C++实验报告三：数据结构之链表

实验目的

> 1.掌握对链表的基础认识
>
> 2.掌握链表的动态创建过程（头插法，尾插法）
>
> 3.掌握链表的打印，查询，删除的功能
>
> 4.掌握链表的翻转，插入功能
>
> 5.掌握插入排序在链表中的应用，感受链表的独特之处

实验内容一

> **任务一：**
>
> 1．实验要求
>
> 编程实现如下功能
>
> （1）根据输入的一系列整数，以0标志结束，用***头插法***建立单链表，并输出单链表中各元素值，观察输入的内容与输出的内容是否一致。
>
> （2）根据输入的一系列整数，以0标志结束，用***尾插法***建立单链表，并输出单链表中各元素值，观察输入的内容与输出的内容是否一致。
>
> （3）在单链表的第i个元素之后插入一个值为x的元素，并输出插入后的单链表中各元素值。
>
> （4）删除单链表中第i个元素，并输出删除后的单链表中各元素值。
>
> （5）在单链表中查找第i个元素，如果查找成功，则显示该元素的值，否则显示该元素不存在。

实验方案一

```
//ny dpcc
#include<iostream>
#include<string>
#include<algorithm>
#define rep(i,a,n) for(int i=a;i<=n;i++)
#define frep(i,a,n) for(int i=a;i>=n;i--)
using namespace std;
struct Node
{
    int data;
```

```
    struct Node* next;
};
//带虚拟头节点头插法
struct Node* headinsert(struct Node* head)
{
    int cnt=1;
    cout<<"现在开始带头节点头插法创建链表"<<'\n'<<'\n';
    cout<<"0代表终止"<<'\n'<<'\n';
    struct Node* s;
    head = new struct Node;
    head->next = NULL;
    int data;
    cout<<"请输入第一个节点"<<'\n'<<'\n';
    cin >> data;
    while (data!=0)
    {
        s = new struct Node;
        s->data = data;
        s->next = head->next;
        head->next = s;
        cout<<"请输入第"<<cnt+1<<"个节点"<<'\n'<<'\n';
        cin >> data;
        cnt++;
    }
    return head;
}
//不带虚拟头节点的头插法
struct Node *headinsertx(struct Node *head)
{
    int cnt=1;
    cout<<"现在开始不带虚拟头节点的链表"<<'\n'<<'\n';
    Node *s;
    head=new struct Node;
    int data;
    cout<<"请输入第一个节点的值"<<'\n'<<'\n';
    cin>>data;
    head->data=data;
    head->next=NULL;
    cout<<"请输入第二个节点的值"<<'\n'<<'\n';
    cin>>data;
    cnt=2;
    while(data!=0)
    {
        s=new struct Node;
        s->data=data;
        s->next=head;
        head=s;
        cout<<"请输入第"<<cnt+1<<"个节点"<<'\n'<<'\n';
        cin>>data;
        cnt++;
    }
    return head;
    }
//带虚拟头节点尾插法
struct Node * tailinsert(struct Node *head)
{
```

```
    int cnt=1;
    cout<<"现在开始尾插法的创建过程"<<'\n'<<'\n';
    cout<<"输入0代表已经创建完毕"<<'\n'<<'\n';
    head=new struct Node;
    struct Node *s,*r=head;
    int data;
    cout<<"请输入第一个节点"<<'\n'<<'\n';
    cin>>data;
    while(data!=0)
    {
        s=new struct Node;
        s->data=data;
        r->next=s;
        r=s;
        cout<<"请输入第"<<cnt+1<<"个节点"<<'\n'<<'\n';
        cin>>data;
        cnt++;
    }
    r->next=NULL;
    return head;
}
//不带虚拟头节点的尾插法
struct Node*tailinsertx(struct Node *head)
{
    int cnt=1;
    cout<<"现在开始不带虚拟头节点的尾插法"<<'\n'<<'\n';
    head=new struct Node;
    struct Node *s,*r;
    int data;
    cout<<"请输入第一个节点"<<'\n'<<'\n';
    cin>>data;
    head->data=data;
    head->next=NULL;
    r=head;
    cout<<"请输入第2个节点"<<'\n'<<'\n';
    cin>>data;
    cnt=2;
    while(data!=0)
    {
        s=new struct Node;
        s->data=data;
        r->next=s;
        r=s;
        cout<<"请输入第"<<cnt+1<<"个节点"<<'\n'<<'\n';
        cin>>data;
        cnt++;
    }
    r->next=NULL;
    return head;
}
//带虚拟头节点的打印方法打印链表
void printlink(struct Node* head)
{
    int cnt=1;
    struct Node* x = head->next;
    cout << "开始打印链表" << '\n'<<'\n';
```

```
    while (x!= NULL)
    {
        cout <<"第"<<cnt<<"个节点是"<< x->data<<'\n'<<'\n';
        x = x->next;
        cnt++;
    }
}
//不带虚拟头节点的打印
void printlinkx(struct Node*head)
{
    int cnt=1;
    struct Node* x = head;
    cout << "开始打印链表" << '\n'<<'\n';
    while (x!= NULL)
    {
        cout <<"第"<<cnt<<"个节点是"<< x->data<<'\n'<<'\n';
        x = x->next;
        cnt++;
    }
}
//获取链表个数
int getlinkcnt(struct Node* head)
{
    struct Node* x;
    x = head;
    int cnt = 0;
    while (x->next != NULL)
    {
        cnt++;
        x = x->next;
    }
    return cnt;
}
//在某个元素后插入一个值为data的节点
void insert(struct Node* head, struct Node* x, int num)
{
    struct Node* point;
    point = head;
    int cnt = 1;
    while (cnt != num)
    {
        cnt++;
        point = point->next;
    }
    x->next = point->next;
    point->next = x;

}
//删除链表中的第i个元素
struct Node* deleteNode(struct Node* head, int num)
{
    if(num>getlinkcnt(head))
    {
        cout<<"无法删除"<<'\n';
        return head;
    }
```

```
    struct Node* x;
    x = head;
    int cnt = 1;
    if (num == 1)
    {
        head = head->next;
        return head;
    }
    while (cnt != num - 1)
    {
        x = x->next;
        cnt++;
    }
    x->next = x->next->next;
    return head;
}
//在单链表中查找第i个元素，
//如果查找成功，则显示，否则显示改元素不存在
void seeknumfromlink(struct Node* head, int num)
{
    if (getlinkcnt(head)< num)
    {
        cout << "不存在该元素" << '\n'<<'\n';
    }
    else
    {
        int cnt = 0;
        struct Node* x;
        x = head;
        while (cnt != num)
        {
            x = x->next;
            cnt++;
        }
        cout << "找到" << "第" << num << "个元素了" << '\n'<<'\n';
        cout << "这个元素的值是:" << x->data << '\n'<<'\n';
    }
}
//修改节点的值
void  changelink(struct Node*head, int num, int data)
{
    struct Node* x;
    x = head;
    int cnt = 1;
    while (cnt != num)
    {
        x = x->next;
        cnt++;
    }
    x->data = data;
}
int main()
{
//带头插入法创建链表
struct Node* head1;
head1=headinsertx(head1);
```

```
printlinkx(head1);
//带头节点的尾插入法
/*
struct Node *head2;
head2=headinsert(head2);
printlink(head2);
//不带头节点的头插入法
struct Node *head3;
head3=headinsertx(head3);
printlinkx(head3);
//不带尾头节点的尾插入法
struct Node *head4;
head4=tailinsertx(head4);
printlinkx(head4);

int num;
cout<<"请输入你要查询的链表元素下标"<<'\n'<<'\n';
cout<<"输入0代表你不想查询了"<<'\n';
/*while(1)
{
cin>>num;
seeknumfromlink(head1,num);
}
*/
/*
cout<<"现在开始链表的插入过程"<<'\n'<<'\n';
cout<<"如果插入位置在0那么意味着插入结束了"<<'\n'<<'\n';
/*
while(1)
{
    struct Node*x;
    x=new struct Node;
    int data;
    cout<<"请输入插入的节点"<<'\n'<<'\n';
    cin>>data;
    x->data=data;
    int num;
    cout<<"请输入你要插入的位置"<<'\n'<<'\n';
    cin>>num;
        if(num==0)
    {
        cout<<"插入工作完成"<<'\n'<<'\n';
        break;
    }
    insert(head1,x,num);
    printlinkx(head1);
}
*/
cout<<"接下来是删除工作"<<'\n'<<'\n';
while(1)
{
    cout<<"请输入你要删除的链表的下标"<<'\n'<<'\n';
    int num;
    cin>>num;
head1=deleteNode(head1,num);
    printlinkx(head1);
```

```
}
return 0;
}
```

实验结果一

创建的结果

> 现在开始尾插法的创建过程
>
> 输入0代表已经创建完毕
>
> 请输入第一个节点
>
> 1
> 请输入第2个节点
>
> 2
> 请输入第3个节点
>
> 3
> 请输入第4个节点
>
> 4
> 请输入第5个节点
>
> 5
> 请输入第6个节点
>
> 6
> 请输入第7个节点
>
> 0
> 开始打印链表
>
> 第1个节点是1
>
> 第2个节点是2
>
> 第3个节点是3
>
> 第4个节点是4
>
> 第5个节点是5
>
> 第6个节点是6
>
> 现在开始带头节点头插法创建链表
>
> 0代表终止
>
> 请输入第一个节点
>
> 1
> 请输入第2个节点
>
> 2
> 请输入第3个节点
>
> 3
> 请输入第4个节点

4
请输入第5个节点

5
请输入第6个节点

6
请输入第7个节点

0
开始打印链表

第1个节点是6

第2个节点是5

第3个节点是4

第4个节点是3

第5个节点是2

第6个节点是1

现在开始不带虚拟头节点的链表

请输入第一个节点的值

1
请输入第二个节点的值

2
请输入第3个节点

3
请输入第4个节点

4
请输入第5个节点

5
请输入第6个节点

6
请输入第7个节点

0
开始打印链表

第1个节点是6

第2个节点是5

第3个节点是4

第4个节点是3

第5个节点是2

第6个节点是1

现在开始不带虚拟头节点的尾插法

> 请输入第一个节点
>
> 1
> 请输入第2个节点
>
> 2
> 请输入第3个节点
>
> 3
> 请输入第4个节点
>
> 4
> 请输入第5个节点
>
> 5
> 请输入第6个节点
>
> 6
> 请输入第7个节点
>
> 0
> 开始打印链表
>
> 第1个节点是1
>
> 第2个节点是2
>
> 第3个节点是3
>
> 第4个节点是4
>
> 第5个节点是5
>
> 第6个节点是6

查询的结果

> 现在开始尾插法的创建过程
>
> 输入0代表已经创建完毕
>
> 请输入第一个节点
>
> 1
> 请输入第2个节点
>
> 2
> 请输入第3个节点
>
> 3
> 请输入第4个节点
>
> 4
> 请输入第5个节点
>
> 5
> 请输入第6个节点
>
> 6
> 请输入第7个节点

0
开始打印链表

第1个节点是1

第2个节点是2

第3个节点是3

第4个节点是4

第5个节点是5

第6个节点是6

请输入你要查询的链表元素下标

输入0代表你不想查询了
1
找到第1个元素了

这个元素的值是:1

2
找到第2个元素了

这个元素的值是:2

3
找到第3个元素了

这个元素的值是:3

4
找到第4个元素了

这个元素的值是:4

5
找到第5个元素了

这个元素的值是:5

6
找到第6个元素了

这个元素的值是:6

7
不存在该元素

插入的结果

现在开始不带虚拟头节点的链表

请输入第一个节点的值

1
请输入第二个节点的值

2
请输入第3个节点

3
请输入第4个节点

4
请输入第5个节点

5
请输入第6个节点

6
请输入第7个节点

0
开始打印链表

第1个节点是6

第2个节点是5

第3个节点是4

第4个节点是3

第5个节点是2

第6个节点是1

现在开始链表的插入过程

如果插入位置在0那么意味着插入结束了

请输入插入的节点

1
请输入你要插入的位置

1
开始打印链表

第1个节点是6

第2个节点是1

第3个节点是5

第4个节点是4

第5个节点是3

第6个节点是2

第7个节点是1

请输入插入的节点

2
请输入你要插入的位置

2
开始打印链表

第1个节点是6

> 第2个节点是1
>
> 第3个节点是2
>
> 第4个节点是5
>
> 第5个节点是4
>
> 第6个节点是3
>
> 第7个节点是2
>
> 第8个节点是1
>
> 请输入插入的节点
>
> 3
> 请输入你要插入的位置
>
> 0
> 插入工作完成

删除的结果

> 现在开始不带虚拟头节点的链表
>
> 请输入第一个节点的值
>
> 1
> 请输入第二个节点的值
>
> 2
> 请输入第3个节点
>
> 3
> 请输入第4个节点
>
> 4
> 请输入第5个节点
>
> 5
> 请输入第6个节点
>
> 6
> 请输入第7个节点
>
> 0
> 开始打印链表
>
> 第1个节点是6
>
> 第2个节点是5
>
> 第3个节点是4
>
> 第4个节点是3
>
> 第5个节点是2
>
> 第6个节点是1
>
> 接下来是删除工作

> 请输入你要删除的链表的下标
>
> 1
> 开始打印链表
>
> 第1个节点是5
>
> 第2个节点是4
>
> 第3个节点是3
>
> 第4个节点是2
>
> 第5个节点是1
>
> 请输入你要删除的链表的下标
>
> 1
> 开始打印链表
>
> 第1个节点是4
>
> 第2个节点是3
>
> 第3个节点是2
>
> 第4个节点是1
>
> 请输入你要删除的链表的下标
>
> 1
> 开始打印链表
>
> 第1个节点是3
>
> 第2个节点是2
>
> 第3个节点是1
>
> 请输入你要删除的链表的下标
>
> 1
> 开始打印链表
>
> 第1个节点是2
>
> 第2个节点是1

实验内容二

> **任务二：**
>
> 在实验任务一的基础上，实现插入排序

实验方案二

```
void insertSort(struct Node * L){
    // p，u指针用来处理原链表， q，r指针用来在有序链表中找到合适的插入位置
    struct Node * p,q,r,u;
    // 指向原链表，通过while(p)，迭代将每个节点插入到有序链表中
    p = L->next;
    // L->next = NULL,链表置为空，等待插入有序节点
```

```
    L->next = NULL;

    //循环处理直到原链表处理完
    while(p)
    {
        // 从有序链表的链表头开始比较，r始终在q前一位
        // 新节点就插入在r、q之间
        r = L;
        q = L->next;
        // 循环第一次的时候q值为null，没有执行此循环
        while(q && q->data <= p->data)
        {
            r = q;
            q = q->next;
        }
        // 先保留原来指针的信息
        u = p->next;

        // 完成插入的过程
        p->next = q;
        r->next = p;

        // 插入完成后，p指针指向原链表下一个，继续处理
        p = u;
    }
}
```

实验结果二

现在开始带头节点头插法创建链表

0代表终止

请输入第一个节点

1
请输入第2个节点

3
请输入第3个节点

2
请输入第4个节点

4
请输入第5个节点

5
请输入第6个节点

7
请输入第7个节点

```
0
开始打印链表

第1个节点是7

第2个节点是5

第3个节点是4

第4个节点是2

第5个节点是3

第6个节点是1

开始打印链表

第1个节点是1

第2个节点是2

第3个节点是3

第4个节点是4

第5个节点是5

第6个节点是7
```

实验心得

> 通过本次实验，我对链表的创建和指针的使用有了进一步的理解，同时，对链表的功能的大体实现都有所知晓，此外我也掌握了动态分配内存的写法，一种是New,一种是malloc，两种各有利弊，均可使用，并且，我对指针的使用时机和插入排序的本质，以及链表和数组在哪些方面孰优孰劣也有所了解。

实验日期

> 12月1日