脑萤幕的截图

描述已自动生成电脑软件截图

描述已自动生成

以下是源代码

LinkNode.h

*//  
// Created by Devil\_Zero on 2024/3/16.  
//*#ifndef LINKNODE\_LINKNODE\_H  
#define LINKNODE\_LINKNODE\_H  
typedef int ElemType;  
typedef struct LNode{  
 ElemType data;  
 struct LNode\* next;  
}LinkNode;  
void CreateListF(LinkNode\*& L, ElemType a[], int n);  
void CreateListR(LinkNode\*& L, ElemType a[], int n);  
bool InitList(LinkNode\*& L);  
bool DestroyList(LinkNode\*& L);  
bool ListEmpty(LinkNode\* L);  
int ListLength(LinkNode\* L);  
void DispList(LinkNode\* L);  
bool GetElem(LinkNode \*L, int i, ElemType &e);  
bool LocateElem(LinkNode\* L, ElemType e);  
bool ListInsert(LinkNode\* L,int i, ElemType e);  
bool ListDelete(LinkNode\* L, int i, ElemType &e);  
#endif *//LINKNODE\_LINKNODE\_H*

LinkNode.cpp

#include <iostream>  
#include "LinkNode.h"  
void CreateListF(LinkNode\*& L, ElemType a[], int n){  
 LinkNode\* s;  
 L = (LinkNode\*)malloc(sizeof(LinkNode));  
 L->next = NULL;  
 for(int i = 0; i < n; i++){  
 s = (LinkNode\*)malloc(sizeof(LinkNode));  
 s->data = a[i];  
 s->next = L->next;  
 L->next = s;  
 }  
}  
void CreateListR(LinkNode\*& L, ElemType a[], int n){  
 LinkNode\* s, \*r;  
 r = L;  
 for(int i = 0; i < n; i++){  
 s = (LinkNode\*)malloc(sizeof(LinkNode));  
 s->data = a[i];  
 r->next = s;  
 r = s;  
 r -> next = NULL;  
 }  
 r->next = NULL;  
}  
bool InitList(LinkNode\*& L){  
 L = (LinkNode\*)malloc(sizeof(LinkNode));  
 if(L == NULL) return false;  
 L -> next = NULL;  
 return true;  
}  
bool DestroyList(LinkNode\*& L){  
 LinkNode\* pre = L, \*p = L -> next;  
 while(p != NULL){  
 free(pre);  
 pre = p;  
 p = pre -> next;  
 }  
 free(pre);  
 return true;  
}  
bool ListEmpty(LinkNode\* L){  
 return (L -> next == NULL);  
}  
int ListLength(LinkNode\* L){  
 int n = 0;  
 LinkNode\* p = L;  
 while(p -> next != NULL){  
 n++;  
 p = p -> next;  
 }  
 return n;  
}  
void DispList(LinkNode\* L){  
 LinkNode\* p = L -> next;  
 while(p != NULL){  
 printf("%d ", p -> data);  
 p = p -> next;  
 }  
 printf("\n");  
}  
bool GetElem(LinkNode \*L, int i, ElemType &e){  
 int j = 0;  
 LinkNode\* p = L;  
 if(i <= 0) return false;  
 while(j < i && p != NULL){  
 j++;  
 p = p -> next;  
 }  
 if(p == NULL) return false;  
 else{  
 e = p -> data;  
 return true;  
 }  
}  
bool LocateElem(LinkNode\* L, ElemType e){  
 int i = 1;  
 LinkNode\* p = L -> next;  
 while(p != NULL && p -> data != e){  
 p = p -> next;  
 i++;  
 }  
 if(p == NULL) return 0;  
 else return i;  
}  
bool ListInsert(LinkNode\* L,int i, ElemType e){  
 int j = 0;  
 LinkNode\* p = L, \*s;  
 if(i <= 0) return false;  
 while(j < i - 1 && p != NULL){  
 j++;  
 p = p -> next;  
 }  
 if(p == NULL) return false;  
 else{  
 s = (LinkNode\*)malloc(sizeof(LinkNode));  
 s -> data = e;  
 s -> next = p -> next;  
 p -> next = s;  
 return true;  
 }  
}  
bool ListDelete(LinkNode\* L, int i, ElemType &e){  
 int j = 0;  
 LinkNode \*p = L, \*q;  
 if(i <= 0) return false;  
 while(j < i - 1 && p != NULL){  
 j++;  
 p = p -> next;  
 }  
 if(p == NULL) return false;  
 else{  
 q = p -> next;  
 if(q == NULL) return false;  
 e = q -> data;  
 p -> next = q -> next;  
 free(q);  
 return true;  
 }  
}

main.cpp

#include <iostream>  
#include <cstdio>  
#include <cstdlib> *//malloc*#include "LinkNode.h"  
using namespace std;  
int main() {  
 LinkNode\* L;  
 ElemType a[5] = {1, 2, 3, 4, 5};  
 CreateListF(L, a, 5);

//CreateListR(L, a, 5);  
 DispList(L);  
 ElemType e;  
 GetElem(L, 3, e);  
 printf("Got element %d\n", e);  
 printf("Element location %d\n", LocateElem(L, 3));  
 ListInsert(L, 3, 6);  
 DispList(L);  
 ListDelete(L, 3, e);  
 DispList(L);  
 return 0;  
}