# ****《数据结构》实验报告****

|  |  |
| --- | --- |
| 姓名：    高心阳 | 学号：    084623237 |
| 班级：   计算机232 | 日期：   2024/9/4 |
| 程序名：          带模板的顺序表 | |

### ****一、上机实验的问题和要求：****

将实验指导书中的顺序表类改为模板类并实现

### ****二、程序设计的基本思想，原理和算法描述：****

（包括程序的结构，数据结构，输入/输出设计，符号名说明等）

思想：将普通的顺序表头文件改写成模板类

原理：将作为数据的int类型全部替换为模板类。

### ****三、源程序及注释（请将自己创建的.h和.cpp文件内容直接复制到这里）：****

|  |
| --- |
| SeqList.h |
| #pragma once  const int MaxSize = 20;  template<class DataType>  class SeqList {  private:  DataType data[MaxSize];  int length;  public:  SeqList() { length = 0; }  SeqList(DataType a[], int n);  ~SeqList(){}  void Insert(int i, DataType x);  DataType Delete(int i);  int Locate(DataType x);  void PrintList();  };  template<class DataType>  SeqList<DataType>::SeqList(DataType a[], int n) {  if (n > MaxSize)throw"参数非法";  for (int i = 0; i < n; i++) {  data[i] = a[i];  }  length = n;  }  template<class DataType>  void SeqList<DataType>::Insert(int i, DataType x) {  if (length > MaxSize)throw"上溢";  if (i<1 || i>length + 1)throw"位置非法";  for (int j = length; j >= i; j--) {  data[j] = data[j - 1];  }  data[i - 1] = x;  length++;  }  template<class DataType>  DataType SeqList<DataType>::Delete(int i) {  if (length == 0)throw"下溢";  if (i<1 || i>length)throw"位置非法";  DataType x = data[i - 1];  for (int j = i; j < length; j++) {  data[j - 1] = data[j];  }  length--;  return x;  }  template<class DataType>  int SeqList<DataType>::Locate(DataType x) {  for (int i = 0; i < length; i++) {  if (data[i] == x)return i + 1;  }  return 0;  }  template<class DataType>  void SeqList<DataType>::PrintList() {  for (int i = 0; i < length; i++) {  cout << data[i] << " ";  }  cout << endl;  } |

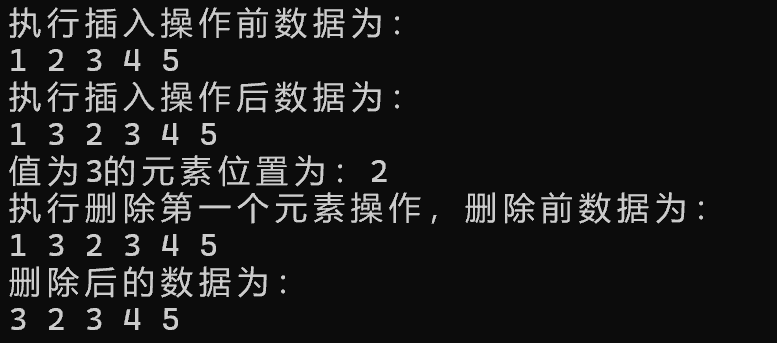
|  |
| --- |
| SeqList\_main.cpp |
| #include <iostream>  using namespace std;  #include "SeqList.h"  int main() {  int r[5] = { 1,2,3,4,5 };  SeqList<int> L(r, 5);  cout << "执行插入操作前数据为：" << endl;  L.PrintList();  try {  L.Insert(2, 3);  }  catch (char\* s) {  cout << s << endl;  }  cout << "执行插入操作后数据为：" << endl;  L.PrintList();  cout << "值为3的元素位置为：";  cout<<L.Locate(3)<<endl;  cout << "执行删除第一个元素操作，删除前数据为：" << endl;  L.PrintList();  try {  L.Delete(1);  }  catch (char\* s) {  cout << s << endl;  }  cout << "删除后的数据为：" << endl;  L.PrintList();  return 0;  } |

|  |
| --- |
| LinkList.h |
| #pragma once  template<class DataType>  struct Node {  DataType data;  Node<DataType>\* next;  };  template<class DataType>  class LinkList {  private:  Node<DataType>\* first;  public:  LinkList();  LinkList(DataType a[], int n);  ~LinkList();  int Locate(DataType x);  void Insert(int i, DataType x);  DataType Delete(int i);  void PrintList();  };  template<class DataType>  LinkList<DataType>::LinkList() {  first = new Node<DataType>;  first->next = NULL;  }  template<class DataType>  LinkList<DataType>::LinkList(DataType a[], int n) {  Node<DataType>\* r, \* s;  first = new Node<DataType>;  r = first;  for (int i = 0; i < n; i++) {  s = new Node<DataType>;  s->data = a[i];  r->next = s;  r = s;  }  r->next = NULL;  }  template<class DataType>  LinkList<DataType>::~LinkList() {  Node<DataType>\* q = NULL;  while (first != NULL) {  q = first;  first = first->next;  delete q;  }  }  template<class DataType>  int LinkList<DataType>::Locate(DataType x) {  Node<DataType>\* q = first->next;  int count = 1;  while (q != NULL) {  if (q->data == x)return count;  q = q->next;  count++;  }  return 0;  }  template<class DataType>  void LinkList<DataType>::Insert(int i, DataType x) {  Node<DataType>\* p = first, \* s = NULL;  int count = 0;  while (p != NULL && count < i - 1) {  p = p->next;  count++;  }  if (p == NULL)throw"非法位置";  else {  s = new Node<DataType>;  s->data = x;  s->next = p->next;  p->next = s;  }  }  template<class DataType>  DataType LinkList<DataType>::Delete(int i) {  Node<DataType>\* p = first, \* q = NULL;  DataType x;  int count = 0;  while (p != NULL && count < i - 1) {  p = p->next;  count++;  }  if (p == NULL || p->next == NULL)throw"位置非法";  else {  q = p->next;  x = q->data;  p->next = q->next;  delete q;  return x;  }  }  template<class DataType>  void LinkList<DataType>::PrintList() {  Node<DataType>\* p = first->next;  while (p != NULL) {  cout << p->data << " ";  p = p->next;  }  cout << endl;  } |

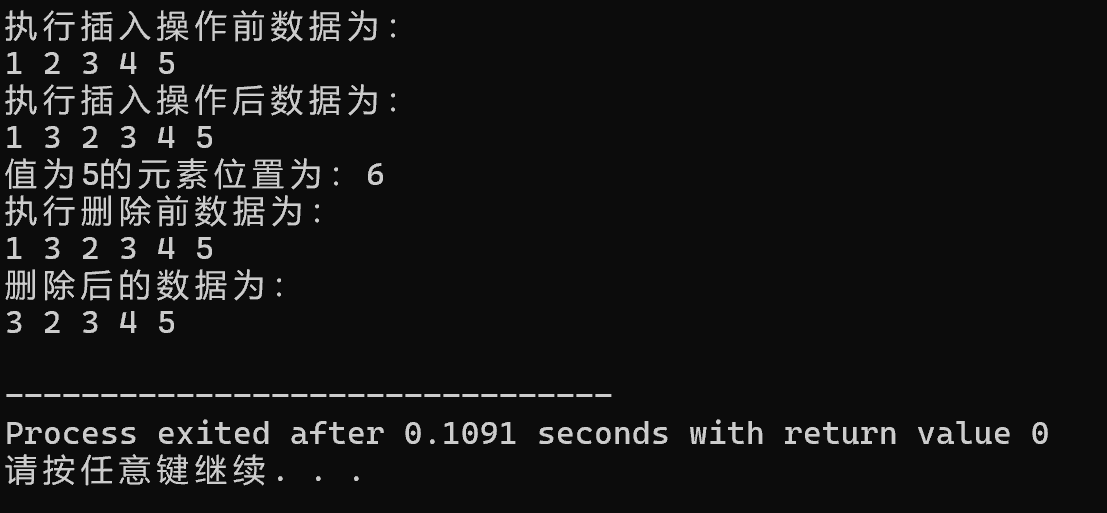
|  |
| --- |
| LinkList\_main.cpp |
| #include <iostream>  using namespace std;  #include "LinkList.h"  int main() {  int r[5] = { 1,2,3,4,5 };  LinkList<int> L(r, 5);  cout << "执行插入操作前数据为：" << endl;  L.PrintList();  try {  L.Insert(2, 3);  }  catch (char\* s) {  cout << s << endl;  }  cout << "执行插入操作后数据为：" << endl;  L.PrintList();  cout << "值为5的元素位置为：";  cout << L.Locate(5) << endl;  cout << "执行删除前数据为：" << endl;  L.PrintList();  try {  L.Delete(1);  }  catch (char\* s) {  cout << s << endl;  }  cout << "删除后的数据为：" << endl;  L.PrintList();  return 0;  return 0;  } |

### ****四、运行输出结果：（用谷歌浏览器，可以直接将截图粘贴到编辑器中）****

（可以将运行结果抓图贴至此处）



Seqlist\_main.cpp运行结果



Linklist\_main.cpp运行结果

### ****调试和运行程序过程中产生的问题及采取的措施：****

1. 问题：多个main函数位于同一个项目中无法正常编译运行  
   措施：分成两个项目进行编译运行
2. 问题：带模板的类的函数在类外定义时，错误添加class  
   措施：删除多余的class

### ****对算法的程序的讨论、分析，改进设想，其它经验教训：****

看清代码结构，不要写多余的内容，如多余的class。

复习类模板的写法。

复习指针的各种操作。

### ****对数据结构教学的意见和建议：****

无。