线性表

存储 { 顺序
链式

顺序表

```
typedef struct SqList{
    T elem[MAX_SIZE];  // T *elem;
    int size, len;
}SqList;
```

```
void insert(SqList &L, T value, int index){
    for(int i=L.len-1; i>=index; i--){
        L.elem[i+1]=L.elem[i];
    }
    L.elem[index]=value;
    L.len++;
}

T delete(SqList &L, int index){
    T tmp=L.elem[index];
    for(int i=index; i<L.len-1; i++){
        L.elem[i]=L.elem[i+1]
    }
    L.len--;
    return tmp;
}
```

```
void merge(SQList &A, SQList B){
  int len = A.len + B.len - 1;
  int i = A.len - 1, j = B.len - 1;
  while(i >= 0 && j >= 0){
      if(A.elem[i] >= B.elem[j];
      A.elem[len--] = A.elem[i--];
      }else{
      A.elem[len--] = B.elem[j--];
      }
  }
  while(j >= 0)  A.elem[len--] = B.elem[j--];
  while(i >= 0)  A.elem[len--] = A.elem[j--];
  A.len = len;
}
```

單鏈表

```
typedef struct LNode{
        T elem;
        LNode *next;
    }LNode, *LkList;
void insert(LKList &L, T value, int index){
  LKList P = L->next; int i=0;
  while(P && i<index){
        p=p->next; i++;
  }
  LkList s; s->elem = value;
  s->next = p->next;
  p->next = s;
}
```

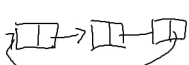
```
T   delete (LKlist &L, int index){
      LKlist p=L->next, q, tmp ; int i=0;
      while(p && i<index){
          q=p; p=p->next ; i++;
      }
      tmp=p;
      q->next = p->next;      // q->next = q->next->next;
       free(p);
      return  tmp->elem;
    }
}
void  create (LKlist &L, T arr[],int len){    // 头插
    for(int i=0 ; i<len; i++){
        LKlist  s;  s->elem = arr[i] ;
       s->next=L->next;
       L->next = s;
    }
}
void  create( LKlist &L, T arr[], int len){   // 尾插
    LKlist  p=L;
    for(int i=0; i<len ; i++){
     LKlist  s;  s->elem = arr[i];   s->next = NULL;
     p->next = s;    p=p->next;
    }
}
```

```
void merge(LKList &A, LKList &B){
    LKList p=A, q=A->next, s=B->next;
    while(q && s){
        if(q->elem >= s->elem){
            p->next=s;
            s=s->next;
        }else{
            p->next=q;
            q=q->next;
        }
        p=p->next;
    }
    if(q) p->next=q;
    if(s) p->next=s;
}
```

静态链表

```
typedef struct LNode{
    T elem;
    int cur;
}LNode, LKList[MAX_SIZE];
```

循环链表



双向链表

```
void reverse(LNList &L){
    p=L->next; L->next=NULL;
    while(p){
      q=p->next;
      p->next=L->next;
      L->next=p;
      p=q;
    }
}
```