计算机科学技术学院实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称** | 面向对象程序设计 | | | **学 号** | 2405111B08 |
| **实验项目** | 实验二 - 用链表实现线性表 | | | **姓 名** | 王子睿 |
| **学 时** | 2 | **项目性质** | 综合型 | **班 级** | 2405111B |
| **指导教师** | 蒋振刚 | **实验地点** | 南实训424 | **日 期** | 2025年3月31日 |
| 1. **实验目的和要求**   实验目的：练习类的构造和析构，了解他们在对象生命周期中的作用。  实验要求：  设计和实现一个线性表类DList，满足：  1）用链表实现存储；  2）实现线性表的基本操作；  3）测试代码验证 List的基本操作。 | | | | | |
| 1. **实验环境**   Visual Studio Code  MinGW-w64 | | | | | |
| 1. **实验内容与过程**   // 第2个实验：用链表实现线性表。  // 目的：练习类的构造和析构，了解他们在对象生命周期中的作用。  // 要求：设计和实现一个线性表类DList，满足：  // 1）用链表实现存储；  // 2）实现线性表的基本操作；  // 3）测试代码验证 List的基本操作。  #include <iostream>  class DList  {  private:      struct Node      {          int data;          Node \*next;      };      Node \*head;      Node \*tail;      int currentSize;  public:      DList()      {          head = nullptr;          tail = nullptr;          currentSize = 0;      }      DList(const DList &list)      {          head = nullptr;          tail = nullptr;          currentSize = 0;          Node \*current = list.head;          while (current != nullptr)          {              insert(currentSize, current->data);              current = current->next;          }      }      ~DList()      {          Node \*current = head;          while (current != nullptr)          {              Node \*temp = current;              current = current->next;              delete temp;          }      }      void insert(int index, int value)      {          // Range check          if (index < 0 || index > currentSize)          {              std::cout << "Index out of range." << std::endl;              return;          }          // Create a new node          Node \*newNode = new Node;          newNode->data = value;          // Insert at the beginning          if (index == 0)          {              newNode->next = head;              head = newNode;              if (currentSize == 0)              {                  tail = newNode;              }          }          // Insert at the end          else if (index == currentSize)          {              tail->next = newNode;              tail = newNode;              tail->next = nullptr;          }          // Insert in the middle          else          {              Node \*current = head;              for (int i = 0; i < index - 1; i++)              {                  current = current->next;              }              newNode->next = current->next;              current->next = newNode;          }          currentSize++;      }      void remove(int index)      {          // Range check          if (index < 0 || index >= currentSize)          {              std::cout << "Index out of range." << std::endl;              return;          }          // Remove the first node          if (index == 0)          {              Node \*temp = head;              head = head->next;              delete temp;              if (currentSize == 1)              {                  tail = nullptr;              }          }          // Remove the last node          else if (index == currentSize - 1)          {              Node \*current = head;              for (int i = 0; i < index - 1; i++)              {                  current = current->next;              }              Node \*temp = tail;              tail = current;              tail->next = nullptr;              delete temp;          }          // Remove in the middle          else          {              Node \*current = head;              for (int i = 0; i < index - 1; i++)              {                  current = current->next;              }              Node \*temp = current->next;              current->next = current->next->next;              delete temp;          }          currentSize--;      }      void display()      {          Node \*current = head;          while (current != nullptr)          {              std::cout << current->data << " ";              current = current->next;          }          std::cout << std::endl;      }      void reverse()      {          Node \*current = head;          Node \*prev = nullptr;          Node \*next = nullptr;          while (current != nullptr)          {              next = current->next;              current->next = prev;              prev = current;              current = next;          }          head = prev;      }      int &operator[](int index) const      {          // Range check          if (index < 0 || index >= currentSize)          {              std::cout << "Index out of range." << std::endl;          }          Node \*current = head;          for (int i = 0; i < index; i++)          {              current = current->next;          }          return current->data;      }      friend std::ostream &operator<<(std::ostream &os, const DList &list)      {          Node \*current = list.head;          while (current != nullptr)          {              os << current->data << " ";              current = current->next;          }          return os;      }      friend std::istream &operator>>(std::istream &is, DList &list)      {          int value;          while (is >> value)          {              list.insert(list.currentSize, value);          }          return is;      }      const int size() const      {          return currentSize;      }      const int find(int value) const      {          Node \*current = head;          for (int i = 0; i < currentSize; i++)          {              if (current->data == value)              {                  return i;              }              current = current->next;          }          return -1;      }  };  int main()  {      DList list;      std::cin >> list;      std::cout << list << std::endl;      list.reverse();      std::cout << list << std::endl;      std::cout << list.size() << std::endl;      std::cout << list.find(5) << std::endl;      return 0;  }  2.drawio | | | | | |
| 1. **实验结果与分析**   001  主函数创建了一个容量为10的动态线性表，然后依次测试了从cin中输入数据、输出数据到cout、翻转、移除指定位置的元素、在指定位置插入某元素、下标索引、元素数量查询、指定元素下标查询功能。 | | | | | |
| 1. **实验心得**   通过实现动态列表，我进一步理解了链表这种数据结构的工作原理及其在实际应用中的灵活性。特别是链表的插入、删除等操作，让我对数据结构有了更深入的认识。在编写find方法时，我学习了如何通过遍历链表来查找特定值的位置，这不仅锻炼了我的逻辑思维能力，也提高了我在算法设计方面的能力。 | | | | | |
| 1. **教师评语** | | | | | |
| 1. **实验成绩**   教师签名： 蒋振刚 批阅日期： 2025 年 4 月 5 日 | | | | | |

注：项目性质为 演示型、验证型、设计型、综合型和创新型。