

《数据结构课程实践》

**实验报告**

**实验名称：双链表和循环链表**

**姓 名：杨天诏**

**学 号：20081831**

**专 业：通信工程**

**实验时间：10月17日**

**杭州电子科技大学**

**通信工程学院**

**（实验报告要求：填写三四五六一共四部分内容，实验报告文件命名方式：*数据结构实验报告1-姓名-学号*，完成后上传电子版到校园网的网络教学平台，无需打印。）**

1. **实验目的**

掌握双链表和循环链表的基本算法设计。

1. **实验内容**
2. 实现**循环双链表**的头插法和尾插法的算法代码。
3. 编写main()函数，用任务1中的头插法和尾插法分别建立两个循环双链表；对于每个循环双链表，实现正序和逆序两种输出算法。
4. （附加题1, 循环双链表）对于一个带头节点的循环双链表，编写两个函数，分别实现在链表尾部插入节点以及删除节点的算法。
5. （附加题2）参考实验2对于单链表的定义，给定一个带头节点的单链表L，设计算法查找倒数第m个节点上的数据。(提示：算法能否只使用一个循环体？)
6. **算法设计**
7. 对于实验内容1和2中的CreateListF(DLinkNode \*&L, ElemType a[], int n)，CreateListR(DLinkNode \*&L, ElemType a[], int n)，DispList(DLinkNode \*L)，DispListRev(DLinkNode \*L)四个函数，**给出算法代码**，简要说明算法思想；

void CreateListF(DLinkNode\*& L, ElemType a[], int n)

//头插法建双链表

{

DLinkNode\* s;

L = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建头结点

L->prior = L->next = NULL; //单链表只需将后指针域置为NULL

for (int i = 0; i < n; i++)

{

s = (DLinkNode\*)malloc(sizeof(DLinkNode));//创建新结点

s->data = a[i];

s->next = L->next; //将结点s插在原开始结点之前,头结点之后

if (L->next != NULL) //双链表需要判断L是否存在后继节点

L->next->prior = s;

L->next = s;

s->prior = L;

}

}

void CreateListR(DLinkNode\*& L, ElemType a[], int n)

//尾插法建双链表

{

DLinkNode\* s, \* r;

L = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建头结点

L->prior = L->next = NULL;

r = L; //r始终指向终端结点,开始时指向头结点

for (int i = 0; i < n; i++)

{

s = (DLinkNode\*)malloc(sizeof(DLinkNode));//创建新结点

s->data = a[i];

r->next = s; s->prior = r; //将结点s插入结点r之后

r = s;

}

r->next = NULL; //尾结点next域置为NULL

}

void DispList(DLinkNode\* L)

{

DLinkNode\* p = L->next; //p指向双链表头节点的下一节点

while (p != NULL) //p指向不为空时，循环输出该节点的数据，并指向下一个节点

{

printf("%c ", p->data);

p = p->next;

}

printf("\n");

}

void DispListRev(DLinkNode\* L)

{

DLinkNode\* p = L;

if (p->next!= NULL)//当p指向节点不为空时候，采用递归方式，当p指向尾节点后不满足条件，结束递归

{

DispListRev(p->next);

}

if (p->prior != NULL) //递归结束后判断p的前置节点是否为空，不为空则输出数据，为空则输出回车

{

printf("%c ", p->data);

}

else

{

printf("\n");

}

}

题目二main函数为：

int main()

{

int n = 8,i;

DLinkNode\* h, \* l,\* m;

ElemType a[] = { 'A','a','b','c','a','a','b','b' };

CreateListF(h, a, n);//头插法

CreateListR(l, a, n);//尾插法

printf("头插法得到的正序输出为\n");

DispList(h);

printf("头插法得到的逆序输出为\n");

DispListRev(h);

printf("尾插法得到的正序输出为\n");

DispList(l);

printf("尾插法得到的逆序输出为\n");

DispListRev(l);

return 0;

}

1. 给出附加题1、2的算法代码，并画示意图说明算法思想。

附加题1函数代码：

bool ListIn(DLinkNode\*& L, ElemType e)//插入节点

{

int i = 0,j = 0;

DLinkNode\* p = L, \* s;

while (p->next!= NULL)

{

i++;

p = p->next;

}

s = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建新结点s

s->data = e;

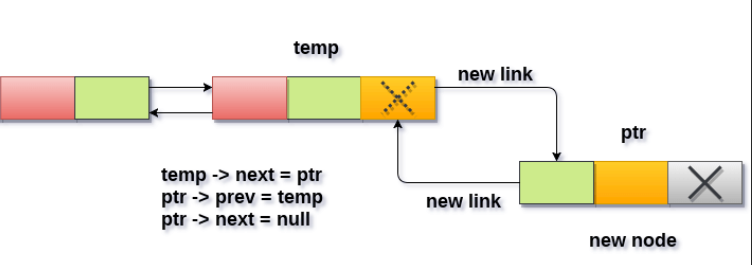
s->prior = p;

p->next = s;

s->next = NULL;

return true;

}



bool ListDe(DLinkNode\*& L)//删除节点

{

int i = 0;

DLinkNode\* p = L;

while (p->next != NULL)

{

i++;

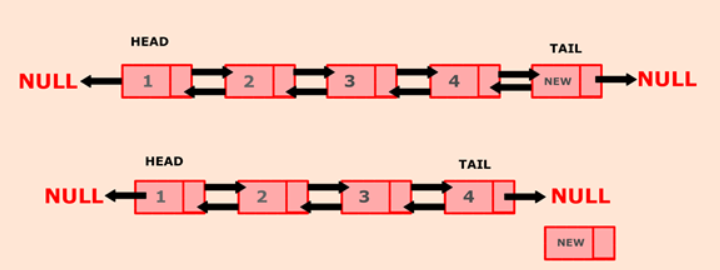
p = p->next;

}

p->prior->next = NULL;

return true;

}



如图所示，节点new是列表的结尾。将尾部的前一个节点(即节点4)作为列表的尾部。节点4的下一个将指向null

附加题一主程序代码：

int main()

{

int n = 2;

DLinkNode\* h;

ElemType a[] = { 'A','a' },b='1';

InitList(h);

CreateListF(h, a, n);//头插法

printf("原始列表为\n");

DispList(h);

ListIn(h, b);

printf("尾部插入节点后的列表为\n");

DispList(h);

ListDe(h);

printf("尾部删除节点后的列表为\n");

DispList(h);

return 0;

}

附加题二函数代码：

bool Getl(LinkNode\* L, int i)

{

int j;

LinkNode\* q=L;

if (i <= 0) return false; //i错误返回假

for (j = 0; j < i; ++j)

{

if (q)

{

q = q->next;

}

else

{

return NULL;

}

}

LinkNode\* p = L;

while (q)

{

p=p->next;

q=q->next;

}

printf("倒数第%d个节点的数据为%c", i,p->data);

}

int main()

{

LinkNode\* h;

ElemType a[10] = { '1','2','3','4','5,','6','7','8','9','10' };

CreateListR(h, a, 10);

Getl(h, 2);

}

1. **运行结果与分析（给出运行结果截图，如有必要，请加以简要说明）**

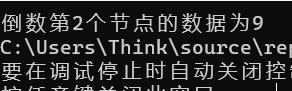
题目二运行结果为：



附加题1运行结果



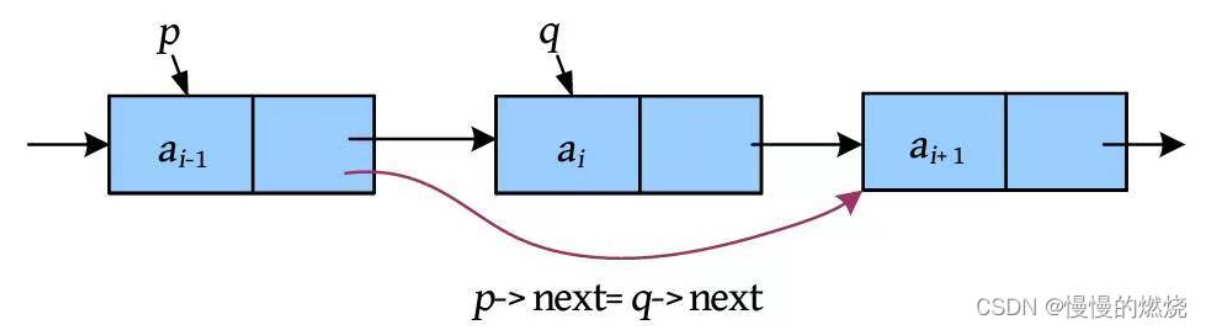
附加题二运行结果：



1. **思考题**
2. 循环双链表相对于单链表有什么特点？单链表/双链表/适用于需要双向查找节点值的情况/循环双链表各适合于哪些操作，请各举一例说明。

单链表只有后继指针，双链表有后继指针和前驱指针，循环单链表尾指针指向了头指针，循环双链表头结点的前驱指针指向了尾结点，尾结点的后继指针指向了头结点。单链表适用于节点的增加删除，双链表适用于需要双向查找节点值的情况，

2. 同步指针（前后两个指针）是如何实现单链表删除节点的操作的，请画示意图加以说明。



将指向i-1结点的指针p的指针域(next)，存储了指向i结点的q的指针域的地址，其实q->next存储了第i+1结点的地址，这样就把第i个结点删除了。

1. **实验小结（记录实验过程中遇到的主要问题和心得）**

在设计逆序输出的算法时，采用递归的方式来输出，需要考虑到递归的最后一段指向的是NULL，输出会显示错误，需要加一个判断语句才能正确输出。