数据结构实验报告-示例

学号：XXXXXX 姓名：XXXX 专业：XXXXXXX

知识范畴：线性表 完成日期：XXXX年09月20日

实验题目：单向链表结点排序与结点逆序连接的实现

|  |  |  |  |
| --- | --- | --- | --- |
| 课程目标1（60%） | 课程目标2（40%） | 得分(100分)  期末成绩占比5% | 批阅人签字 |
| 程序代码及测试 | 写作、排版、代码注释等 |  |
|  |  |  |

实验内容及要求：

从键盘输入数据，建立带附加头结点的单向链表；按结点连接次序输出数据结点的数据；实现数据结点按结点数据值由小到大的次序连接(结点排序)；按结点连接次序输出排序后的数据结点的数据；实现数据结点逆序连接；按结点连接次序输出逆序连接后的数据结点的数据。

从键盘实现数据输入与输出的方法自拟。

实验目的：复习已学过的C/C++语言的单向链表知识；掌握单向链表结点排序与结点逆序连接算法；熟悉实验上机环境以及C/C++语言系统。

数据结构设计简要描述：

采用带附加头结点单向链表；每个结点包括整型类型的数据域和一个指针域。

typedef struct node { int data; struct node \*next; }NodeTp, \*NodePtr;

算法设计简要描述：

结点排序采用升序链表插入结点算法实现；结点逆序连接采用“先入后出”法重构链表。

输入/输出设计简要描述：

从键盘输入以空格(或CR或TAB)分隔的若干不等于0的整数，直到输入0时停止输入，按整数输入次序建立结点并顺序连接结点。

输出各结点的整数值时，每个整数采用8列字符域宽。

输入与输出有文字提示。

编程语言说明：

使用Visual C++编程。 主要代码采用C语言实现 ；动态存储分配采用C++的new和delete操作符实现；输入与输出采用C++的cin和cout流；程序注释采用C/C++规范。

主要函数说明：

NodePtr crt(); //输入若干非0整数，用先入先出法建立带附加头结点单向链表

//返回附加头结点指针

void prt(NodePtr h); //按结点连接次序输出各数据结点的值

//自变量h传入单向链表附加头结点指针

void sort(NodePtr h); //结点排序算法

void reverse(NodePtr h);//结点逆序算法

程序测试简要报告（**此处请贴运行效果图**）：

1. **测试实例1**

***程序输入***

3 -5 12 -4 0

***程序输出***

原连接次序： 3 -5 12 -4

排序后次序： -5 -4 3 12

逆序后次序： 12 3 -4 -5

***结论***

程序输出结果与期望输出结果相符。

1. **测试实例2**

***程序输入***

20 10 0

***程序输出***

原连接次序： 20 10

排序后次序： 10 20

逆序后次序： 20 10

***结论***

程序输出结果与期望输出结果相符。

1. **测试实例3**

***程序输入***

100 0

***程序输出***

原连接次序： 100

排序后次序： 100

逆序后次序： 100

***结论***

程序输出结果与期望输出结果相符。

源程序代码：

//数据结构实验1：单向链表结点排序与结点逆序连接的实现

#include <iostream>

#include <iomanip>

using namespace std;

//定义结点及结点指针数据类型

typedef struct node { int data; struct node \*next; }NodeTp, \*NodePtr;

NodePtr crt() //先入先出建立链表

{

int x;NodePtr h,last,p;

h = new NodeTp; h->next = NULL; //建附加头结点

last = h; //last指向链表当前最后一个结点

cin >> x;

while (x)

{

p = new NodeTp;p->data = x;p->next = NULL; //建新结点

last->next = p;last = p; //新结点插入到链表尾

last = p;

cin >> x;

}

return h; //返回附加头结点指针

}

void prt(NodePtr h) //按结点连接次序输出各数据结点值

{

NodePtr p = h->next; //p指向第1个数据结构

while (p) { cout << setw(8) << p->data;p = p->next; }

}

void sort(NodePtr h) //排序算法

{

NodePtr pr, p, q, s;

q = h->next; //q指向第1数据结点

h->next = NULL;//断开附加头结点，成为空链表（即升序链表初态）

while (q) //依次摘下q所指结点，插入升序链表

{

s = q;q = q->next;

pr = h;p = h->next; //开始寻找插入位置

while (p&&p->data < s->data) { pr = p;p = p->next; }

pr->next = s;s->next = p;//插入s所指结点到升序链表中

}

}

void reverse(NodePtr h) //逆序算法

{

NodePtr q, s;

q = h->next; //q指向第1数据结点

h->next = NULL; //断开附加头结点，成为空链表

while (q) //依次摘下q所指结点，“先入后出”法重构链表

{

s = q;q = q->next;

s->next = h->next;h->next = s; //s所指结点插入到链表头

}

}

int main()

{

NodePtr h; //定义单向链表附加头结点指针

cout << "输入若干非0整数，直到输入0时停止，建立单向链表"<<endl;

h = crt();

cout << "原连接次序：";prt(h);cout << endl;

sort(h); //结点排序

cout << "排序后次序：";prt(h);cout << endl;

reverse(h); //逆序连接

cout << "逆序后次序：";prt(h);cout << endl;

return 0;

}