《数据结构》实验报告

**实验一 顺序表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 李爽 | 班级 | 22级软件工程专升本1班 | 学号 | 2206831522 |
| 实验名称 | 顺序表 | | | | |
| 实验时间 | 2023年4月6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：Windows7或以上版本，Visual Studio 2019  **三、实验原理：**  1、算法描述：  （1）显示函数：显示函数的主要功能是在屏幕上输出顺序表中的所有元素。实现过程比较简单，只需要使用一个循环来遍历顺序表中的所有元素，并使用 cout 语句将每个元素输出到屏幕上即可。  （2）查找函数：查找函数的主要功能是在顺序表中查找指定元素，并返回该元素在顺序表中的位置。实现过程需要遍历整个顺序表，将每个元素与待查找元素进行比较，如果找到了目标元素，就返回该元素在顺序表中的位置；如果遍历完整个顺序表都没有找到目标元素，则返回一个特殊值表示元素不在顺序表中。  （3）插入函数：插入函数的主要功能是向顺序表中插入一个新元素。实现过程需要先判断插入位置是否合法，如果合法，则需要将插入位置及其后面的元素依次向后移动一位，为新元素腾出位置。然后，将新元素插入到顺序表中指定的位置，并将顺序表长度加 1。  （4）删除函数：删除函数的主要功能是从顺序表中删除指定位置的元素。实现过程需要先判断待删除位置是否合法，如果合法，则需要将待删除位置后面的所有元素依次向前移动一位，填补被删除元素的空缺。最后，将顺序表长度减 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。  #include<iostream>  // 用于 rand() 和 srand() 函数  #include<cstdlib>  // 用于 time() 函数  #include<ctime>  using namespace std;  // 定义常量  // 顺序表的最大容量  const int MAXSIZE = 10;  // 随机生成数的最大值  const int MAXNUM = 100;  // 定义顺序表结构体  struct SeqList{  // 存储数据元素的数组  int data[MAXSIZE];  // 当前顺序表中的元素个数  int length;  };  // 初始化顺序表  void InitList(SeqList &L)  {  // 将顺序表的长度初始化为 0  L.length = 0;  }  // 随机生成 10 个 100 以内的整数并存储到顺序表中  void CreateList(SeqList &L)  {  // 用系统时间作为随机数种子，确保每次运行程序时生成的随机数不同  srand(time(NULL));  for(int i=0;i<MAXSIZE;i++){  // 生成 1 ~ 100 的随机整数并存储到顺序表中  L.data[i] = rand() % MAXNUM + 1;  }  // 顺序表的长度设为 MAXSIZE  L.length = MAXSIZE;  }  // 输出顺序表中的所有元素  void DisplayList(SeqList L)  {  cout << "当前顺序表中的元素为：";  for(int i=0;i<L.length;i++){  cout << L.data[i] << " ";  }  cout << endl;  }  // 在顺序表中查找指定元素，并返回其位置（从 0 开始计数），若找不到返回 -1  int FindElem(SeqList L, int x)  {  for(int i=0;i<L.length;i++){  if(L.data[i] == x){  return i;  }  }  return -1;  }  // 向顺序表的指定位置插入元素  void InsertElem(SeqList &L, int x, int pos)  {  // 如果顺序表已满，无法插入新元素  if(L.length == MAXSIZE){  cout << "顺序表已满，无法插入新元素！" << endl;  return;  }  // 如果插入位置不合法，无法插入新元素  if(pos < 0 || pos > L.length){  cout << "插入位置不合法，无法插入新元素！" << endl;  return;  }  // 将 pos 及其后面的元素后移  for(int i=L.length-1;i>=pos;i--){  L.data[i+1] = L.data[i];  }  // 在 pos 处插入新元素  L.data[pos] = x;  // 顺序表长度加 1  L.length++;  }  // 删除顺序表中指定位置的元素  void DeleteElem(SeqList &L, int pos)  {  // 判断删除位置是否合法  if(pos < 0 || pos >= L.length){  cout << "删除位置不合法！" << endl;  return;  }  // 将后面的元素向前移动  for(int i = pos; i < L.length - 1; i++){  L.data[i] = L.data[i + 1];  }  L.length--; // 顺序表长度减 1  }  // 主函数  int main()  {  // 声明顺序表  SeqList L;  // 插入元素的值和插入位置  int x, pos;  // 查找元素的值  int elem;  // 初始化顺序表  InitList(L);    // 随机生成 10 个 100 以内的整数并存储到顺序表中  CreateList(L);    // 输出顺序表中的所有元素  DisplayList(L);    // 查找元素  cout << "请输入要查找的元素值：";  cin >> elem;  int idx = FindElem(L, elem);  if(idx == -1){  cout << "该元素不在顺序表中！" << endl;  }else{  cout << "该元素在顺序表中的位置为：" << idx << endl;  }    // 删除元素  cout << "请输入要删除的元素位置（从 0 开始计数）：";  cin >> pos;  DeleteElem(L, pos);  DisplayList(L);    // 插入元素  cout << "请输入要插入的元素值和插入位置（从 0 开始计数）：";  cin >> x >> pos;  InsertElem(L, x, pos);  DisplayList(L);  return 0;  }    **五、实验总结：**  本次实验编写的程序主要实现了对顺序表的基本操作，包括随机生成顺序表、显示顺序表、查找顺序表、插入顺序表和删除顺序表等功能。  程序主要由五个函数组成：InitList、DisplayList、FindElem、InsertElem 和 DeleteElem。其中，InitList 函数用于初始化顺序表，DisplayList 函数用于在屏幕上显示顺序表中的所有元素，FindElem 函数用于在顺序表中查找指定元素，InsertElem 函数用于向顺序表中插入一个新元素，DeleteElem 函数用于从顺序表中删除指定位置的元素。  在实现过程中，需要注意以下几点：  需要使用结构体定义顺序表类型，其中包括数据元素和表长度等成员。  在随机生成顺序表时，需要使用 srand 和 rand 函数配合使用，以保证每次生成的随机数序列不同。  在插入和删除操作时，需要特别注意插入位置和删除位置是否合法，避免数组越界等错误。  在删除操作时，需要特别注意删除位置后面的元素是否需要向前移动一位，以保证顺序表的正确性。  **六、教师评语：** | | | | | |