实验一 顺序表、单链表、双链表及其应用

实验周次：第7周 学时：2学时 地点：学院机房

**【实验目的】**

1.领会顺序存储结构和掌握顺序表中各种基本运算算法设计。

2.领会单链表存储结构和掌握单链表中各种基本运算算法设计。

3.领会双链表存储结构和掌握双链表中的各种基本运算算法设计。

**【实验内容】**

实验题1. 编写一个程序sqlist.cpp,实现顺序表的各种基本运算和整体建表算法（假设顺序表的元素类型Elem Type为char），并在此基础上设计一个程序exp2-1.cpp完成以下功能。

1. 初始化顺序表L。
2. 依次插入a,b,c,d,e元素。
3. 输出顺序表L。
4. 输出顺序表L的长度。
5. 判断顺序表L是否为空。
6. 输出顺序表L的第3个元素。
7. 输出元素a的位置。
8. 在第4个元素位置上插入f元素。
9. 输出循序表L。
10. 删除循序表L的第3个元素。
11. 输出顺序表L。
12. 释放顺序表L。

提示内容

Sqlist.cpp

//顺序表运算算法

#include <stdio.h>

#include <malloc.h>

#define MaxSize 50

typedef char ElemType;

typedef struct

{ ElemType data[MaxSize]; //存放顺序表元素

int length; //存放顺序表的长度

} SqList; //声明顺序表的类型

void CreateList(SqList \*&L,ElemType a[],int n) //整体建立顺序表

{

L=(SqList \*)malloc(sizeof(SqList));

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

L->data[i]=a[i];

L->length=n;

}

void InitList(SqList \*&L) //初始化线性表

{

L=(SqList \*)malloc(sizeof(SqList)); //分配存放线性表的空间

L->length=0;

}

void DestroyList(SqList \*&L) //销毁线性表

{

free(L);

}

//文件名:exp2-1.cpp

#include "sqlist.cpp"

int main()

{

SqList \*L;

ElemType e;

printf("顺序表的基本运算如下:\n");

printf(" (1)初始化顺序表L\n");

InitList(L);

printf(" (2)依次插入a,b,c,d,e元素\n");

ListInsert(L,1,'a');

ListInsert(L,2,'b');

ListInsert(L,3,'c');

ListInsert(L,4,'d');

ListInsert(L,5,'e');

printf(" (3)输出顺序表L:");

DispList(L);

printf(" (4)顺序表L长度:%d\n",ListLength(L));

printf(" (5)顺序表L为%s\n",(ListEmpty(L)?"空":"非空"));

GetElem(L,3,e);

printf(" (6)顺序表L的第3个元素:%c\n",e);

printf(" (7)元素a的位置:%d\n",LocateElem(L,'a'));

printf(" (8)在第4个元素位置上插入f元素\n");

ListInsert(L,4,'f');

printf(" (9)输出顺序表L:");

DispList(L);

printf(" (10)删除L的第3个元素\n");

ListDelete(L,3,e);

printf(" (11)输出顺序表L:");

DispList(L);

printf(" (12)释放顺序表L\n");

DestroyList(L);

return 0;

}

**程序：**

//文件名:exp2-1.cpp

#include "sqlist.cpp"

int main()

{

    SqList \*L;

    ElemType e;

    printf("顺序表的基本运算如下:\n");

    printf("  (1)初始化顺序表L\n");

    InitList(L);

    printf("  (2)依次插入a,b,c,d,e元素\n");

    ListInsert(L, 1, 'a');

    ListInsert(L, 2, 'b');

    ListInsert(L, 3, 'c');

    ListInsert(L, 4, 'd');

    ListInsert(L, 5, 'e');

    printf("  (3)输出顺序表L:");

    DispList(L);

    printf("  (4)顺序表L长度:%d\n", ListLength(L));

    printf("  (5)顺序表L为%s\n", (ListEmpty(L) ? "空" : "非空"));

    GetElem(L, 3, e);

    printf("  (6)顺序表L的第3个元素:%c\n", e);

    printf("  (7)元素a的位置:%d\n", LocateElem(L, 'a'));

    printf("  (8)在第4个元素位置上插入f元素\n");

    ListInsert(L, 4, 'f');

    printf("  (9)输出顺序表L:");

    DispList(L);

    printf("  (10)删除L的第3个元素\n");

    ListDelete(L, 3, e);

    printf("  (11)输出顺序表L:");

    DispList(L);

    printf("  (12)释放顺序表L\n");

    DestroyList(L);

    return 0;

}

//文件名:exp2-1.cpp

#include "sqlist.cpp"

int main()

{

    SqList \*L;

    ElemType e;

    printf("顺序表的基本运算如下:\n");

    printf("  (1)初始化顺序表L\n");

    InitList(L);

    printf("  (2)依次插入a,b,c,d,e元素\n");

    ListInsert(L, 1, 'a');

    ListInsert(L, 2, 'b');

    ListInsert(L, 3, 'c');

    ListInsert(L, 4, 'd');

    ListInsert(L, 5, 'e');

    printf("  (3)输出顺序表L:");

    DispList(L);

    printf("  (4)顺序表L长度:%d\n", ListLength(L));

    printf("  (5)顺序表L为%s\n", (ListEmpty(L) ? "空" : "非空"));

    GetElem(L, 3, e);

    printf("  (6)顺序表L的第3个元素:%c\n", e);

    printf("  (7)元素a的位置:%d\n", LocateElem(L, 'a'));

    printf("  (8)在第4个元素位置上插入f元素\n");

    ListInsert(L, 4, 'f');

    printf("  (9)输出顺序表L:");

    DispList(L);

    printf("  (10)删除L的第3个元素\n");

    ListDelete(L, 3, e);

    printf("  (11)输出顺序表L:");

    DispList(L);

    printf("  (12)释放顺序表L\n");

    DestroyList(L);

    return 0;

}

实验题2.编写一个程序linklist.cpp,实现单链表的各种 基本运算和整体建表算法（假设单链表的元素类型Elem Type为char），并在此基础上设计一个程序exp2-2.cpp完成以下功能。

1. 初始化单链表h。
2. 依次采用尾插法插入a,b,c,d,e元素。
3. 输出单链表h。
4. 输出单链表L的长度。
5. 判断单链表L是否为空。
6. 输出单链表h的第3个元素。
7. 输出元素a的位置。
8. 在第4个元素位置上插入f元素。
9. 输出单链表h。
10. 删除单链表h的第3个元素。
11. 输出单链表h。
12. 释放单链表h。

提示内容：

//单链表运算算法

#include <stdio.h>

#include <malloc.h>

typedef char ElemType;

typedef struct LNode

{

ElemType data;

struct LNode \*next; //指向后继结点

} LinkNode; //单链表结点类型

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

//头插法建立单链表

{

LinkNode \*s;

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

L->next=NULL;

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

{

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

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

L->next=s;

}

}

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

//尾插法建立单链表

{

LinkNode \*s,\*r;

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

L->next=NULL;

r=L; //r始终指向尾结点,开始时指向头结点

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

{

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

s->data=a[i];

r->next=s; //将结点s插入r结点之后

r=s;

}

r->next=NULL; //尾结点next域置为NULL

}

void InitList(LinkNode \*&L) //初始化线性表

{

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

L->next=NULL; //单链表置为空表

}

void DestroyList(LinkNode \*&L) //销毁线性表

{

LinkNode \*pre=L,\*p=pre->next;

while (p!=NULL)

{ free(pre);

pre=p; //pre、p同步后移一个结点

p=pre->next;

}

free(pre); //此时p为NULL,pre指向尾结点,释放它

}

bool ListEmpty(LinkNode \*L) //判线性表是否为空表

{

return(L->next==NULL);

}

//文件名:exp2-2.cpp

#include "linklist.cpp"

int main()

{

LinkNode \*h;

ElemType e;

printf("单链表的基本运算如下:\n");

printf(" (1)初始化单链表h\n");

InitList(h);

printf(" (2)依次采用尾插法插入a,b,c,d,e元素\n");

ListInsert(h,1,'a');

ListInsert(h,2,'b');

ListInsert(h,3,'c');

ListInsert(h,4,'d');

ListInsert(h,5,'e');

printf(" (3)输出单链表h:");

DispList(h);

printf(" (4)单链表h长度:%d\n",ListLength(h));

printf(" (5)单链表h为%s\n",(ListEmpty(h)?"空":"非空"));

GetElem(h,3,e);

printf(" (6)单链表h的第3个元素:%c\n",e);

printf(" (7)元素a的位置:%d\n",LocateElem(h,'a'));

printf(" (8)在第4个元素位置上插入f元素\n");

ListInsert(h,4,'f');

printf(" (9)输出单链表h:");

DispList(h);

printf(" (10)删除h的第3个元素\n");

ListDelete(h,3,e);

printf(" (11)输出单链表h:");

DispList(h);

printf(" (12)释放单链表h\n");

DestroyList(h);

return 0;

}

实验题3.编写一个程序dlinklist.cpp,实现双链表的各种基本运算和整体建表算法（假设双链表的元素类型Elem Type为char），并在此基础上设计一个程序exp2-3.cpp完成以下功能。

1. 初始化双链表h。
2. 依次采用尾插法插入a,b,c,d,e元素。
3. 输出双链表h。
4. 输出双链表L的长度。
5. 判断双链表L是否为空。
6. 输出双链表h的第3个元素。
7. 输出元素a的位置。
8. 在第4个元素位置上插入f元素。
9. 输出双链表h。
10. 删除双链表h的第3个元素。
11. 输出双链表h。
12. 释放双链表h。

提示内容：

/双链表运算算法

#include <stdio.h>

#include <malloc.h>

typedef int ElemType;

typedef struct DNode

{

ElemType data;

struct DNode \*prior; //指向前驱结点

struct DNode \*next; //指向后继结点

} DLinkNode; //声明双链表结点类型

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;

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) 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 InitList(DLinkNode \*&L) //初始化线性表

{

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

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

}

void DestroyList(DLinkNode \*&L) //销毁线性表

{

DLinkNode \*pre=L,\*p=pre->next;

while (p!=NULL)

{

free(pre);

pre=p; //pre、p同步后移一个结点

p=pre->next;

}

free(p);

}

bool ListEmpty(DLinkNode \*L) //判线性表是否为空表

{

return(L->next==NULL);

}

//文件名:exp2-3.cpp

#include "dlinklist.cpp"

int main()

{

DLinkNode \*h;

ElemType e;

printf("双链表的基本运算如下:\n");

printf(" (1)初始化双链表h\n");

InitList(h);

printf(" (2)依次采用尾插法插入a,b,c,d,e元素\n");

ListInsert(h,1,'a');

ListInsert(h,2,'b');

ListInsert(h,3,'c');

ListInsert(h,4,'d');

ListInsert(h,5,'e');

printf(" (3)输出双链表h:");

DispList(h);

printf(" (4)双链表h长度:%d\n",ListLength(h));

printf(" (5)双链表h为%s\n",(ListEmpty(h)?"空":"非空"));

GetElem(h,3,e);

printf(" (6)双链表h的第3个元素:%c\n",e);

printf(" (7)元素a的位置:%d\n",LocateElem(h,'a'));

printf(" (8)在第4个元素位置上插入f元素\n");

ListInsert(h,4,'f');

printf(" (9)输出双链表h:");

DispList(h);

printf(" (10)删除h的第3个元素\n");

ListDelete(h,3,e);

printf(" (11)输出双链表h:");

DispList(h);

printf(" (12)释放双链表h\n");

DestroyList(h);

return 1;

}