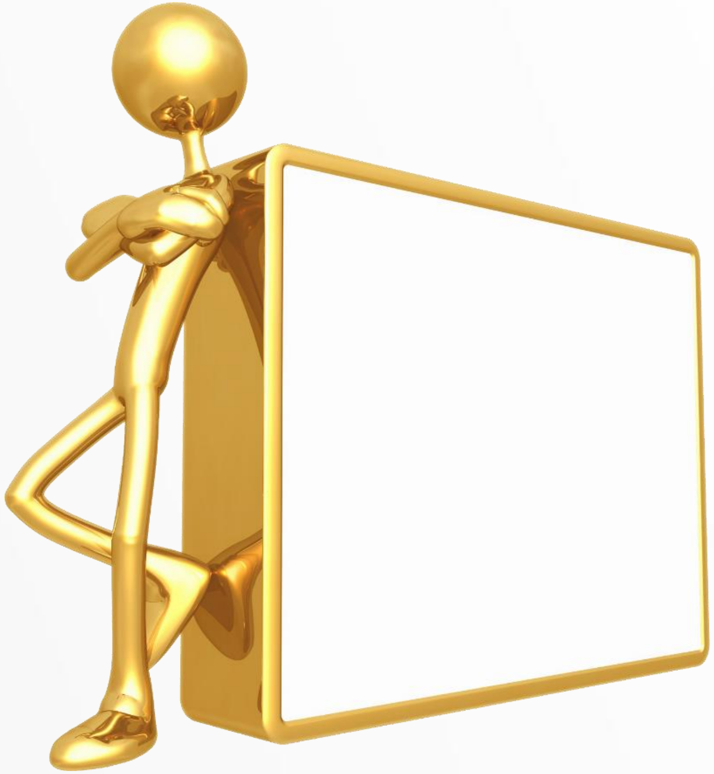
栈的链式存储结构

栈的链式存储结构：
(1) 单链表
(2)循环链表
(3) 双向链表
栈底：链表头部
栈顶：链表头部

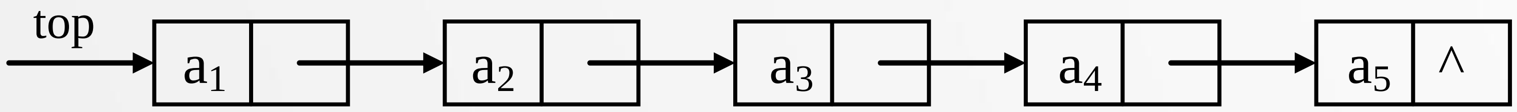


栈的链式存储

- 存储结构设计：

采用单链表的结点结构；

将单链表的首端作为栈顶；



- 类型定义

```
typedef struct node {     /*结点类型定义*/
    StackEntry entry;
    struct node *  next;
} StackNode,* StackNodePtr;
typedef struct stack {   /*链栈类型定义*/
    StackNodePtr top;        /* 指向栈顶的指针 */
} Stack,*StackPtr;
```

空栈时top=NULL

链栈**入栈**操作的实现

```
Status  Stack_Push(StackPtr s, StackEntry item){
     Status  outcome = success;
     StackNodePtr np = MakeNode(item);
    /* 申请结点空间，并装填结点域 */
    if (np = = NULL)
        outcome = overflow; /* 无法分配存储空间，相当于栈满上溢 */
    else {
        np->next = s->top;   /* 所申请到的结点插入在表头 */
        s->top  = np;
    }
   return outcome;
}
```

链栈出栈操作的实现

```
Status  Stack_Pop(StackPtr s, StackEntry *item){
     Status  outcome = success;
     if (Stack_Empty(s))
             outcome = underflow; /* 栈空则下溢 */
    else{
             StackNodePtr  np = s->top;      /* 删除栈顶元素 */
             s->top = np->next;
           *item = np->entry;
           free(np);
      }
      return outcome;
}
```

链栈**取栈顶元素**操作的实现

```
Status  Stack_Top(StackPtr s, StackEntry *item){
    Status  outcome = success;
    if (Stack_Empty(s))
          outcome = underflow; /* 栈空则下溢 */
    else
           *item = s->top->entry;
    return outcome;
}
```