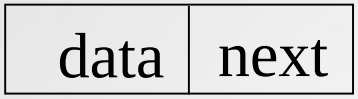
【Project 3】实现 单链表 这种数据结构的基本算法：

- 如何创建一个单链表
- 在单链表中增加一个元素
- 在单链表中删除一个元素
- 单链表的查找

结点和单链表的 C 语言描述

（1）结点的描述



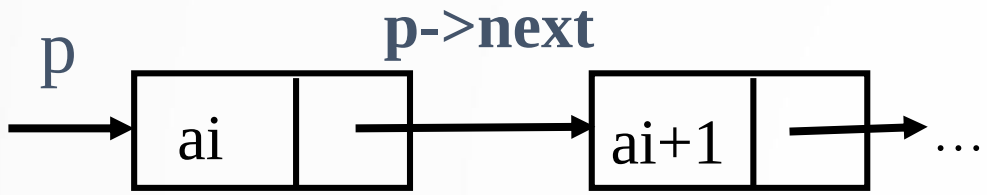

```
typedef struct LNode{
     ElemType          data;
     struct LNode     *next;      // 指向后继的指针域
}LNode, *LinkList
```

例如：LNode *p; 相当于 LinkList p

【读取data】p->data 或者 (*p).data

【读取next】p->next 或者 (*p).next

（2）单链表的描述



算法：表尾插入建立单链表

S1：建立头结点
S2：建立n个数据结点
S3：插入到表尾

```
LinkList creatlink()
{
        LinkList L,p,q;
        int i,n;
        int e;
        L=(LinkList)malloc(sizeof(LNode));
        L->next=NULL;
        q=L;  //q指向当前最后一个结点
        scanf("%d",&n);  //单链表中有n个元素
        for(i=1;i<=n;i++)
        {
                p=(LinkList)malloc(sizeof(LNode));
                scanf("%d",&e); //输入元素值
                p->data=e;
                q->next=p;
                q=p;
        }
        p->next=NULL;
        return L;  //返回头指针
}
```

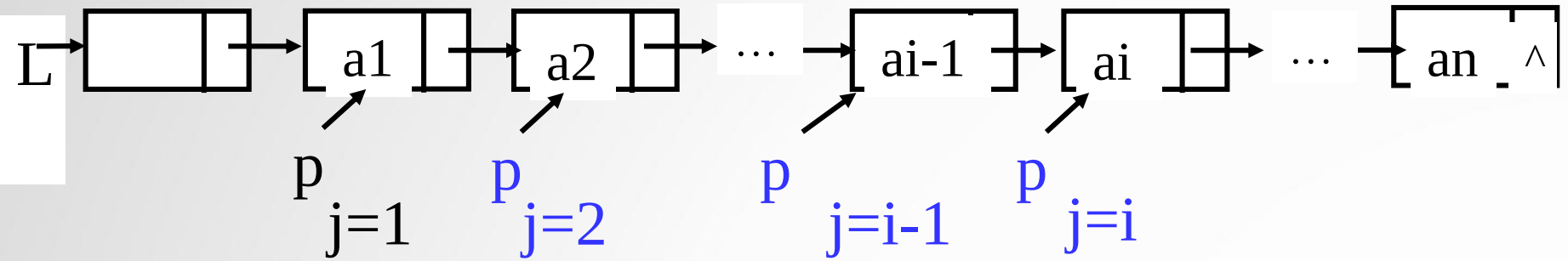
算法：表头插入建立单链表

```
LinkList CreateList_L(LinkList L, int n)
{
    L = (LinkList) malloc (sizeof (LNode));
    L->next = NULL;
    for (i = n; i > 0; --i)
    {
    p = (LinkList) malloc (sizeof (LNode));
    scanf(“%d”,&e); p->data=e;
    p->next = L->next; L->next = p;
    }
    return L;
}
```

算法的时间复杂度为：O(n)

（2）在单链表中查找第i个元素（教材P61）

顺序存储可以随机存取,而链式存储不能，所以，要得到第i个元素，必须先通过计数(报数)的方法，找到它的位置。




```
While (p&&j<i) {p=p→ next; j++;}
```

算法：查找单链表中的某个元素

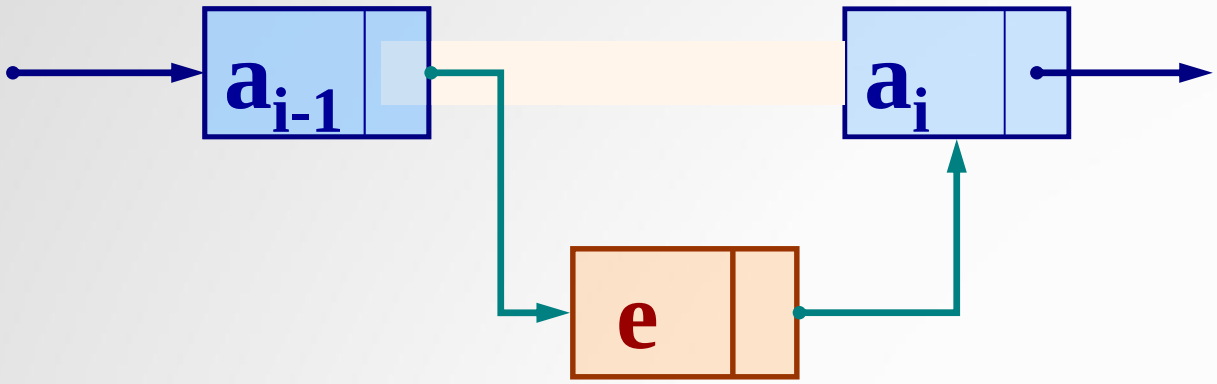
```
int GetElem_L(LinkList L, int i, int *e)
{  // L是带头结点的链表的头指针，以 e 返回第 i 个元素

 p = L->next;    j = 1;  // p指向第一个结点，j为计数器

while (p && j<i)  { p = p->next; j++;  }
    // 顺指针向后查找，直到 p 指向第 i 个元素
    // 或 p 为空
 if ( !p || j>i )
     return -1;                    //  第 i 个元素不存在
 *e = p->data;                    //  取得第 i 个元素
 return 1;
}
```

算法时间复杂度为：O(ListLength(L))=O(n)

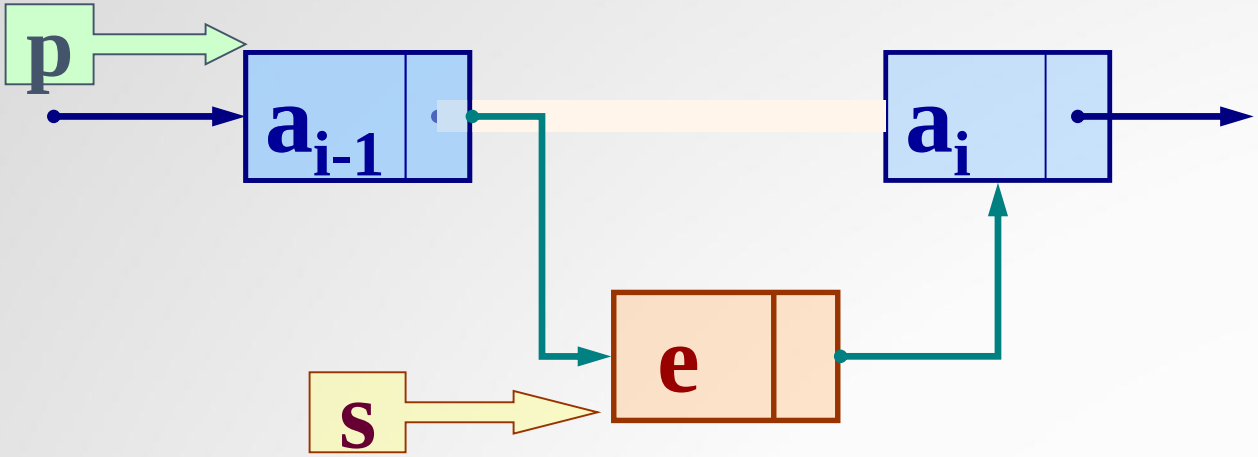
（3）在单链表第i个元素之前插入一个元素e（教材P63）

有序对 $\langle a_{i-1},\ a_i\rangle$

改变为 $\langle a_{i-1},\ e\rangle$ 和 $\langle e,\ a_i\rangle$



在单链表中第 i 个结点之前进行插入的基本操作为:找到线性表中第i-1个结点，然后修改其指向后继的指针。




```
s->next = p->next;

p->next = s;
```

算法：在单链表中插入一个元素

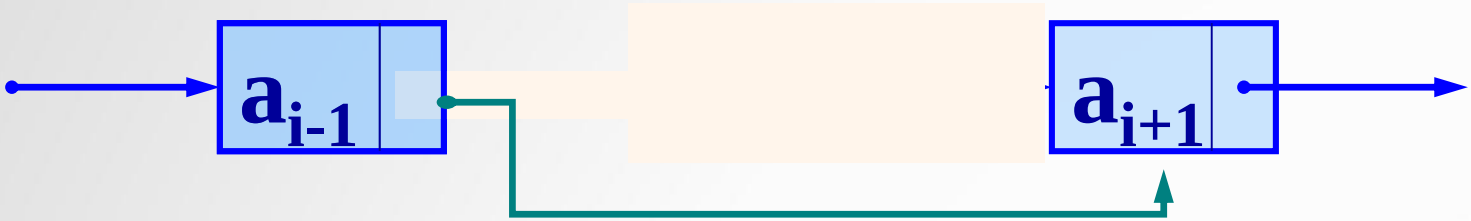
```
int ListInsert_L(LinkList L, int i, ElemType e)
{
    p = L;     j = 0;
    while (p && j < i-1)
          { p = p->next;  ++j; }    // 寻找第 i-1 个结点
    if (!p || j > i-1)
          return -1;          // i 大于表长或者小于1
    s = (LinkList) malloc ( sizeof (LNode));
                                    // 生成新结点
    s->data = e;
    s->next = p->next;          p->next = s; // 插入
    return 1;
}
```

算法的时间复杂度为：O(ListLength(L))=O(n)

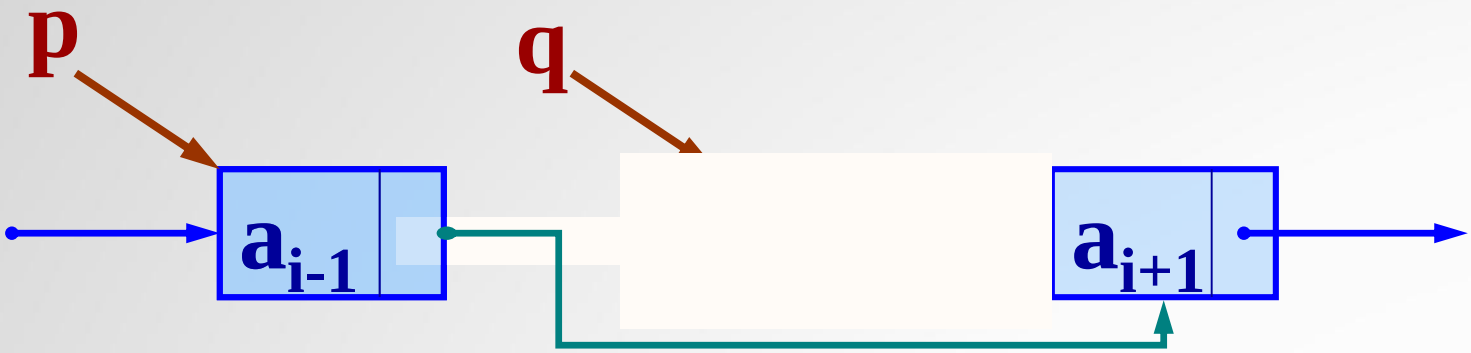
（4）删除单链表中的第i个元素（教材P65）

有序对$\langle a_{i-1}, a_i\rangle$和$\langle a_i, a_{i+1}\rangle$

改变为$\langle a_{i-1}, a_{i+1}\rangle$



在单链表中删除第 i 个结点的基本操作为:找到线性表中第i-1个结点，修改其指向后继的指针。




```
q = p->next;

p->next = q->next;

free(q);
```

算法：在单链表中删除某个元素

```
int ListDelete_L(LinkList L, int i, int  *e)
{
   p = L;     j = 0;
   while (p->next && j < i-1) {  p = p->next;    ++j; }
                        // 寻找第 i 个结点，并令 p 指向其前驱
   if  (!(p->next) || j > i-1)
       return -1;  // 删除位置不合理

   q = p->next;    p->next = q->next;  // 删除并释放结点
   *e = q->data;    free(q);
   return 1;

}
```

算法的时间复杂度为：O(ListLength(L))=O（n）

【Project 4】学生成绩管理系统

构建一个学生成绩管理系统，针对学生的信息（学号、姓名、性别、英语、数学、数据结构、简介）进行处理（查找、修改、插入、删除等），该系统要求采用链式存储实现。

（1）创建学生信息

（2）分别根据学号、姓名进行查找相关学生信息

（3）根据学号，修改相关学生的信息

（4）在信息表中插入一新学生的信息

（5）根据学号、姓名删除相关学生的信息

（6）输出信息表中的学生信息