**实验1 单链表**

**专业班级 数学171 学号 201711010427 姓名 杨力**

**实验项目中文名称**：单链表

**实验项目英文名称**：Single linked list

**实验项目编码**：1

**实验学时**：5

**适用专业**：信息与计算科学、数学与应用数学

**所属课程**：数据结构

**开课学院**：理学院

**开课学期**：第5学期

**教材及实验指导书**：

耿国华主编，《数据结构—（第三版）》，电子工业出版社，2011年

1. **实验目的或任务**

通过指导学生上机实践，对常用数据结构的基本概念及其不同的实现方法的理论得到进一步的掌握，并对在不同存储结构上实现不同的运算方式和技巧有所体会。

1. **实验教学基本要求**

1.了解实验目的及实验原理；

2.编写程序，并附上程序代码和结果图；

3.总结在编程过程中遇到的问题、解决办法和收获。

1. **实验教学的内容或要求**

1.编写函数,实现随机产生或键盘输入一组元素,建立一个带头结点的单链表（无序）

2.编写函数,实现遍历单链表

3.编写函数,实现把单向链表中元素逆置

4.编写函数,建立一个非递减有序单链表

5.编写函数,利用以上算法，建立两个非递减有序单链表,然后合并成一个非递减链表。

6.编写函数,在非递减有序单链表中插入一个元素使链表仍然有序

7.编写函数,实现在非递减有序链表中删除值为x的结点

8.编写一个主函数,在主函数中设计一个简单的菜单,分别调试上述算法

1. **实验类型或性质**

验证性

1. **实验开出要求**

必做

1. **实验所需仪器设备**

1.计算机

2.相关软件（如C,C++,PASCAL,VC,DELPHI等等）

1. **实验所用材料**

计算机耗材

1. **实验结果**
2. 结构体定义及程序所用函数与功能

typedef int SLDataType;

typedef struct SListNode {

SLDataType data;

struct SListNode \*next;

} SListNode;

typedef struct SList {

struct SListNode \*first;

} SList;

// 菜单函数

void menu()

// 初始化单链表

void SListInit(SList \*list)

// 申请单链表节点空间

SListNode \* BuySListNode(SLDataType data)

// 单链表头插

void SListPushFront(SList \*list, SLDataType data)

// 单链表头删

void SListPopFront(SList \*list)

// 遍历打印单链表

void SListPrint(SList \*list)

// 删除值为data的节点

void SListRemove(SList \*list)

// 逆置单链表

void ReverseList(SList\* list)

// 成为非递减有序单链表

void SListSort(SList \*list)

// 在非递减有序单链表中插入一个元素使链表仍然有序

void SListInsertAfter(SList \*list)

// 建立两个非递减有序链表并合并为一个非递减有序单链表

void SListMerge(SList\* Alist, SList\* Blist)

2. 编写函数,实现随机产生或键盘输入一组元素,建立一个带头结点的单链表（无序）

2.1 代码展示：

// 申请单链表节点空间

SListNode \* BuySListNode(SLDataType data) {

SListNode \*node = (SListNode \*)malloc(sizeof(SListNode));

assert(node != NULL);

node->data = data;

node->next = NULL;

return node;

}

// 单链表头插

void SListPushFront(SList \*list, SLDataType data) {

assert(list != NULL);

SListNode \*node = BuySListNode(data);

node->next = list->first;

list->first = node;

}

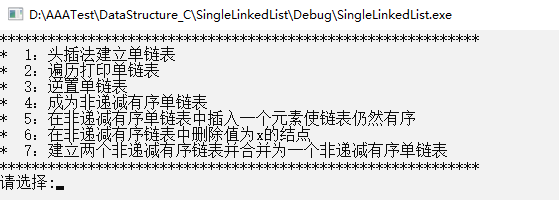
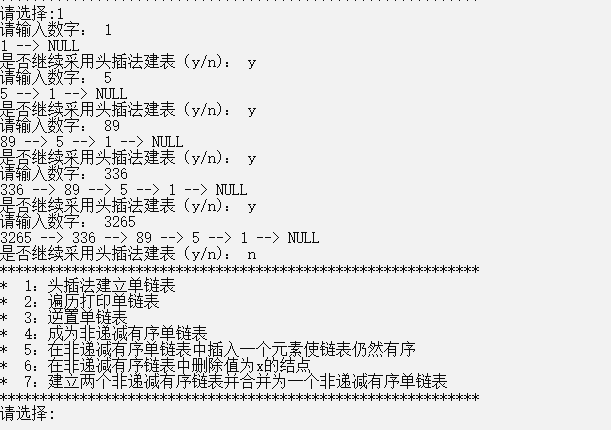
2.2 运行效果图示

图1 头插法建表运行效果图

图2 头插法建表运行效果图

2.3 函数说明

建立单项、不循环、带头节点单链表，将链表新节点申请空间封装为函数便于多次调用。因为有头节点，所以采用时间复杂度O(1)的头插及头删，便于提高程序运行效率。在调用方面采用switch-case语句及do-while循环语句进行用户的自主选择、自主建立单链表。为了后续各项操作，在此将单链表数据域所存放数据类型仅限为int型。

结论：实验结果与预期结果一致。

3. 编写函数,实现遍历单链表

3.1 代码展示：

// 遍历打印单链表

void SListPrint(SList \*list) {

for (SListNode \*cur = list->first; cur != NULL; cur = cur->next) {

printf("%d --> ", cur->data);

}

printf("NULL\n");

}

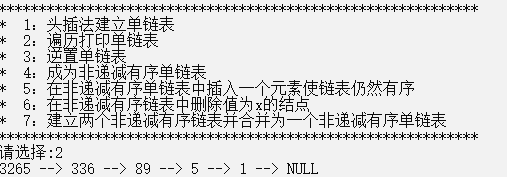
 3.2 运行效果图示

图3 遍历打印单链表运行效果图

3.3 函数说明

遍历打印打链表是很常规且常用的操作，单链表的遍历可以仿照数组遍历输入、输出及单链表的定义即可完成。主要针对for循环的跳出条件与NULL进行相等判断即可。

结论：实验结果与预期结果一致。

4. 编写函数,实现把单向链表中元素逆置

4.1 代码展示

// 逆置单链表

void ReverseList(SList\* list) {

SListNode \*newHead = NULL;

SListNode \*node;

while (list-> first!= NULL) {

node = list->first;

list->first = list->first->next;

node->next = newHead;

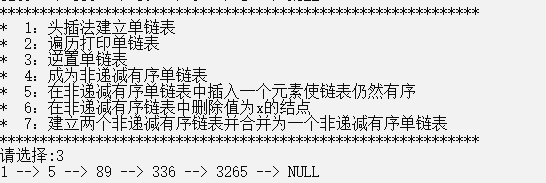
newHead = node;

}

list->first = newHead;

}

4.2 运行效果图示

图4 逆置单链表效果图

4.3函数说明

逆置单链表方式有很多种，简单一点的有头插、头删法及原地逆置法，较为复杂一点的如：递归逆置法、三指针循环遍历逆置法等高阶方法。在此采用较为简单的头插、头删法进行试验及简单说明。

针对函数名应当见名知意ReverseList，其不需要返回值，故函数类型为void，传入变量链表头结点list。在此建立两个辅助链表用于存放及交换链表节点。将该A链表首节点赋给一个空B链表，再将A链表向后走一个节点，再将B链表所接受节点的next指向NULL，再将B链表回退至第一个节点，在此需要辅助每次临时记录B链表的表头。回退的目的是为了接收A链表删除节点的next值。进行while循环，当A链表走至NULL时，链表逆置完成。

结论：实验结果与预期结果一致。

5. 编写函数,建立一个非递减有序单链表

5.1代码展示

// 成为非递减有序单链表

void SListSort(SList \*list) {

SListNode \*cur, \*prev;

int tmp;

for (cur = list->first; cur != NULL; cur = cur->next) {

for (prev = cur->next; prev != NULL; prev = prev->next) {

if (prev->data > cur->data) { // 非递减有序

tmp = cur->data;

cur->data = prev->data;

prev->data = tmp;

}

}

}

}

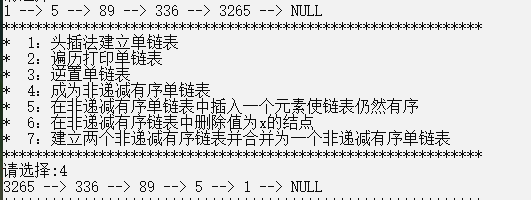
5.2运行效果图示

图5 非递减有序单链表运行效果图

5.3 函数说明

在此，我对于这个实验要求的理解为：将原有无序单链表经排序后成为非递减有序单链表。所以采用一般的冒泡排序法进行完成，比较基础，不多赘述。

6. 编写函数,在非递减有序单链表中插入一个元素使链表仍然有序

6.1 代码展示

// 在非递减有序单链表中插入一个元素使链表仍然有序

void SListInsertAfter(SList \*list) {

SListNode \*prev;

SLDataType data;

printf("请输入数字： ");

cin >> data;

SListPushFront(list, data);

SListSort(list);

}

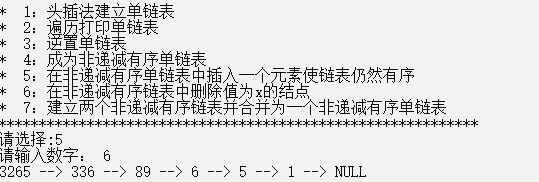
6.2 运行效果图示

图6 插入元素保持有序运行效果图

6.3 函数说明

直接判断该数的大小需要重新写遍历代码，比较麻烦。在此进行代码复用，直接进行一次头插法再进行一次排序即可达到在非递减有序单链表中插入一个元素使链表仍然有序的实验要求。

结论：实验结果与预期结果一致。

7. 编写函数,利用以上算法，建立两个非递减有序单链表,然后合并成一个非递减链表。

7.1 代码展示

// 建立两个非递减有序链表并合并为一个非递减有序单链表

void SListMerge(SList\* Alist, SList\* Blist) {

int flag = 1;

SListNode \*cur = Blist->first;

printf("请采用头插法建立单链表A：\n");

do {

int num;

char p;

printf("请输入数字： ");

scanf("%d", &num);

SListPushFront(Alist, num);

SListPrint(Alist);

printf("是否继续采用头插法建表（y/n)： ");

cin >> p;

if (p != 'y') flag = 0;

} while (flag);

printf("将单链表A整理为非递减有序排列： ");

SListSort(Alist);

SListPrint(Alist);

printf("请采用头插法建立单链表B：\n");

flag = 1; // 切记重置flag...

do {

int num;

char c;

printf("请输入数字： ");

scanf("%d", &num);

SListPushFront(Blist, num);

SListPrint(Blist);

printf("是否继续采用头插法建表（y/n)： ");

cin >> c;

if (c != 'y') flag = 0;

} while (flag);

printf("将单链表B整理为非递减有序排列： ");

SListSort(Blist);

SListPrint(Blist);

printf("将单链表A、单链表B合并为非递减有序单链表：");

SListNode \*node = Alist->first;

while (Alist->first != NULL) {

// 对之前的链表做头删

node = Alist->first;

Alist->first = Alist->first->next;

// 对新链表做头插

node->next = Blist->first;

Blist->first = node;

}

Blist->first = node;

SListSort(Blist);

SListPrint(Blist);

}

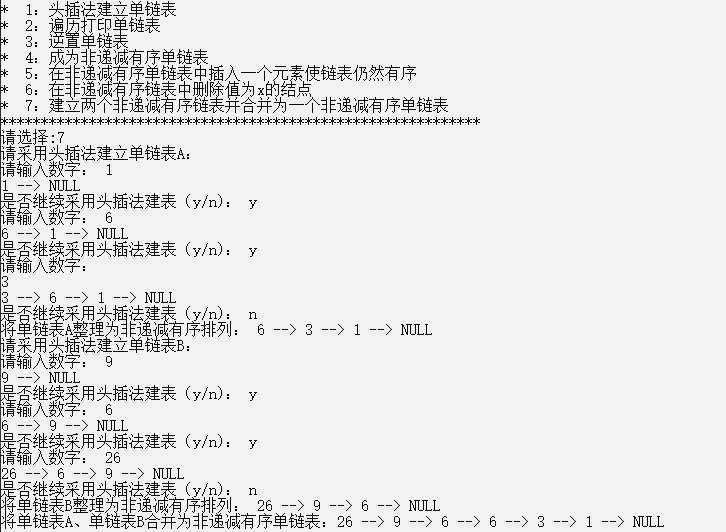
7.2 运行效果图示

图7 合并为一个非递减有序单链表运行效果图

7.3 函数说明

这个函数虽然看起来很复杂，代码量很大，但是大多数还是进行代码复用。主要思路为：在一个函数中用头插法建立两个单链表，然后调用函数进行非递减有序排列，再以逆置单链表的思想，采用while循环将Alist链表头删、头插到Blist链表上，再用函数进行非递减有序排列，最终得到Blist作为合并且非递减有序的链表。

8. 编写函数,实现在非递减有序链表中删除值为x的结点

8.1 代码展示

// 删除值为data的节点

void SListRemove(SList \*list) {

printf("请输入要删除的数据元素： ");

SLDataType data;

cin >> data;

SListNode \*prev = NULL;

SListNode \*cur = list->first;

while (cur != NULL && cur->data != data) {

prev = cur;

cur = cur->next;

}

if (cur == NULL) {

return;

}

if (prev == NULL) {

SListPopFront(list);

return;

}

prev->next = cur->next;

free(cur);

}

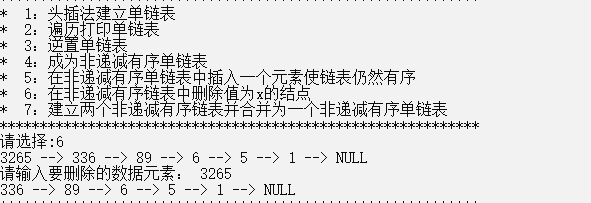
8.2 运行效果图示

图8 删除值为x的节点运行效果图

8.3函数说明

删除操作较为简单，采用while循环遍历值为data的节点进行next值跳过就可以达到删除节点的目的。但是，需要注意的是空表及链表中仅有一个节点时的特殊情况，可能会触发对NULL解引用的错误操作，导致程序崩溃。

1. **实验小结**

通过这次针对单链表的实验，主要进行了单链表的增删查改及排序等操作。实验总共花费了近6个小时左右。虽然之前在课外班都已经学习过了，也对这些操作都比较熟悉了。但是也是近3个月没有复习回顾了，导致手很生，在一些简单操作上花费了大量的时间去debug，十分不应该。这也算是给我敲了一个警钟，即便是提前了解、学习过，但是也要按时去回顾复习，最好是能够学以致用去解决一些笔试题，尽可能的来提高自己的编程能力，在大学期间没有过多琐事和烦恼，应当在这个黄金时期为自己的往后发展打下坚实的基础。

源代码：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <assert.h>

#include <stdio.h>

#include <iostream>

using namespace std;

typedef int SLDataType;

typedef struct SListNode {

SLDataType data;

struct SListNode \*next;

} SListNode;

typedef struct SList {

struct SListNode \*first;

} SList;

// 初始化单链表

void SListInit(SList \*list) {

assert(list != NULL);

list->first = NULL;

}

// 申请单链表节点空间

SListNode \* BuySListNode(SLDataType data) {

SListNode \*node = (SListNode \*)malloc(sizeof(SListNode));

assert(node != NULL);

node->data = data;

node->next = NULL;

return node;

}

// 单链表头插

void SListPushFront(SList \*list, SLDataType data) {

assert(list != NULL);

SListNode \*node = BuySListNode(data);

node->next = list->first;

list->first = node;

}

// 单链表头删

void SListPopFront(SList \*list) {

assert(list); // 没有链表

assert(list->first != NULL); // 有链表，但是链表是空的

SListNode \*old\_first = list->first;

list->first = list->first->next;

free(old\_first);

}

// 删除值为data的节点

void SListRemove(SList \*list) {

printf("请输入要删除的数据元素： ");

SLDataType data;

cin >> data;

SListNode \*prev = NULL;

SListNode \*cur = list->first;

while (cur != NULL && cur->data != data) {

prev = cur;

cur = cur->next;

}

if (cur == NULL) {

return;

}

if (prev == NULL) {

SListPopFront(list);

return;

}

prev->next = cur->next;

free(cur);

}

// 遍历打印单链表

void SListPrint(SList \*list) {

for (SListNode \*cur = list->first; cur != NULL; cur = cur->next) {

printf("%d --> ", cur->data);

}

printf("NULL\n");

}

// 成为非递减有序单链表

void SListSort(SList \*list) {

SListNode \*cur, \*prev;

int tmp;

for (cur = list->first; cur != NULL; cur = cur->next) {

for (prev = cur->next; prev != NULL; prev = prev->next) {

if (prev->data > cur->data) { // 非递减有序

tmp = cur->data;

cur->data = prev->data;

prev->data = tmp;

}

}

}

}

// 在非递减有序单链表中插入一个元素使链表仍然有序

void SListInsertAfter(SList \*list) {

SListNode \*prev;

SLDataType data;

printf("请输入数字： ");

cin >> data;

SListPushFront(list, data);

SListSort(list);

}

// 逆置单链表

void ReverseList(SList\* list) {

SListNode \*newHead = NULL;

SListNode \*node;

while (list-> first!= NULL) {

node = list->first;

list->first = list->first->next;

node->next = newHead;

newHead = node;

}

list->first = newHead;

}

// 建立两个非递减有序链表并合并为一个非递减有序单链表

void SListMerge(SList\* Alist, SList\* Blist) {

int flag = 1;

SListNode \*cur = Blist->first;

printf("请采用头插法建立单链表A：\n");

do {

int num;

char p;

printf("请输入数字： ");

scanf("%d", &num);

SListPushFront(Alist, num);

SListPrint(Alist);

printf("是否继续采用头插法建表（y/n)： ");

cin >> p;

if (p != 'y') flag = 0;

} while (flag);

printf("将单链表A整理为非递减有序排列： ");

SListSort(Alist);

SListPrint(Alist);

printf("请采用头插法建立单链表B：\n");

flag = 1; // 切记重置flag...

do {

int num;

char c;

printf("请输入数字： ");

scanf("%d", &num);

SListPushFront(Blist, num);

SListPrint(Blist);

printf("是否继续采用头插法建表（y/n)： ");

cin >> c;

if (c != 'y') flag = 0;

} while (flag);

printf("将单链表B整理为非递减有序排列： ");

SListSort(Blist);

SListPrint(Blist);

printf("将单链表A、单链表B合并为非递减有序单链表：");

SListNode \*node = Alist->first;

while (Alist->first != NULL) {

// 对之前的链表做头删

node = Alist->first;

Alist->first = Alist->first->next;

// 对新链表做头插

node->next = Blist->first;

Blist->first = node;

}

Blist->first = node;

SListSort(Blist);

SListPrint(Blist);

}

// 菜单函数

void menu() {

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* 1：头插法建立单链表 \n");

printf("\* 2：遍历打印单链表\n");

printf("\* 3：逆置单链表\n");

printf("\* 4：成为非递减有序单链表\n");

printf("\* 5：在非递减有序单链表中插入一个元素使链表仍然有序\n");

printf("\* 6：在非递减有序链表中删除值为x的结点\n");

printf("\* 7：建立两个非递减有序链表并合并为一个非递减有序单链表\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("请选择:");

}

int main() {

SList list;

SList Alist;

SList Blist;

SListInit(&list);

SListInit(&Alist);

SListInit(&Blist);

int n;

char c = 'y';

int flag = 1;

menu();

do {

scanf("%d", &n);

switch (n) {

case 1:

// 头插法建立单链表

do {

int num;

char p;

printf("请输入数字： ");

scanf("%d", &num);

SListPushFront(&list, num);

SListPrint(&list);

printf("是否继续采用头插法建表（y/n)： ");

cin >> p;

if (p != 'y') flag = 0;

} while (flag);

menu();

break;

case 2:

// 遍历打印单链表

SListPrint(&list);

menu();

break;

case 3:

// 逆置单链表

ReverseList(&list);

SListPrint(&list);

menu();

break;

case 4:

// 成为非递减有序单链表

SListSort(&list);

SListPrint(&list);

menu();

break;

case 5:

// 在非递减有序单链表中插入一个元素使链表仍然有序

SListInsertAfter(&list);

SListPrint(&list);

menu();

break;

case 6:

// 在非递减有序链表中删除值为x的结点

SListPrint(&list);

SListRemove(&list);

SListPrint(&list);

menu();

break;

case 7:

SListMerge(&Alist, &Blist);

menu();

break;

}

} while (1);

system("pause");

return 0;

}