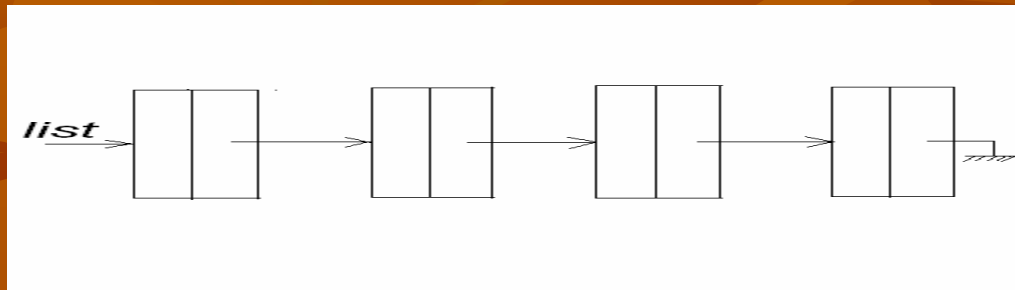
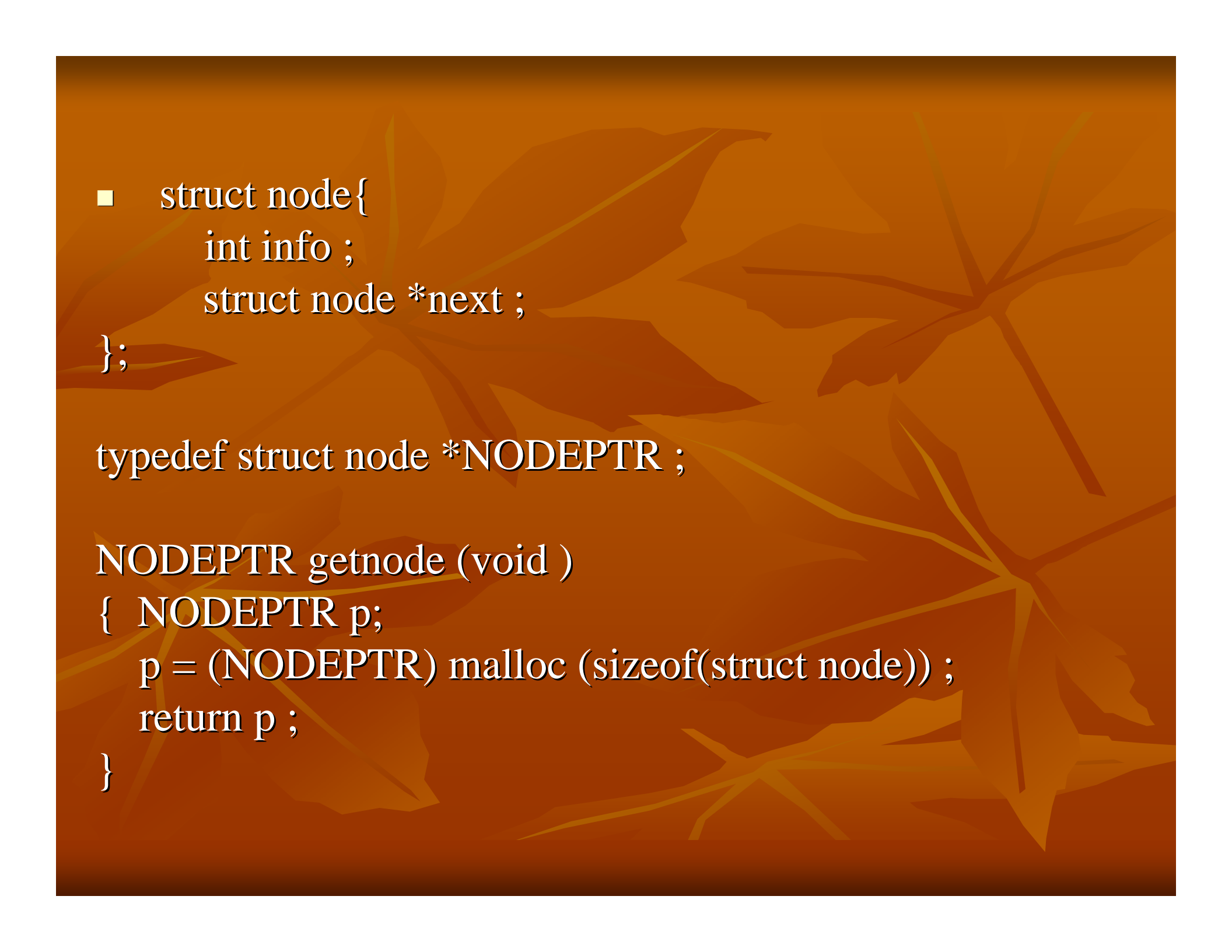


Linked Lists





- struct node{
 int info ;
 struct node *next ;
};

typedef struct node *NODEPTR ;

NODEPTR getnode (void)
{ NODEPTR p;
 p = (NODEPTR) malloc (sizeof(struct node)) ;
 return p ;
}

```
■ void freenode ( NODEPTR p )  
{  
    free(p);  
}
```

```
void insafter(NODEPTR p, int x)  
{  
    NODEPTR q ;  
    if(p==NULL){  
        printf(“void insertion \n”) ;  
        exit(1) ;  
    }  
    q=getnode() ;  
    q->info = x ;  
    q->next = p->next ;  
    p->next = q ;  
}/* end insertion */
```

```
■ void deleafter ( NODEPTR p, int *px )  
{  
    NODEPTR q ;  
    if ((p == NULL) || (p->next == NULL) ){  
        printf (“Void dletion \n”) ;  
        exit(1) ;  
    }  
    q = p->next ;  
    *px = q->info ;  
    p->next = q->next ;  
    freenode(q) ;  
}/* end deletion */
```

■ NODEPTR inserthead (NODEPTR list,
int x)

{

 NODEPTR q;

 q = getnode () ;

 q->info = x ;

 q->next = list ;

 list = q ;

 return (list) ;

}

```
■ NODEPTR deletehead ( NODEPTR list, int
    *px )
{
    NODEPTR q ;
    if(list == NULL ){
        printf("Void dletion \n") ;
        exit(1) ;
    }
    *px = list->info ;
    q = list ;
    list = list->next ;
    free(q) ;
    return list ;
}
```

- Queue (Dynamic Implementation)

- struct queue{

- NODEPTR front, rear ;

- };

- int empty (struct queue *pq)

- {

- return ((pq->front == NULL) ? TRUE

- : FALSE) ;

- }/*end empty */

```
■ void insert ( struct queue *pq, int x)
{
    NODEPTR p ;
    p = getnode() ;
    p->info = x ;
    p->next = NULL ;
    if(pq->rear == NULL)
        pq->front = p ;
    else
        (pq->rear)->next = p ;
    pq->rear = p ;
}/* end insert */
```



```
■ int remove ( struct queue *pq )  
{  
    NODEPTR p ;  
    int x;  
    if (empty(pq)){  
        printf (“queue underflow \n”);  
        exit(1);  
    }  
    p = pq->front;  
    x = p->info;  
    pq->front = p->next ;  
    if(pq->front == NULL)  
        pq->rear = NULL;  
    freenode(p) ;  
    return x ;  
}/* end remove */
```

```
■ NODEPTR search(NODEPTR list,int x)
{  NODEPTR p ;
   for ( p=list; p!=NULL; p=p->next)
       if (p->info == x)
           return p;
   /* x is not in the list */
   return NULL ;
}/* end search */
```