山东大学 软件 学院

高级程序设计语言 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级： 2020级5班 |
| 实验题目：链表 | | | |
| 实验学时：4 | | 实验日期： 2020年12月21日 | |
| 实验目的：  1.学习使用链表。  2.掌握建立有序链的方法。 | | | |
| 硬件环境：  Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11GHz | | | |
| 软件环境：  IntelliJ IDEA Community;  Java jdk1.8.0\_261 | | | |
| 实验步骤与内容：  编写一个程序：   建立两条升序链表（带有头节点）；   将两条链合并成一条升序链；   三条链的数据要分别显示出来。  可以参考下面伪代码  MyList Merge(链表 m1, 链表 m2){  创建链表m3, m3.tail = m3.head;  While (m1 有节点 && m2有节点){  If (m1.head.data< m2.head.data)//M1中第一个节点和M2中第一个节点，比较大小；  {  m3.tail.next = m1.head;  m3.tail= m1.head;  m1.head = m1.head.next;  //将数据较小的节点，从原链表m1中取出，加入M3尾部；  }else  {  …….  //将数据较小的节点，从原链表m2中取出，加入M3尾部；  }  }  if (m1 ==null) 将m3.tail.next =m2 //将m2加入M3尾部；  else m3.tail.next = m1;  return m3  } | | | |
| 结论分析与体会：  1.一个动态数据结构用链来实现，利用对象引用作为链接对象间的链，就可以建立适用于各种情况的数据结构。  2.对象具有定义良好的接口，从而成为一种实现集合的完善机制。  3.动态数据结构的大小规模随需要增长和收缩。  4.通过保存和更新对象引用实现一个链表的管理。  5.通过仔细操作对象引用，可以实现插入和删除操作。  6.动态链表有许多不同的实现。  主要代码：  public class Experiment8 {  public static void main(String[] args) {  Experiment8 experiment8 = new Experiment8();  experiment8.addFromEnd(3);  experiment8.addFromEnd(2);  experiment8.addFromEnd(1);  experiment8.addFromEnd(9);  experiment8.addFromEnd(7);  experiment8.addFromEnd1(4);  experiment8.addFromEnd1(0);  experiment8.addFromEnd1(8);  experiment8.addFromEnd1(5);  experiment8.addFromEnd1(6);   experiment8.getNum(experiment8.head1);  System.*out*.println();  experiment8.getNum(experiment8.head2);  System.*out*.println();  experiment8.merge(experiment8.head1, experiment8.head2);  experiment8.getNum(experiment8.head);   }   private LinkList1 head1;  private LinkList1 head2;  private LinkList1 head;   public Experiment8() {  head1 = null;  head2 = null;  head = null;  }   public void merge(LinkList1 list1, LinkList1 list2) {  while (list1 != null){  addFromEnd0(list1.data);  list1 = list1.link;  }  while (list2 != null){  addFromEnd0(list2.data);  list2 = list2.link;  }  }   public void addFromEnd0(int addNum) {  LinkList1 position = head;  if (position == null) {  head = new LinkList1(null, addNum);  } else {  while (position.link != null) {  position = position.link;  }  position.link = new LinkList1(null, addNum);  }  }  public void addFromEnd(int addNum) {  LinkList1 position = head1;  if (position == null) {  head1 = new LinkList1(null, addNum);  } else {  while (position.link != null) {  position = position.link;  }  position.link = new LinkList1(null, addNum);  }  }   public void addFromEnd1(int addNum) {  LinkList1 position = head2;  if (position == null) {  head2 = new LinkList1(null, addNum);  } else {  while (position.link != null) {  position = position.link;  }  position.link = new LinkList1(null, addNum);  }  }   public void getNum(LinkList1 list) {  sorting(list);  while (list != null) {  System.*out*.print(list.data + "\t");  list = list.link;  }  }   public void sorting(LinkList1 list) {  int temp;  LinkList1 position = list;  while (position.link != null) {  LinkList1 min = position;  LinkList1 current = position.link;  while (current != null) {  if (current.data < min.data) min = current;  current = current.link;  }  temp = min.data;  min.data = position.data;  position.data = temp;  position = position.link;  }  }   class LinkList1 {  public LinkList1 link;  public int data;   public LinkList1() {  link = null;  data = 0;  }   public LinkList1(LinkList1 myLink, int myData) {  data = myData;  link = myLink;  }  } } | | | |