链表建立头指针head和表头结点：

typedef struct node{....}NODE；//定义结点

NODE \*head; //说明头指针

NODE \* p; //为进行后续相关操作，说明一个指向结点的指针变量p

建立不包含数据的表头结点：

p=(NODE\*)malloc(sizeof(NODE)); //申请表头结点

p->link=NULL; 表头结点的link设置为NULL

head=p; //head指向p；

当要插入结点：

p=（NODE\*）malloc(sizeof(NODE)); //申请一个数据结点；

gets(p ->name) //存数据

p->link =head->link;

head ->link =p;

在第i个结点后插入新结点：

insertnode(NODE\*head,NODE \*P,int i){

NODE \*q;int n=0;

for(q=head;n<i&&q->link!=NULL;++n){q=q->link;} //定位第i个结点

p->link=q->link; q->link=p; }

删除第i个结点：先定位i-1个结点，q指向i-1个结点，p指向被删除的结点。

deletenode（NODE \*head,int i）{

NODE \*q,\*p;int n;

for(n=0,q=head;n<i-1&&q->link!=NULL;++n){q=q->link;}

if(i>0&&q->link!=NULL){

p=q->link;

q->link=p->link;

free (p)} }

文件型指针变量 FILE \*变量名

文件使用方式：1. “r” (只读):打开文本文件用于输入 2.“w”（只写）：打开文本用于输出 3.“a”（追加）：将数据存放到文本文件的尾部 4.“rb”（只读）：打开二进制文件输入 5.“wb”（只写）：打开二进制文件输出 6.“ab”（追加）：将数据存到二进制文件尾部 7.“r+”（读写）：打开文本文件读/写 8.“w+”(读写） 打开新的文本文件读写 9.“a+”（读追加）：打开文本文件读/追加操作 10.“rb+”（读写）：打开二进制文件读写 11.“wb+”（读写）：新二进制文件读写 12.“ab+”打开二进制文件读/追加

打开文件 fopen，原型如下：FILE \*fopen (char \* filename,char \* mode);filename为文件名，mode为文件使用方式，含义如上

fclose 关闭文件 原型如下：int close（FILE \* fp），正常关闭反悔0，关闭发生错误返回非0；

字符输入：fgetc，原型：int fgetc(FILE \* fp); 文件结束时返回文件结束标记EOF,其值为-1。

字符输出：fputc,原型 int fputc(char ch,FILE \*fp),成功时返回输出的字符，失败返回EOF。

字符串输入函数：fgets,原型 char \* fgets(char \*buf,int n,FILE \*fp) 从fp指向的文件读取长度不超过n-1个字符的字符串，放到buf中，操作正确返回buf首地址，操作错误返回NULL。

字符串输出函数 fputs:int fputs(char \*str,FILE \*fp)，str为字符串指针或字符数组名，fp为即将被写入文件的文件型指针。写入文件时，字符‘\0’会自动舍去。调用成功返回0，否则返回EOF。

格式化

格式化输入：fscanf:int fscanf(FILE \*fp,char \* format,...)

格式化输出：fprintf: int fprintf(FILE \*fp,char \*format,...)

文件数据块读函数fread， int fread（char \* buffer,unsigned size,unsigned count,FILE \*fp）;buffer指向数据库存放内存起始地址，size是输入的字节数，count是输入大小为size字节的数据块个数，fp是文件指针。

文件数据块写函数 fwrite,int fwrite(char \*buffer,unsigned size,unsigned count,FILE \*fp);功能如上

改变文件位置：fseek，int fseek(FILE \*fp,long offset,int position);position起点，offset位移量即移动的字节数。失败返回-1.

重返文件头：rewind ,void rewind(FILE \* fp);使fp文件重返开头位置

位置指针当前值 long ftell（FILE \* fp）位置指针从文件开始到当前位置位移字节数

状态检测 int feof(FILE \* fp)当到达文件尾返回0，否则返回非0

错误状态 int ferror(FILE \* fp),文件没错返回0否则返回非0；

清楚错误标志函数： void clearerr(FILE \*fp)，清除fp所指文件的错误标志。文件错误标志和文件结束标志记为0；