2019/05/04 链表结构 - AcWing

> 指针

链表结构

- 1、单链表的定义
  ①类型和变量的说明

```
struct Node{
    int data;
    Node *next;
};
Node *p;
```

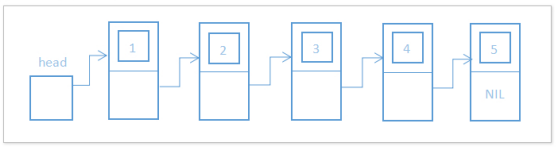
②申请存储单元

```
p=new Node;
```

③指针变量的赋值

```
指针变量名=NULL;
```

- 2、单链表的结构、建立、输出



下面是建立并输出单链表的程序。

```
#include<bits/stdc++.h>
using namespace std;

struct Node{
    int data;
    Node *next;
};
Node *head,*p,*r;

int x;

int main(){
    cin>>x;
    head=new Node;
    r=head;

    while(x!=-1){
        p=new Node;

        p->data=x;
        p->next=NULL;
        r->next=p;

        r=p;
        cin>>x;
    }

    p=head->next;

    while(p->next!=NULL){
        cout<<p->data<<" ";
        p=p->next;
    }
    cout<<p->data<<endl;

    system("pause");
}
```

- 3、单链表的操作
  ①查找 “数据域满足一定条件的结点”

```
p=head->next;
while((p->data!=x)&&(p->next!=NULL))
    p=p->next;
if(p->data==x)
    找到了就处理;
else
    输出不存在;
```

如果想找到所有满足条件的结点，则修改如下：

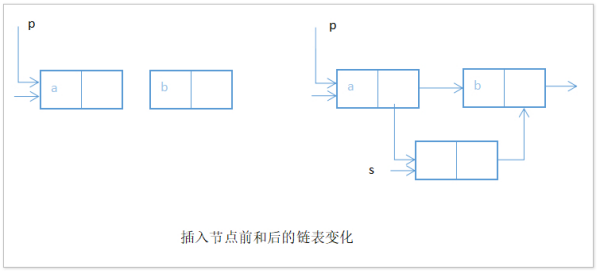
```
p=head->next;
while(p->next!=NULL){
    if(p->data==x)
    p=p->next;
}
```

②取出单链表的第 i 个结点的数据域

```
void get(Node *head,int i){
```

```
    Node *p;int j;
    p=head->next;
    j=1;

    while((p!=NULL)&&(j<i)){
        p=p->next;
        j++;
    }

    if((p!=NULL)&&(j==i))
        cout<<p->data;
    else
        cout<<"i not exsit!";
}
```

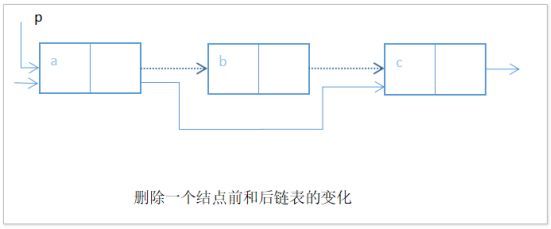
③插入一个结点在单链表中去



插入节点前和后的链表变化

```
void insert(Node *head,int i,int x){
    Node *p,*s;int j;
    p=head;j=0;

    while((p!=NULL)&&(j<i-1)){
        p=p->next;
        j++;
    }

    if(p==NULL)
        cout<<"no this position!";
    else{
        s=new Node;

        s->data=x;
        s->next=p->next;
        p->next=s;
    }
}
```

④删除单链表中的第 i 个结点



删除一个结点前和后链表的变化

```
void delete(Node *head,int i){//删除第i个元素
    Node *p,*s;int j;
    p=head;j=0;

    while((p->next!=NULL)&&(j<i-1)){
        p=p->next;
        j++;
    } //p指向第i-1个结点

    if(p->next==NULL)
        cout<<"no this position!";
    else{
        s=p->next;
        p->next=p->next->next;//或p->next=s->next
        free(s);
    }
}
```

⑤求单链表的实际长度

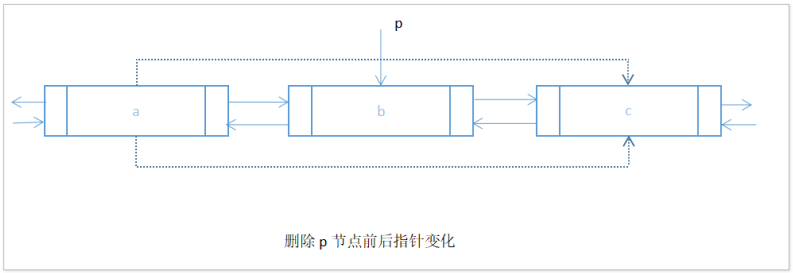
```
int len(Node *head){
    int n=0;
    p=head;

    while(p!=NULL){
        n++;
        p=p->next;
    }

    return n;
}
```

- 4、双向链表

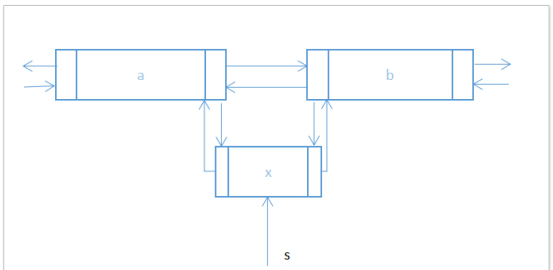
  【数据结构的定义】

```
struct node{
    int data;
    node *pre,*next;
}
node *head,*p,*q,*r;
```

下面是双向链表的插入和删除过程



删除 p 节点前后指针变化



在 p 节点之前插入 s 节点前后的指针变化

```
void insert(node *head,int i,int x){
    node *s,*p;
    int j;

    s=new node;
    s->data=x;

    p=head;j=0;

    while((p->next!=NULL)&&(j<i)){
        p=p->next;
        j++;
    }

    if(p==NULL)
        cout<<"no this position!";
    else{
        s->pre=p->pre;
        p->pre=s;
        s->next=p;
        p->pre->next=s;
    }
}
```

```
void delete(node *head,int i){
    int j;node *p;
    p=head;j=0;

    while((p->next!=NULL)&&(j<i)){
        p=p->next;
        j++;
    }

    if(p==NULL)
        cout<<"no this position!";
    else{
        p->pre->next=p->next;
        p->next->pre=p->pre;
    }
}
```

```
using namespace std;

struct node{
    int d;
    node *next;
};

int n,m;
node *head,*p,*r;

int main(){
    int i,j,k,l;
    cin>>n>>m;
    head=new node;
    hesd->d=1;head->next=NULL;r=head;

    for(i=2;i<=n;i++){
        p=new node;

        p->d=i;
        p->next=NULL;
        r->next=p;

        r=p;
    }

    r->next=head;r=head;

    for(i=1;i<=n;i++){
        for(j=1;j<=m-2;j++)
            r=r->next;

        cout<<r->next->d<<" ";
```

```
        r->next=r->next->next;
        r=r->next;
    }
}
```

```
输入：8 5
输出：5 2 8 7 1 4 6 3
```

全文完