链表

本节选自一本通

8.3-0

```
#include<iostream>//1. 单链表结构、建立、输出
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）
int x;
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x;
	}
	p=head->next;// 头指针没有数据，从头指针后面的也就是第一个结点开始
	while(p!=NULL)
	{
		cout<<p->data<<" ";
		p=p->next;
	}
	return 0;
}
```

```
1 2 3 4 5 -1
1 2 3 4 5
```

8.3-1a

```
#include<iostream>//1a. 查找结点
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）
int x,y;//x 是输入的结点，y 是需要查找的结点
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x; // 输入结点
	}

	cin>>y;
	p=head->next;// 查找结点
	while(p->data!=y&&p->next!=NULL)// 该段程序找到第一个数据立即回应
	{		//p!=NULL 会出错误
		p=p->next;
	}
	if(p->data==y) cout<<" 找到！ ";
	else cout<<" 没找到！ ";
	return 0;
}
```

```
3 5 6 6 5 -1
5
找到！
```

```
3 5 6 6 5 -1
7
没找到！
```

8.3-1b

```
#include<iostream>//1b. 查找结点
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）
int x,y;//x 是输入的结点，y 是需要查找的结点
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x; // 输入结点
	}

	cin>>y;
	p=head->next;// 查找结点
	while(p!=NULL)
	{
		if(p->data==y) cout<<" 找到！ ";
		p=p->next;
	}
	return 0;
}
```

```
3 6 6 6 -1
6
找到！ 找到！ 找到！
```

8.3-2

```
#include<iostream>//2. 取出第 i 点数据域
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）

void quchu(int y)
{
	int j;
	p=head->next;
	j=1;
	while(p!=NULL&&j<y)
	{
		p=p->next;
		j=j+1;
	}
	if(p!=NULL&&j==y) cout<<p->data;
	else cout<<y<<"not exsit";
}

int x,y;//x 是输入的结点，y 是需要取出的结点
```

```
3 7 8 9 10 -1
5
10
```

```
3 7 8 9 10 -1
20
20 not exsit
```

```
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)//-1 结束输入
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x; // 输入结点
	}
	cin>>y;
	quchu(y);
	return 0;
}
```

8.3-3

```
#include<iostream>//3. 插入结点
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）

int x,y,z;
void insert(int z,int y)// 插入结点
{
	Node *s;//s 是创建插入的结点
	int j=0;
	p=head;
	while(p!=NULL&&j<z)
	{
		p=p->next;
		j=j+1;
	}
	if(p==NULL) cout<<"no this position";
	else
	{
		s=new Node;
		s->data=y;
		s->next=p->next;
		p->next=s;
	}
}
```

```
5 6 7 8 -1
5 6 7 8
3 9
5 6 7 9 8
```

```
5 6 7 8 -1
5 6 7 8
20 22
no this position5 6 7 8
```

```
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x; // 输入链表
	}

	p=head->next;// 输出链表
	while(p!=NULL)
	{
		cout<<p->data<<" ";
		p=p->next;
	}

	cin>>z>>y;//z 插入位置，y 数据
	insert(z,y);
	p=head->next;// 输出链表
	while(p!=NULL)
	{
		cout<<p->data<<" ";
		p=p->next;
	}

	return 0;
}
```

8.3-4

```
#include<iostream>//4. 删除链表
#include<cstdlib>// 这个头文件定义 free 函数
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）
int x,y;
void shanchu(int y)// 删除第 y 个链表
{
	Node *s;
	int j=0;
	p=head;
	while(p->next!=NULL&&j<y-1)
	{
		p=p->next;
		j=j+1;
	}
	if(p->next==NULL) cout<<"no this position";
	else
	{
		s=p->next;
		p->next=p->next->next;// 或者 p->next=s->next
		free(s); // 释放刚才删除结点占用的内存空间
	}
}
```

```
5 6 7 8 -1
5 6 7 8
3
5 6 8
```

```
5 6 7 8 -1
5 6 7 8
20
no this position5 6 7 8
```

```
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x;
	}

	p=head->next;// 输出链表
	while(p!=NULL)
	{
		cout<<p->data<<" ";
		p=p->next;
	}

	cin>>y;// 删除 y 结点
	shanchu(y);
	p=head->next;// 输出删除 y 结点后的链表
	while(p!=NULL)
	{
		cout<<p->data<<" ";
		p=p->next;
	}
	return 0;
}
```

8.3-5

```
#include<iostream>//5. 求单链表长度
using namespace std;
struct Node
{
	int data;
	Node *next;
};
Node *head,*p,*r;//r 指向链表的当前结点（最后结点）

int len()// 求链表长度函数
{
	int n=0;
	p=head;
	while(p!=NULL)
	{
		n=n+1;
		p=p->next;
	}
	cout<<n-1;
}

int x;
```

```
5 6 7 8 -1
5 6 7 8
长度：4
```

```
int main()
{
	cin>>x;
	head=new Node;// 创建头结点
	r=head;
	while(x!=-1)
	{
		p=new Node;// 申请新结点
		p->data=x;
		p->next=NULL;
		r->next=p;
		r=p;
		cin>>x;
	}

	p=head->next;// 输出链表
	while(p!=NULL)
	{
		cout<<p->data<<" ";
		p=p->next;
	}
	cout<<endl<<" 长度： ";
	len();// 链表长度

	return 0;
}
```

8.3-6

```
#include<iostream>// 双向链表创建及输出
using namespace std;
struct node
{
	int data;
	node *next,*pre;
}*head,*tail,*p;
int i,n;
int main()
{
	head=new node;// 建立头结点
	head->next=NULL;
	head->pre=NULL;
	tail=head;// 建立头结点

	cin>>n;
	for(i=1;i<=n;++i)// 输入链表
	{
		p=new node;
		cin>>p->data;

		p->pre=tail;
		tail->next=p;
		p->next=NULL;
		tail=p;
	}

	p=new node;// 建立尾结点
	p->pre=tail;
	p->next=NULL;
	tail->next=p;
	tail=p;// 建立尾结点
```

```
4
2 3 5 6
2 3 5 6
6 5 3 2
```

```
	p=head->next;
	while(p!=tail)//正向输出
	{
		cout<<p->data<<" ";
		p=p->next;
	}
	cout<<endl;

	p=tail->pre;
	while(p!=head)//反向输出
	{
		cout<<p->data<<" ";
		p=p->pre;
	}
	return 0;
}
```

8.3-7

```
#include<iostream>// 双向循环链表创建及输出
using namespace std;
struct node
{
	int data;
	node *next,*pre;
}*head,*tail,*p;
int i,n;
int main()
{
	head=new node;// 建立头结点
	head->next=NULL;
	head->pre=NULL;
	tail=head;// 建立头结点
	head->data=0;

	cin>>n;
	for(i=1;i<=n;++i)// 输入链表
	{
		p=new node;
		cin>>p->data;

		p->pre=tail;
		tail->next=p;
		p->next=NULL;
		tail=p;
	}
```

head、tail 都设置为 0

```
4
2 3 5 6
2 3 5 6 0 0 2 3 5 6 0 0
6 5 3 2
```

```
        p=new node;// 建立尾结点
        p->pre=tail;
        p->next=NULL;
        tail->next=p;
        tail=p;// 建立尾结点
        p->data=0;
        tail->next=head;// 建立循环

        p=head->next;
        for(i=1;i<=n*3;i++)// 正向输出，观察循环
        {
                cout<<p->data<<" ";
                p=p->next;
        }
        cout<<endl;

        p=tail->pre;
        while(p!=head)// 反向输出
        {
                cout<<p->data<<" ";
                p=p->pre;
        }
        return 0;
}
```

指针和链表一本通训练指导

8.3-1

```
#include<iostream>//1. 交换变量
using namespace std;
int a,b,t;
int *p,*q;
int main()
{
	cin>>a>>b;
	p=&a;
	q=&b;
	t=*p,*p=*q,*q=t;
	cout<<a<<" "<<b;
	return 0;
}
```

8.3-2

```
#include<iostream>//2. 字符串倒序输出
#include<cstring>
using namespace std;
const int MaxLen=1e5+10;//const 常量 ,MaxLen=1e5+10 非常大的数字
int len;
char s[MaxLen],*p;
int main()
{
	cin>>s;
	len=strlen(s);
	p=s+len-1;//p 指向字符串最后一个字符
	do
	{
		cout<<*p;
		p--;
	}while(p>=s);
	return 0;
}
```

8.3-3

```
#include<iostream>//3. 查找字符
using namespace std;
const int MaxLen=1e5+10;
char ch;//ch 表示需要查找的字符
char s[MaxLen];//s 表示输入的字符数组

char* FindChar(char *p,char ch)// 函数返回值是指针
{//p 指针刚开始指向字符数组的第一个字符
	while(*p!=ch&&*p!='\0') p++;
	if(*p==ch) return p;
	else return NULL;
}
int main()
{
	cin>>s>>ch;

	char *ans=FindChar(s,ch);// 函数返回的地址记录到 ans
	if(ans!=NULL)// 如果 ans 不为 NULL
		cout<<ans-s+1;// 计算在字符数组中的序号
	else cout<<"no";
	return 0;
}
```

```
abcd
d
4
```

```
abcd
f
no
```

8.3-4

```
#include<cstdio>
#include<iostream>//4. 约瑟夫问题
using namespace std;
struct node{
	int num;// 记录这个节点对应猴子的编号
	node *next,*pre;//next 指向节点的后继，pre 指向节点的前驱
}*head,*tail,*p; //head 表示头指针，tail 表示尾指针
int n,m,i,j;
int main()
{
	while(scanf("%d%d",&n,&m),n||m)// 判断 n,m 是否为 0,0
	{
		head=new node;
		head->num=1;
		head->next=head->pre=NULL;
		tail=head;// 先建立编号为 1 的结点，头指针和尾指针都指向它

	for(i=2;i<=n;i++)
	{
		p=new node;
		p->num=i;// 建立编号为 i 的结点
		tail->next=p;
		p->pre=tail;//p 指向的结点插入到 tail 指向的结点的后面
		p->next=NULL;
		tail=p;//tail 更新为 p，下次就插在它后面
	}
```

```
        head->pre=tail;
        tail->next=head;// 循环链表，头尾相连
        p=head;
        for(i=1;i<n;i++)// 循环 n-1 次，删除一个结点
        {
                for(j=1;j<m;j++)
                {
                        p=p->next;// 经过 m-1 次循环，p 指针指向到要删除的结点
                }
                p->next->pre=p->pre;
                p->pre->next=p->next;// 删除结点 p
                p=p->next;// 剩下的猴子从刚删除的猴子的后继开始报数

        }
        printf("%d\n",p->num);// 剩余最后一个结点即答案
        }
        return 0;
}
```

```
6 2
5
12 4
1
8 3
7
0 0
```

删除数组中的元素（链表）

【题目描述】

给定 N 个整数，将这些整数中与 M 相等的删除。

假定给出的整数序列为：1,3,3,0,-3,5,6,8,3,10,22,-1,3,5,11,20,100,3,9,3。

应该将其放在一个链表中，链表长度为 20。

要删除的数是 3，删除后，链表中只剩 14 个元素：1 0 -3 5 6 8 10 22 -1 5 11 20 100 9

要求：必须使用链表，不允许使用数组，也不允许不删除元素直接输出。

程序中必须有链表的相关操作：建立链表，删除元素，输出删除后链表中元素，释放链表。

【输入格式】

输入包含 3 行：

第一行是一个整数 n(1 <= n <= 200000)，代表数组中元素的个数。

第二行包含 n 个整数，代表数组中的 n 个元素。每个整数之间用空格分隔；每个整数的取值在 32 位有符号整数范围以内。

第三行是一个整数 k，代表待删除元素的值（k 的取值也在 32 位有符号整数范围内）。

【输出格式】

输出只有 1 行：将数组内所有待删除元素删除以后，输出数组内的剩余元素的值，每个整数之间用空格分隔。

【样例输入】

20

1 3 3 0 -3 5 6 8 3 10 22 -1 3 5 11 20 100 3 9 3

3

【样例输出】

1 0 -3 5 6 8 10 22 -1 5 11 20 100 9

8.3-5

```
#include<iostream>//5. 删除元素
using namespace std;
struct node
{
	int num;
	node *next,*pre;
}*head,*tail,*p;
int n,k;
int main()
{
	cin>>n;
	head=new node;// 新建一个虚拟头结点（不存储元素的值）
	head->pre=head->next=NULL;
	tail=head; // 方便删除操作（不会越界），此时头尾指针都指向它

	for(int i=1;i<=n;i++)
	{
		p=new node;
		cin>>p->num;// 输入元素的值，储存到新建的结点中
		p->pre=tail;
		tail->next=p;//p 结点插入到 tail 结点的后面
		p->next=NULL;
		tail=p;// 将 tail 更新为 p，下次插在它后面
	}

	p=new node;// 建立虚拟尾结点，方便删除和查询、输出
	p->pre=tail;
	p->next=NULL;
	tail->next=p;
	tail=p;
```

```
    cin>>k;
    p=head->next;//p 开始指向虚拟头结点的后继，即第 1 个元素
    while(p!=tail)// 遍历到虚拟尾结点停止
    {
        if(p->num==k)// 如果找到与 k 相同的元素，删除对应结点
        {
            p->next->pre=p->pre;
            p->pre->next=p->next;
        }
        p=p->next;
    }

    p=head->next;////p 指向虚拟头结点的后继，即第 1 个元素
    while(p!=tail)
    {
        cout<<p->num<<" ";
        p=p->next;
    }
    return 0;
}
```

统计学生信息（使用动态链表完成）

【题目描述】

利用动态链表记录从标准输入输入的学生信息（学号、姓名、性别、年龄、得分、地址）。其中，学号长度不超过 20，姓名长度不超过 40，性别长度为 1，地址长度不超过 40。

【输入格式】

包括若干行，每一行都是一个学生的信息，如：

00630018 zhouyan m 20 10.0 28#460

输入的最后以”end”结束。

【输出格式】

将输入的内容倒序输出。每行一条记录，按照下面的格式输出：

学号 姓名 性别 年龄 得分 地址

【样例输入】

00630018 zhouyan m 20 10 28#4600

0063001 zhouyn f 21 100 28#460000

0063008 zhoyan f 20 1000 28#460000

0063018 zhouan m 21 10000 28#4600000

00613018 zhuyan m 20 100 28#4600

00160018 zouyan f 21 100 28#4600

01030018 houyan m 20 10 28#4600

0630018 zuyan m 21 100 28#4600

10630018 zouan m 20 10 28#46000

end

【样例输出】

10630018 zouan m 20 10 28#46000

0630018 zuyan m 21 100 28#4600

01030018 houyan m 20 10 28#4600

00160018 zouyan f 21 100 28#4600

00613018 zhuyan m 20 100 28#4600

0063018 zhouan m 21 10000 28#4600000

0063008 zhoyan f 20 1000 28#460000

0063001 zhouyn f 21 100 28#460000

00630018 zhouyan m 20 10 28#4600

8.3-6

```
#include<iostream>//6. 统计学生信息 -- 倒序输出，倒序链表
#include<cstring>
using namespace std;
struct data
{
	char num[25],name[45],sex,score[45],address[45];
	int age;
	bool scan()
	{
		cin>>num;
		if(strcmp(num,” end” )==0) return true;// 输入 end 结束，返回 true
		cin>>name>>sex>>age>>score>>address; return false;// 输入没有停止，
返回 false
	}
	void print()
	{
		cout<<num<<" "<<name<<" "<<sex<<" "<<age<<" "<<score<<"
"<<address<<endl;
	}
};
struct node
{
	data d;//d 表示该结点对应元素的信息
	node *next,*pre;
}*head,*tail,*p;
int n,k;
```

```
int main()
{
	head=new node;// 新建虚拟头结点
	head->pre=head->next=NULL;
	tail=head;

	while(1)
	{
		p=new node;
		if(p->d.scan()) break;// 输入信息。如果他们停止输入，跳出循环
		p->pre=tail;
		tail->next=p;//p 指向的结点插入到 tail 指向的结点的后面
		p->next=NULL;
		tail=p;//tail 更新为 p，下次插在它后面
	}
	p=tail;
	while(p!=head)
	{
		p->d.print();
		p=p->pre;

	}
	return 0;
}
```