数据结构

struct 结构
void 不返回
del delete 删除
deldata 删除数据
add 增加
null 零值
insert 插入
delpos 删除数据
value 数值

pointer 指针
stack 栈
init 建栈
push 入栈
display 显示
pop 出栈
gettop 取栈顶
empty 测试栈
setnull 释放栈
queue 队列
front 队头
rear 队尾
getlen 获得长度
get 获得
len 长度
data 数据

栈

```
#include <iostream>// 栈的代码实现 （选自东方博宜）
// 常量：栈的长度
#define MAXN 5
using namespace std;
int stack[MAXN];// 数组模拟栈
int top = -1;// 初始化栈指针
// 出栈
int pop()
{
	int r;
	if(top < 0)
	{
		r = -1;
		cout<<" 栈空！ "<<endl;
	}
	else
	{
		r = stack[top];
		top--;
	}
	return r;
}
// 入栈
```

```
// 入栈
void push(int value)//value 价值
{
	if(top >= MAXN - 1)
	{
		cout<<" 栈满 "<<endl;
	}
	else
	{
		top++;
		stack[top] = value;
	}
}
// 显示栈
void display()//display 显示
{
	int i;
	for(i = top;i >= 0;i--)
	{
		cout<<stack[i]<<" ";
	}
	cout<<endl;
}
```

```
int main()
{
	int order;//指令
	int x,t;
	cout<<"输入指令：  ";
	while(1 == 1)
	{
		cout<<"1:入栈,2:出栈,3:显示栈!"<<endl;
		cin>>order;
		if(order == 1)
		{
			cin>>x;
			push(x);
			display();
		}
		else if(order == 2)
		{
			t = pop();
			if(t != -1)
			{
				cout<<t<<"出栈!"<<endl;
				display();
			}
		}
		else if(order == 3)
		{
			display();
		}
	}
	return 0;
}
```

队列

```
#include <iostream>// 队列代码实现（选自东方博宜）
#define MAXN 5
using namespace std;
int queue[MAXN] = {0}; // 队列
int front = 0;// 头指针
int rear = 0; // 尾指针

void addqueue(int value)// 入队
{
	if(rear >= MAXN)
	{
		cout<<" 队满！"<<endl;
	}
	else
	{
		queue[rear] = value;
		rear++;
	}
}

// 出队
```

```
int delqueue()//出队
{
	int r;
	if(front == rear)
	{
		cout<<"队空"<<endl;
		r = -1;
	}
	else
	{
		r = queue[front];
		front++;
	}
	return r;
}
//显示队
void display()
{
	int i;
	for(i = front;i < rear;i++)
	{
		cout<<queue[i]<<" ";
	}
	cout<<endl;
}
```

```
int main()
{
	int order;// 口令
	int value,t;
	cout<<" 输入指令 "<<endl;
	while(1 == 1)
	{
		cout<<"1 入队,2 出队,3 显示! "<<endl;
		cin>>order;
		if(order == 1)
		{
			cin>>value;
			addqueue(value);
			display();
		}
		else if(order == 2)
		{
			t = delqueue();
			if(t != -1)
			{
				cout<<t<<" 已经出队 "<<endl;
				display();
			}
			else
			{
				cout<<" 队列已空 "<<endl;
			}
		}
		else if(order == 3)
		{
			display();
		}
		else  {	cout<<" 口令错误! "<<endl;	}
	}
}
```

```cpp
#include <iostream>// 循环队列代码实现 （选自东方博宜）
#define MAXN 5
using namespace std;
int queue[MAXN] = {0}; // 队列
int front = 0;// 头指针
int rear = 0; // 尾指针

void addqueue(int value)// 入队
{

	if(front == 0 && (rear + 1) % MAXN == front)// 元素从未出队的情况下，队满
	{
		cout<<" 队满 "<<endl;

	}
	else if((rear + 1) % MAXN == front)// 有元素出队，队满
	{
		cout<<" 队满 "<<endl;
	}
	else
	{
		queue[rear] = value;
		rear = (rear + 1) % MAXN;
	}
}
```

```
int delqueue()// 出队
{
	int r;
	if(front == rear)
	{
		cout<<" 队空 "<<endl;
		r = -1;
	}
	else
	{
		r = queue[front];
		queue[front] = -1;// 出队标记
		front = (front + 1) % MAXN;
	}
	return r;
}

void display()// 显示队
{
	if(front == rear)
	{
		cout<<" 队空 "<<endl;
	}
	else
	{
		int i = front;
		do
		{
			cout<<queue[i]<<" ";
			i = (i + 1) % MAXN;
		}
		while(i != rear);
	}
	cout<<endl;
}
```

```
int main()
{
	int order;// 口令
	int value,t;
	cout<<" 输入指令 "<<endl;
	while(1 == 1)
	{
		cout<<"1 入队,2 出队,3 显示! "<<endl;
		cin>>order;
		if(order == 1)
		{
			cin>>value;
			addqueue(value);
			display();
		}
		else if(order == 2)
		{
			t = delqueue();
			if(t != -1)
			{
				cout<<t<<" 已经出队 "<<endl;
				display();
			}
			else
			{
				cout<<" 队列已空 "<<endl;
			}
		}
		else if(order == 3)
		{
			display();
		}
		else  {	cout<<" 口令错误! "<<endl;	}
	}
}
```

结构体

本节选自一本通

【题目描述】

输入学生的人数，然后再输入每位学生的分数和姓名，求获得最高分数的学生的姓名。

【输入】

第一行输入一个正整数 N（N <= 100），表示学生人数。接着输入 N 行，每行格式如下：

分数 姓名

分数是一个非负整数，且小于等于100；

姓名为一个连续的字符串，中间没有空格，长度不超过 20。

数据保证最高分只有一位同学。

【输出】

获得最高分数同学的姓名。

【输入样例】

```
5
87 lilei
99 hanmeimei
97 lily
96 lucy
77 jim
```

【输出样例】

```
hanmeimei
```

8.0-1a

```cpp
#include<iostream>
using namespace std;
struct node// 结构体
{
	string name;// 姓名
	int score;// 分数

} a[101];
int main()
{
	int n,i,j;
	int max=0,k;
	cin>>n;
	for(i=0;i<n;i++)
	{
		cin>>a[i].score;
		cin>>a[i].name;
		if(a[i].score>max)
		{
			max=a[i].score;
			k=i;
		}
	}
	cout<<a[k].name;
	return 0;
}
```

8.0-1b

```
#include<iostream>
using namespace std;
struct node// 结构体
{
	string name;// 姓名
	int score;// 分数

} a[101];
int main()
{
	int n,i,j;
	cin>>n;
	for(i=0;i<n;i++)
	{
		cin>>a[i].score;
		cin>>a[i].name;
	}
	for(i=n-1;i>0;i--)// 冒泡排序
	{
		for(j=0;j<i;j++)
		{
			if(a[j].score<a[j+1].score)
			swap(a[j],a[j+1]);
		}
	}
	cout<<a[0].name;
	return 0;
}
```

```
m n
m n
109 110
```

```
abc
def
ert
```

【题目描述】
目前正是甲流盛行时期，为了更好地进行分流治疗，医院在挂号时要求对病人的体温和咳嗽情况进行检查，对于体温超过 37.5 度（含等于 37.5 度）并且咳嗽的病人初步判定为甲流病人（初筛）。现需要统计某天前来挂号就诊的病人中有多少人被初筛为甲流病人。

【输入】
第一行是某天前来挂号就诊的病人数 n。(n<200)

其后有 n 行，每行是病人的信息，包括三个信息：姓名（字符串，不含空格，最多 8 个字符）、体温（float）、是否咳嗽（整数，1 表示咳嗽，0 表示不咳嗽）。每行三个信息之间以一个空格分开。

【输出】
按输入顺序依次输出所有被筛选为甲流的病人的姓名，每个名字占一行。之后在输出一行，表示被筛选为甲流的病人数量。

【输入样例】
5
Zhang 38.3 0
Li 37.5 1
Wang 37.1 1
Zhao 39.0 1
Liu 38.2 1
【输出样例】
Li
Zhao
Liu
3

8.0-2

```
#include<iostream>//2. 甲流病人初筛
#include<cstring>
using namespace std;
struct node{
	string name;
	float tw;// 体温
	int ks;// 咳嗦
}a[200];
int main()
{
	int n,i,k=0;//k 统计甲流病人数量
	cin>>n;
	for(i=0;i<n;i++)
	{
		cin>>a[i].name>>a[i].tw>>a[i].ks;
	}
	for(i=0;i<n;i++)
	{
		if(a[i].tw>=37.5&&a[i].ks==1)
		{
			cout<<a[i].name<<endl;
			k++;
		}
	}
	cout<<k;
	return 0;
}
```

【题目描述】
在一次考试中，每个学生的成绩都不相同，现知道了每个学生的学号和成绩，求考第 k 名学生的学号和成绩。

【输入】
第一行有两个整数，分别是学生的人数 n（1 ≤ n ≤ 100），和求第 k 名学生的 k（1 ≤ k ≤ n）。

其后有 n 行数据，每行包括一个学号（整数）和一个成绩（浮点数），中间用一个空格分隔。

【输出】
输出第 k 名学生的学号和成绩，中间用空格分隔。（注：请用 %g 输出成绩）

【输入样例】
5 3
90788001 67.8
90788002 90.3
90788003 61
90788004 68.4
90788005 73.9
【输出样例】
90788004 68.4

8.0-3

```
#include<iostream>// 谁考了第 k 名
#include<cstring>
using namespace std;
struct node{
	char number[10];// 字符，学号
	double score;// 成绩
}a[101];

int main()
{
	int n,i,j=0,t,k;
	cin>>n>>k;//n 名学生，第 k 名学生学号和成绩
	for(i=1;i<=n;i++)
	{
		cin>>a[i].number>>a[i].score;
	}
	for(i=1;i<n;i++)// 选择排序
	{
		for(j=i+1;j<=n;j++)
		{
			if(a[i].score<a[j].score) swap(a[i],a[j]);
		}
	}
	cout<<a[k].number<<" "<<a[k].score;
	return 0;
}
```

【题目描述】
给出班里某门课程的成绩单，请你按成绩从高到低对成绩单排序输出，如果有相同分数则名字字典序小的在前。

【输入】
第一行为 n (0 < n < 20)，表示班里的学生数目；

接下来的 n 行，每行为每个学生的名字和他的成绩， 中间用单个空格隔开。名字只包含字母且长度不超过 20，成绩为一个不大于 100 的非负整数。

【输出】
把成绩单按分数从高到低的顺序进行排序并输出，每行包含名字和分数两项，之间有一个空格。

【输入样例】
4
Kitty 80
Hanmeimei 90
Joey 92
Tim 28
【输出样例】
Joey 92
Hanmeimei 90
Kitty 80
Tim 28

8.0-4

```
#include<iostream>//4. 成绩排序
#include<cstring>
using namespace std;
struct node{
	char name[20];
	int score;
}a[20];
int main()
{
	int i,j,n;
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>a[i].name>>a[i].score;
	}
	for(i=1;i<n;i++)
	{
		for(j=i+1;j<=n;j++)
		{//strcmp 字符串比较函数
			if(a[i].score<a[j].score||a[i].score==a[j].score&&strcmp(a[i].
name,a[j].name)>0)
				swap(a[i],a[j]);
		}
	}
	for(i=1;i<=n;i++)
	{
		cout<<a[i].name<<" "<<a[i].score<<endl;
	}
	return 0;
}
```

【题目描述】
病人登记看病，编写一个程序，将登记的病人按照以下原则排出看病的先后顺序：

1. 老年人（年龄 >= 60 岁）比非老年人优先看病。

2. 老年人按年龄从大到小的顺序看病，年龄相同的按登记的先后顺序排序。

3. 非老年人按登记的先后顺序看病。

【输入】
第 1 行，输入一个小于 100 的正整数，表示病人的个数；

后面按照病人登记的先后顺序，每行输入一个病人的信息，包括：一个长度小于 10 的字符串表示病人的 ID（每个病人的 ID 各不相同且只含数字和字母），一个整数表示病人的年龄，中间用单个空格隔开。

【输出】
按排好的看病顺序输出病人的 ID，每行一个。

【输入样例】
5
021075 40
004003 15
010158 67
021033 75
102012 30
【输出样例】
021033
010158
021075
004003
102012

8.0-5a

```
#include<iostream>//5. 病人排队
#include<cstring>
using namespace std;
struct node{
	char id[10];// 病人编号 id
	int age;// 年龄
}c[101];
int main()
{
	int n,i,j;
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>c[i].id>>c[i].age;
	}
	for(i=n;i>=1;i--)// 冒泡排序
	{
		for(j=1;j<=i;j++)
		{
			if(c[j].age<c[j+1].age&&c[j+1].age>=60)
				swap(c[j],c[j+1]);
		}
	}
	for(i=1;i<=n;i++)
	{
		cout<<c[i].id<<endl;
	}
	return 0;
}
```

8.0-5b

```
#include<iostream>//5. 病人排队
using namespace std;
struct node{
	char id[10];// 病人编号
	int age;// 年龄
};
node a[101],b[101],c,t;
int main()
{
	int n,f=0,s=0,i,j;
	cin>>n;
	for(i=0;i<n;i++)
	{
		cin>>c.id>>c.age;
		if(c.age>=60) a[f++]=c;//a 数组储存 60 以上的
		else b[s++]=c;// b 数组储存 60 以下的
	}
	for(i=0;i<f-1;i++) // 选择排序
	{
		for(j=i+1;j<f;j++)
		{
			if(a[i].age<a[j].age)
			{
				swap(a[i].age,a[j].age);
				swap(a[i].id,a[j].id);
			}
		}
	}
	for(i=0;i<f;i++)// 输出 60 以上老人（经过排序）
		cout<<a[i].id<<endl;
	for(i=0;i<s;i++)
		cout<<b[i].id<<endl;
	return 0;
}
```

共用体

例如，身份证和学籍号可以填写在同一个变量中。

指针

```
#include<iostream>// 指针输入输出
using namespace std;
int a,b;
int *p,*q;
int main()
{
	cin>>a>>b;
	p=&a;
	q=&b;
	cout<<*p<<" "<<*q<<endl;   // 数值
	cout<<p<<" "<<q;    // 地址
	return 0;
}
```

```
35 67
35 67
0x4a3020 0x4a3024
```

8.2-2

```
#include <bits/stdc++.h>// 指针 （选自东方博宜）
using namespace std;
int main(){
	int x = 10;

	int *p = &x;		// 定义指针，指向 x 的地址

	cout<<p<<endl;//p 是 int* 类型的指针（整数的地址）

	cout<<*p<<endl;//*p 不是地址，是 p 这个地址指向的值
	*p = *p + 2;// 修改指针指向的变量的值
	cout<<p<<" "<<*p<<endl;
	cout<<x<<endl;

	int arr[5] = {0}; // 数组的本质是数组中下标为 0 的元素的地址
	cout<<&arr<<" "<<&arr[0]<<" "<<arr<<endl;

	int a,b;
	scanf("%d%d",&a,&b);	// 采用 scanf 读入
	printf("a=%d\n",a);
	printf("b=%d\n",b);
	printf("%d+%d=%d\n",a,b,a+b);

	return 0;
}
```

```
0x6cfed8
10
0x6cfed8 12
12
0x6cfec4 0x6cfec4
0x6cfec4
35 56
a=35
b=56
35+56=91
```

8.2-3

```cpp
#include <bits/stdc++.h>// 指针与普通变量区别（选自东方博宜）
using namespace std;
int main()
{
	int x = 10;
	int y = x;
	y = y + 2;
	cout<<y<<" "<<x<<endl;

	int *p = &x;
	*p = *p + 2;
	cout<<*p<<" "<<x<<endl;

	int a = 10;
	int *p2 = &a;
	cout<<p2<<" "<<a<<endl;

	(*p2)++;// 注意 ++ 的优先级高于 *
	cout<<*p2<<" "<<a<<endl;
	return 0;
}
```

```
12 10
12 12
0x6cfedc 10
11 11
```

8.2-4

```
#include <bits/stdc++.h>// 结构体指针（选自东方博宜）
using namespace std;
struct stu
{
	char name[100];
	double score;
};
int main()
{
	struct stu s;	// 定义结构体变量
	strcpy(s.name,"ZhangSan");//strcpy 字符串复制
	s.score = 99.99;
	cout<<s.name<<" "<<s.score<<endl;	// 输出结构体
	stu *stu1 = &s;	// 定义指针，指向结构体
	// 在结构体指针中，不能用 . 来指向结构体成员变量（stu 是地址）
	cout<<stu1->name<<" "<<stu1->score<<endl;

	cout<<(*stu1).name<<" "<<(*stu1).score<<endl;	//(*stu) 是结构体
	//new 结构体对象，直接赋值给一个指针
	stu *stu2 = new stu;
	strcpy(stu2->name,"ZhangSan");
	stu2->score = 99.8;
	cout<<stu2->name<<" "<<stu2->score<<endl;
	delete stu2;// 释放内存
	//malloc 结构体对象，直接赋值给一个指针
	stu *stu3 = NULL;
	stu3 = (stu*)malloc(sizeof(stu));
	strcpy(stu3->name,"WangWu");
	stu3->score = 98;
	cout<<stu3->name<<" "<<stu3->score<<endl;
	free(stu3);// 释放内存
	return 0;
}
```

```
ZhangSan 99.99
ZhangSan 99.99
ZhangSan 99.99
ZhangSan 99.8
WangWu 98
```

8.2-5

```
#include <bits/stdc++.h>// 通过函数输出结构体（选自东方博宜）
using namespace std;
struct stu {
	string name;
	double score;
};
void show(struct stu s)
{
	cout<<s.name<<" "<<s .score<<endl;
}
void show2(struct stu *s)
{
	cout<<s->name<<" "<<s->score<<endl;
}
int main()
{
	struct stu s;
	s.name = "ZhangSan";
	s .score = 99;
	show(s);
	stu *t = &s;
	show2(t);
	return 0;
}
```

```
ZhangSan 99
ZhangSan 99
```

链表

本节选自东方博宜

```
#include <bits/stdc++.h>// 单向链表代码实现（选自东方博宜）
using namespace std;

struct Node{                    //结点定义
	int data;
	struct Node *next;
};

Node *head = NULL; //头结点     null 零值

void add(int x)//在链表末尾追加
{
	if(head != NULL)//如果链表中有数据
	{
		Node *a = new Node;             //要追加的节点
		a->data = x;
		a->next = NULL;

		Node *p = head;           //移动到最后一个节点
		while(p->next != NULL)
		{
			p = p->next;
		}
		p->next = a;
	}
	else//如果链表中没有数据
	{
		head = new Node;//创建新节点
		head->data = x;
		head->next = NULL;
	}
}
```

```
void insert(int n,int x)// 在中间追加元素（在第 n 个元素的位置）
{
	Node *d = new Node;// 新节点
	d->data = x;
	d->next = NULL;

	if(n == 1)// 如果是头结点
	{
		d->next = head;
		head = d;
	}
	else
	{
		int i;
		Node *p = head;
		for(i = 1;i <= n - 2;i++)
		{
			p = p->next;
			if(p == NULL)
			{
				break;
			}
		}
		if(p == NULL)
		{
			cout<<"n 有误 "<<endl;
		}
		else
		{
			d->next = p->next;
			p->next = d;
		}
	}
}
```

```
void deldata(int data)// 删除某个数值
{
	Node *p = head,*pre = NULL;
	while(p != NULL)// 如果链表中有数据
	{
		if(data == p->data)
		{
			if(p == head)
			{
				head = p->next;
			}
			else
			{
				pre->next = p->next;
			}
			delete p;//delete 删除
			break;
		}
		pre = p;
		p = p->next;// 遍历链表，查找需要删除的数值
	}
}
```

```
void delpos(int n)// 删除某个位置的元素
{
	Node *p = head,*t;
	if(n == 1)  // 如果要删除头节点
	{
		if(head != NULL)
		{
			head = head->next;
			delete p;
		}
		else
		{
			cout<<" 链表空 "<<endl;
		}
	}
	else
	{
		int i;
		for(i = 1;i <= n - 2;i++)		// 移动到要删除位置之前的节点
		{
			p = p->next;
			if(p == NULL) break;
		}
		if(p == NULL || p->next == NULL)
		{
			cout<<"n 的值有误 !"<<endl;
		}
		else
		{
			t = p->next;
			p->next = t->next;
			delete t;
		}
	}
}
```

```
void display()//输出链表
{
	Node *p = head;
	while(p != NULL)
	{
		cout<<p->data<<" ";
		p = p->next;
	}
	cout<<endl;
}
```

```
int main()
{
	int order,x,p;
	cout<<" 输入指令 :";
	while(1 == 1)
	{
		cout<<"1: 追加 , 2: 插入 , 3: 删除值 , 4: 删除位置 , 5: 显示 !"<<endl;
		cin>>order;
		if(order == 1)
		{
			cin>>x;
			add(x);
			display();
		}
		else if(order == 2)
		{
			cin>>p>>x;
			insert(p,x);
			display();
		}
		else if(order == 3)
		{
			cin>>x;
			deldata(x);
			display();
		}
		else if(order == 4)
		{
			cin>>x;
			delpos(x);
			display();
		}
		else if(order == 5)	{	display();	}
	}
}
```

```
#include <bits/stdc++.h>// 链式栈代码实现（选自东方博宜）
using namespace std;

struct Stack{// 栈元素
	int data;
	struct Stack *next;
};
Stack *top = NULL;    // 栈顶指针

void push(int x)    // 入栈
{
	Stack *p = new Stack;
	p->data = x;
	p->next = top;
	top = p;// 修改栈顶位置
}

void pop()  // 出栈
{
	Stack *p = top;
	if(p != NULL)
	{
		cout<<p->data<<" 出栈 "<<endl;
		p = p->next;
		delete top;
		top = p;// 修改指针位置
	}
	else
	{
		cout<<" 栈空 "<<endl;
	}
}

// 获得栈长度
```

```
int getlen()    //获得栈长度
{
	int len = 0;
	Stack *p = top;
	while(p != NULL)
	{
		len++;
		p = p->next;
	}
	return len;
}

void display()    //显示栈元素
{
	Stack *p = top;
	while(p != NULL)
	{
		cout<<p->data<<" ";
		p = p->next;
	}
	cout<<endl;
}
```

```
int main()
{
	int order,x;
	cout<<" 输入指令 :"<<endl;
	while(1 == 1)
	{
		cout<<"1: 入栈 ,2: 出栈 ,3: 显示 ,4: 求栈长 !"<<endl;
		cin>>order;
		if(order == 1)
		{
			cin>>x;
			push(x);
			display();
		}
		else if(order == 2)
		{
			pop();
			display();
		}
		else if(order == 3)
		{
			display();
		}
		else if(order == 4)
		{
			cout<<getlen()<<endl;
		}
	}
	return 0;
}
```

```
#include <bits/stdc++.h>// 链式队列代码实现（选自东方博宜）
using namespace std;

struct Queue{   // 队列节点
	int data;
	struct Queue *next;
};
Queue *front = NULL;    // 队头
Queue *rear = NULL;    // 队尾

void add(int value)    // 入队
{
	Queue *e = new Queue;// 创建新节点
	e->data = value;
	e->next = NULL;

	if(front == NULL) // 如果队列是空的
	{
		front = e;
	}
	else
	{
		rear->next = e;
	}
	rear = e;
}

  // 出队
```

```
void del()    //出队
{
	Queue *t;

	if(front != NULL) //如果队列有元素
	{
		cout<<front->data<<"出队"<<endl;
		t = front;
		front = front->next;
		if(front == NULL) rear = NULL;
		delete t;
	}
	else
	{
		cout<<"队列空"<<endl;
	}
}

void display()    //显示队
{
	Queue *p = front;
	while(p != NULL)
	{
		cout<<p->data<<" ";
		p = p->next;
	}
	cout<<endl;
}
```

```
int main()
{
	int order,x;
	cout<<" 输入指令 :"<<endl;
	while(1 == 1)
	{
		cout<<"1: 入队 ,2: 出队 ,3: 显示队 !"<<endl;
		cin>>order;
		if(order == 1)
		{
			cin>>x;
			add(x);
			display();
		}
		else if(order == 2)
		{
			del();
			display();
		}
		else if(order == 3)
		{
			display();
		}
	}
	return 0;
}
```

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;
}
```