本节主题：

栈的链式存储结构及其基本运算的实现

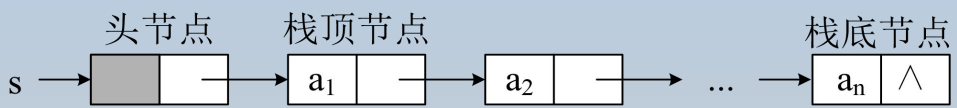
栈的链式存储结构

- 采用单链表：头节点后保存栈顶
- 存储结构定义

```
typedef struct linknode
{
    ElemType data;
    struct linknode *next;
} LiStack;
```

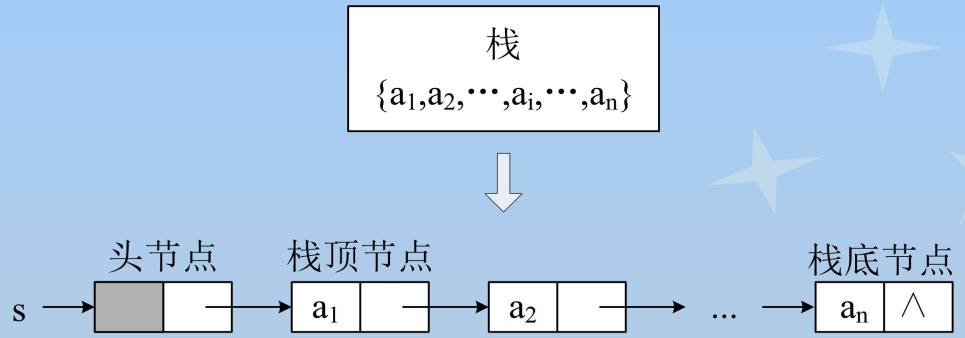
- 优点
  - 不存在栈满上溢的情况

栈
$\{a_1,a_2,\cdots,a_i,\cdots,a_n\}$



链栈的4要素

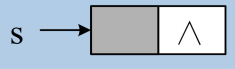
- 栈空条件
  - s->next=NULL
- 栈满条件
  - 不考虑
- 进栈e操作
  - 将包含e的节点插入到头节点之后
- 退栈操作
  - 取出头节点之后节点的元素并删除之



初始化栈initStack(&s)

- 建立一个空栈s

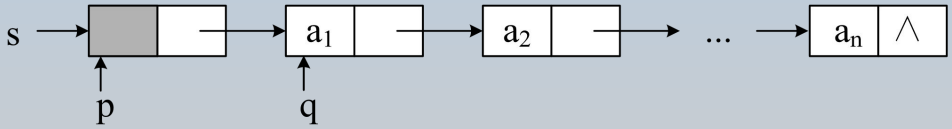
```
void InitStack(LiStack *&s)
{
    s=(LiStack *)malloc(sizeof(LiStack));
    s->next=NULL;
}
```



销毁栈ClearStack(&s)

- 释放栈s占用的全部存储空间

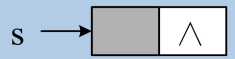
```
void DestroyStack(LiStack *&s)
{
    LiStack *p=s,*q=s->next;
    while (q!=NULL)
    {
        free(p);
        p=q;
        q=p->next;
    }
    free(p);
}
```



判断栈是否为空StackEmpty(s)

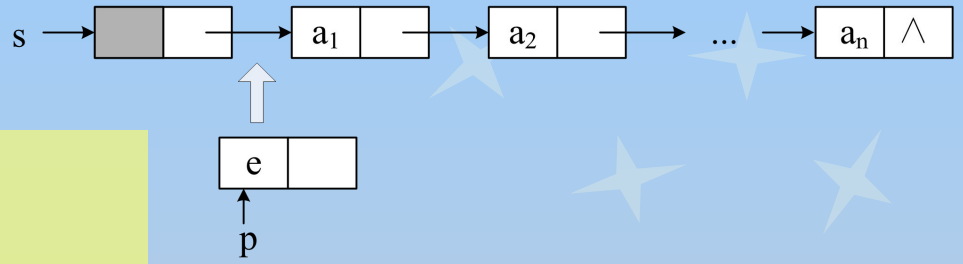
- 栈S为空的条件是s->next==NULL，即单链表中没有数据节点

```
bool StackEmpty(LiStack *s)
{
   return(s->next==NULL);
}
```



进栈Push(&s,e)

- 将新数据节点插入到头节点之后

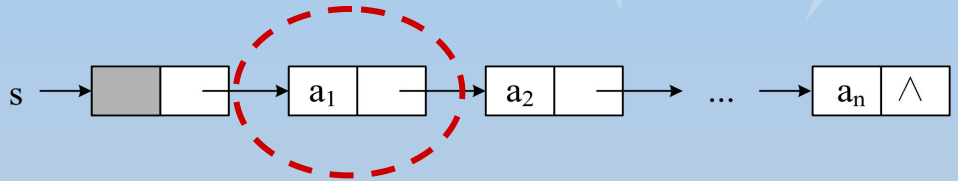



```
void Push(LiStack *&s,ElemType e)
{
    LiStack *p;
    p=(LiStack *)malloc(sizeof(LiStack));
    p->data=e;
    p->next=s->next;
    s->next=p;
}
```

出栈Pop(&s,&e)

- 在栈不为空的条件下，将头节点后继数据节点的数据域赋给e，然后将其删除。

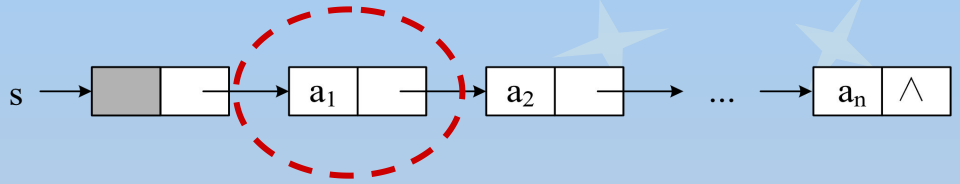
```
bool Pop(LiStack *&s,ElemType &e)
{
    LiStack *p;
    if (s->next==NULL)
        return false;
    p=s->next;
    e=p->data;
    s->next=p->next;
    free(p);
    return true;
}
```



取栈顶元素GetTop(s,e)

- 在栈不为空的条件下，将头节点后继数据节点的数据域赋给e。

```
bool GetTop(LiStack *s,ElemType &e)
{
   if (s->next==NULL)
      return false;
   e=s->next->data;
   return true;
}
```



应用示例：括号是否配对？ x+y*(a+b-(x-7)*y)

- 问题
  - 判断输入的表达式中括号是否配对（假设只含有左、右圆括号）
- 解题思路
  - 配对时返回true，否则返回false。
  - 设置一个栈St，扫描表达式exp：
    - 遇到左括号时进栈；
    - 遇到右括号时：若栈顶为左括号，则出栈，否则返回false。
  - 当表达式扫描完毕，栈为空时返回true；否则返回false。

```
bool Match(char exp[],int n)
{
   int i=0;
   char e;
   bool match=true;
   SqStack *st;
   InitStack(st);
   //扫描exp中的每一个字符
   if (!StackEmpty(st))
      match=false;
   DestroyStack(st);
   return match;
}
```

```
while (i<n && match)
{
   if (exp[i]=='(')
      Push(st,exp[i]);
   else if (exp[i]==')')
   {
      if (GetTop(st,e)==true)
      {
         if (e!='(')
            match=false;
         else
            Pop(st,e);
      }
      else
         match=false;
   }
   i++;
}
```