《数据结构》实验报告

**实验一 顺序表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 马昆 | 班级 | 软件工程专升本1班 | 学号 | 2206831544 |
| 实验名称 | **顺序表** | | | | |
| 实验时间 | 2023年4月27日 | | 成绩 | |  |
| 1. **实验目的：**   1.掌握线性表的顺序存储结构；  2.验证顺序表及其基本操作的实现；  3.理解算法与程序的关系，能够将顺序表算法转换为对应的程序。  **二、实验工具：**  软件平台：Windows11 专业工作站版，Visual Studio 2022  **三、实验原理：**  1、算法描述：  （1）显示函数：在顺序表类中添加一个成员函数PrintLIst()，通过for循环遍历顺序表数据数组并以C++静态初始化数据的格式打印其中的数据。  （2）查找函数：通过下标获取数据，第一个元素下标为1，如果查找的下标不在1~最大长度就抛出异常，如果在其范围内就返回数据数据的对应下标-1的元素。  （3）插入函数：如果数据数组长度长度已满就抛出异常，如果没满就在数据数组的最末尾添加数据，并将成员变量length加1。  （4）删除函数：如果删除的位置不在1~数据数组长度内，就抛出异常，如果在其内，就将数据数组中对应下标位置-1（不包括）后的数据都向前挪一位，并将成员变量length减1。  **四、实验步骤和内容：**  1.建立一个顺序表，随机产生10个100以内的整数，并按要求完成：  （1）编写显示函数，在屏幕上显示顺序表中的10个整数；  （2）编写查找函数，从键盘输入任一整数在顺序表中查找，若找到，返回该元素在顺序表中的位置，否则提示无此元素；  （3）编写插入函数，从键盘输入待插入元素及插入位置，将完成插入后的顺序表输出；  （4）编写删除函数，从键盘输入待删除元素位置，将该位置元素删除后的顺序表输出。  **代码**  **SeqList.h**  #pragma once  #include <iostream>  using namespace std;  const int MaxSize = 200; // 设置顺序表最多能存储的元素的数量  template <typename DataType = int> // 相当于Java的泛型，在C++叫模板  class SeqList  {  public:  SeqList(); // 无参构造，建立空的顺序表  SeqList(DataType a[], int n); // 有参构造，建立长度为n的顺序表  ~SeqList(); // 析构函数，就是在对象销毁时会被执行  int Length(); // 获取线性表的长度  DataType Get(int i); // 按位查找，查找第i个元素  int Locate(DataType x); // 按值查找，查找值为x的元素  void Insert(int i, DataType x); // 插入操作，在第i个位置插入值为x的元素  DataType Delete(int i); // 删除操作，删除第i个元素  int Empty(); // 判断线性表是否为空  void PrintList(); // 遍历操作，按序号依次输出各元素  private:  DataType data[MaxSize]; // 存放数据元素的数组  int length; // 线性表的长度  };  // 无参构造，建立空的顺序表  template <typename DataType>  SeqList<DataType>::SeqList() {  this->length = 0;  };  // 有参构造，建立长度为n的顺序表  template <typename DataType>  SeqList<DataType>::SeqList(DataType a[], int n) {  if (n > MaxSize) throw "参数非法";  for (int i = 0; i < n; i++) {  this->data[i] = a[i];  }  this->length = n;  }  // 析构函数，就是在对象销毁时会被执行  template <typename DataType>  SeqList<DataType>::~SeqList() {  cout << "被销毁了" << endl;  }  // 获取线性表的长度  template <typename DataType>  int SeqList<DataType>::Length() {  return this->length;  }  // 按位查找，查找第i个元素  template <typename DataType>  DataType SeqList<DataType>::Get(int i) {  if (i < 1 || i > this->length) {  throw "位置不存在";  }  return this->data[i - 1];  }  // 按值查找，查找值为x的元素  // 0：没找到  template <typename DataType>  int SeqList<DataType>::Locate(DataType x) {  for (int i = 0; i < length; i++) {  if (this->data[i] == x) {  return i + 1;  }  }  return 0;  }  // 插入操作，在第i个位置插入值为x的元素  template <typename DataType>  void SeqList<DataType>::Insert(int i, DataType x) {  if (length >= MaxSize) throw "已满";  if (i < 1 || i > length + 1) throw "位置错误";  for (int j = length; j >= i - 1; j--) {  this->data[j + 1] = this->data[j];  }  this->data[i - 1] = x;  this->length++;  }  // 删除操作，删除第i个元素  template <typename DataType>  DataType SeqList<DataType>::Delete(int i) {  if (i < 1 || i > length) throw "位置错误";  DataType del = this->Get(i);  for (int j = i - 1; j < length - 1; j++) {  this->data[j] = this->data[j + 1];  }  this->length--;  return del;  }  // 判断线性表是否为空  template <typename DataType>  int SeqList<DataType>::Empty() {  return this->length ? 0 : 1;  }  // 遍历操作，按序号依次输出各元素  template <typename DataType>  void SeqList<DataType>::PrintList() {  if (this->Empty()) {  cout << "{}" << endl;  return;  }  int i = 0;  cout << "{";  for (; i < length - 1; i++) {  cout << this->data[i] << ",";  }  cout << this->data[i] << "}" << endl;  }  **Main.cpp**  #include <iostream>  #include "SeqList.h"  #include <random>  using namespace std;  // 菜单  template <typename DataType = int>  void Menu(int choose, SeqList<>\* list);  int main() {  default\_random\_engine engine; // 创建随机数生成引擎  engine.seed(time(0)); // 设置种子  // 随机10个0~100的数字  int nums[10]{};  for (int i = 0; i < 10; i++) {  nums[i] = engine() % 101;  }  // 创建单链表  SeqList<> list = SeqList<>(nums, 10);  // 打印单链表  list.PrintList();  int choose = -1;  while (choose) {  cout << "--------------------------------" << endl;  cout << "元素[" << list.Length() << "]个|最大容量[" << MaxSize << "]个|顺序表状态";  list.PrintList();  cout << "1、插入一个数字" << endl;  cout << "2、指定位置插入一个数字" << endl;  cout << "3、获取某个数的具体位置" << endl;  cout << "4、获取一个具体位置的数" << endl;  cout << "5、删除一个指定位置的数" << endl;  cout << "0、退出演示" << endl;  cout << "请输入对应服务编号：";  cin >> choose;  try {  Menu(choose, &list);  }  catch (const char\* e) {  cout << "出错啦，错误消息：" << e << endl;  }  cout << endl << "--------------------------------" << endl;  }  }  // 菜单  template <typename DataType = int>  void Menu(int choose, SeqList<>\* list) {  int in\_1 = 0, in\_2 = 0, temp\_1 = 0;  switch (choose) {  // 插入一个数字  case 1:  cout << "请输入要插入的数字：";  cin >> in\_1;  list->Insert(list->Length() + 1, in\_1);  break;  // 指定位置插入一个数字  case 2:  cout << "请输入指定位置与要插入的数字：";  cin >> in\_1 >> in\_2;  list->Insert(in\_1, in\_2);  break;  // 获取某个数的具体位置  case 3:  cout << "请输入要查询的数：";  cin >> in\_1;  temp\_1 = list->Locate(in\_1);  if (temp\_1) {  cout << "所查数第一次出现的位置是：" << temp\_1 << endl;  }  else {  cout << "没有查到哦~" << endl;  }  break;  // 获取一个具体位置的数  case 4:  cout << "请输入要查找的位置：";  cin >> in\_1;  temp\_1 = list->Get(in\_1);  cout << "所在位置的数是：" << temp\_1 << endl;  break;  // 删除一个指定位置的数  case 5:  cout << "请输入要删除的位置：";  cin >> in\_1;  list->Delete(in\_1);  break;  case 0:  cout << "感谢体验，再见~" << endl;  break;  default:  cout << "输入有误" << endl;  }  }  **截图**      **五、实验总结：**  在本次实验中，学习了线性表的基本思想，并实现了顺序存储结构，在实验过程中，出现很多语法错误，对C++语法不太了解，到处碰壁。但是在为了实现顺序表的过程中了解了许多C++的知识，了解到C++的强大，多了解了一门语言，心里还是很开心的。  **六、教师评语：** | | | | | |