***西安财经大学****信息学院*

***姓名***王婧

***学号***1831050171

***班级*** 软件工程

***年级***1802

***指导教师*** 杨新安

*《数据结构》 课程设计报告*

***课程设计报告名称***有序单链表的的指定元素的保序插入 *实验室* 实验楼316

*完成****日期*** *2019-10-26*

|  |
| --- |
| 1. 目的   掌握链表的特点及存储实现；   1. 设计要求（15分） 2. **问题描述**   设一单链表中元素值递增有序,写一算法,将元素X插到表中适当位置,并保持单链表的有序性。   1. **需求分析** 2. 初始化链表 3. 插入元素并使之有序 4. 查看元素 5. 概要设计（20分） 6. 主界面设计（5分）   菜单   1. 存储结构设计   **单链表：**  内存   1. 系统功能设计 2. 初始化链表   1   1. 插入元素使之有序   2   1. 打印链表   3   1. 菜单   4   1. 模块设计（15分） 2. 模块设计   1）初始化链表  1  2）插入元素使之有序  2  3）打印链表  3  4）菜单  4   1. 函数主要调用关系图   0   1. 详细设计（15分） 2. 首先给一个结构体：里面有节点的值和指向下一个节点的指针   2）初始化链表：让头结点的值为0，指向NULL，然后依次让用户输入有序的数值，数值以尾插的方式插入链表  3）插入元素并使其保持有序：定义两个指针，这里考虑到单链表无法找到前一个指针，所以我的解决方法是同时设两个指针，再加上一个计数器来进行计数操作。让待插入的元素和节点依次比较，如果待插入节点大于当前节点，在与下一个节点比较，用i记数，如果找到，退出循环，让另一个指针指向待插入的位置。  4）打印链表：依次遍历链表，打印个节点的值  5）菜单：让用户可以选择自己要进行的操作  六、测试分析（15分）   1. 菜单   菜单   1. 初始化链表   初始化   1. 插入元素之前打印链表   打印列表，插入前   1. 插入元素   插入元素   1. 插入元素之后的链表   插入之后   1. 退出   退出  七、设计总结（10分）  以前的学习过程中一般使用的是利用动态顺序表来进行这个操作，具体思路是通过调整指针来完成，这次我使用单链表来完成同样的操作，实际上除了在找结点的前一个节点需要自己想出解决办法以外，其他的思路比调整指针要简单许多。  链表弥补了许多数组的缺点：数组在添加新元素时，如果添加的位置是数组的首部或者数组的中间，那么我们必须设计算法移动相关元素，并且涉及到很多元素；但是链表在搜索或随机读写内部元素时远不如数组操作起来方便。  八、附程序清单（10分）  #include <iostream>  #include <malloc.h>  using namespace std;  typedef struct Lnode  {  int data;  struct Lnode \*next;  }Node;  //链表初始化  int Create\_Init\_List(Node\* &L)  {  L = (Node\*)malloc(sizeof(Node));  if (!L) return -1;  else  {  L->data = 0;  L->next = NULL;  }  int m;  cout<<"请输入第一个数字(以0表示结束)："<<endl;  cin >> m;  Node\* fp;  fp = L;  while (m != 0)  {  //尾插法如下：  Node\* p = (Node\*)malloc(sizeof(Node));  p->data = m;  fp->next = p;  fp = p;  p->next = NULL;  cout << "请输入一个数字：" << endl;  cin>>m;  }  return 1;  }  //插入元素  void Insert\_Static\_List(Node\* L, int number)  {  Node\* p = (Node\*)malloc(sizeof(Node));    Node\* q = L->next;  Node\* fp = L->next;  int i = 0;  while (number > q->data)  {  q = q->next;  i = i + 1;  }  if (number <= q->data)  {  p->data = number;  while (i > 1)  {  fp = fp->next;  i--;  }  fp->next = p;  p->next = q;  }  }  //打印链表  void Travel\_List(Node\* L)  {  Node\* p;  p = L->next;  while (p != NULL)  {  cout<<p->data<<" ";  p = p->next;  }  }  /\*  发现是要求写一个系统  int main()  {  Node\* L;  int data;  Create\_Init\_List(L);  cout << "插入之前的序列：" << endl;  Travel\_List(L);  cout<<"请输入要插入的数："<<endl;  cin >> data;  Insert\_Static\_List(L, data);  cout << "插入之后的序列:" << endl;  Travel\_List(L);  system("pause");  return 0;  }\*/  void menu()  {  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("\*\t欢迎使用有序单链表保序插入系统\t\*\n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("\*\t\t请选择功能列表\t\t\t\*\n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("\*\t\t1.插入元素\t\t\t\*\n");  printf("\*\t\t2.查看链表元素\t\t\t\*\n");  printf("\*\t\t0.退出系统\t\t\t\*\n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  }  int main()  {  menu();  Node\* L;  int data;  cout << "输入数据" << endl;  Create\_Init\_List(L);  while (1)  {  menu();  cout << "要进行的操作" << endl;  int n;  cin >> n;  switch (n)  {  case 1:  cout << "请输入要插入的数：" << endl;  cin >> data;  Insert\_Static\_List(L, data);  break;  case 2:  Travel\_List(L);  break;  case 0:  exit(0);  default:  break;  }  cin >> n;  }  return 0;  } |