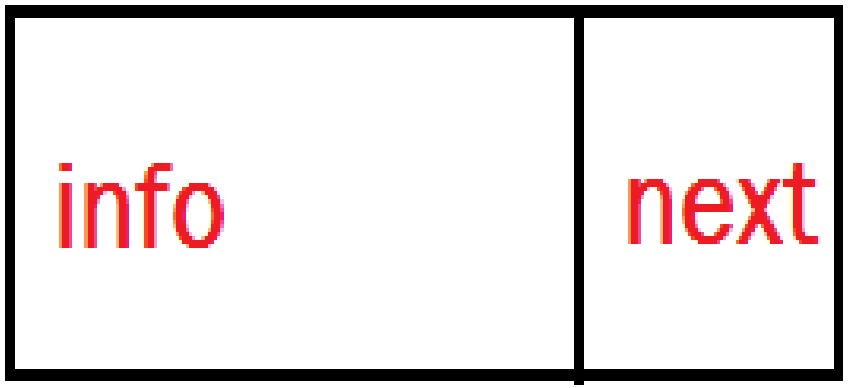
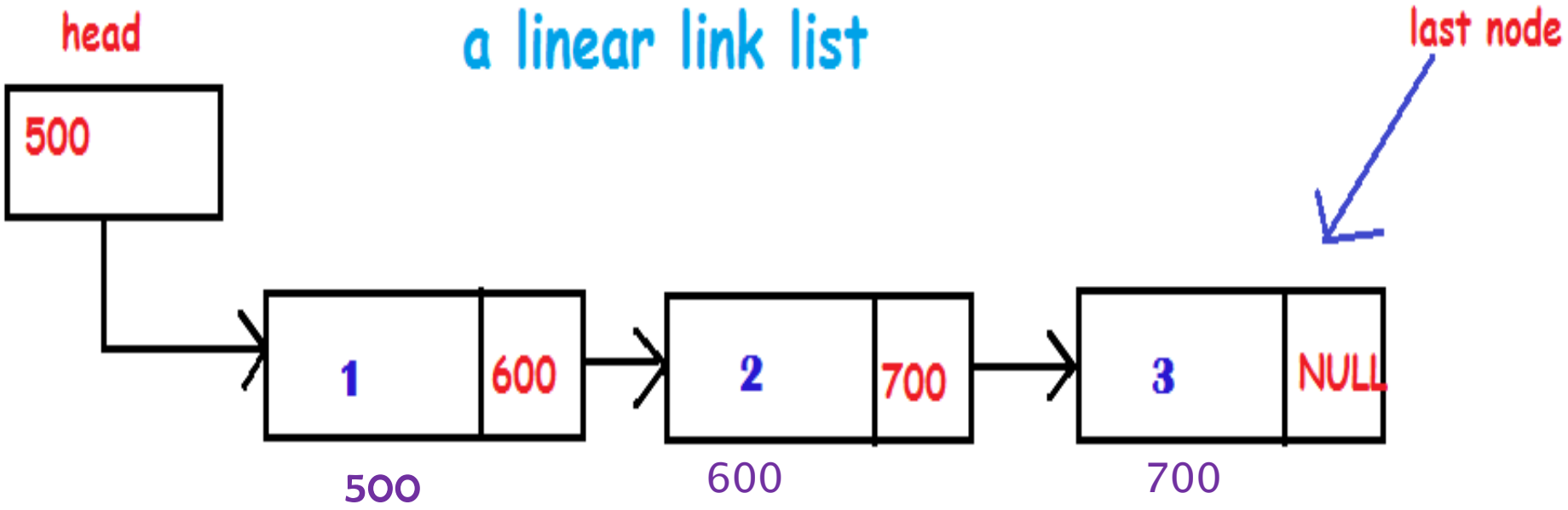
Link list

single node




head
a linear link list
last node
500
1
600
2
700
3
NULL
500
600
700

Singly Link List

A singly link list is a dynamic data structure . It may grow or shrink , which depends on the operation you perform.

In C , linked list is created using structures, pointers and dynamic memory allocation function malloc . We consider head as an external pointer . This helps in creating and accessing other nodes in the linked list.

Linked list is created in C using structures

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

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

node *ptr;
ptr=(node*)malloc(sizeof(node));
```

Inserting nodes in linked list

To insert an element following 3 things should be done

1. Allocating a node
2. Assigning the data
3. Adjusting the pointers

Now inserting a new node in the linked list has the following instances

1. Insertion at the beginning of the list
2. Insertion at the end of the list
3. Insertion at the specified position within the list

Function to insert node at the beginning

```
void insert_begin()
{
node *ptr;
ptr=(node*)malloc(sizeof(node));
ptr->info=item;
if(head==NULL)
ptr->next=NULL;
else
ptr->next=head;
head=ptr;
}
```

Function to insert node at specific location

```
void insert_spe()
{
node *ptr;
int pos;
ptr=(node*)malloc(sizeof(node));
ptr->info=item;
ptr->next=NULL;
if(head==NULL)
{
ptr->next=NULL;
head=ptr;
}
else
{ printf("enter the position");
scanf("%d",&pos);
node *temp=head;
int count=1;
while(count<pos-1)
```

```
{
temp=temp->next;
count++;
}
ptr->next=temp->next;
temp->next=ptr;
}
}
```

Function to insert node at end position

```
void insert_at_end()
{
node *ptr,*temp;
ptr=(node*)malloc(sizeof(node));
ptr->info=item;
ptr->next=NULL;
if(head==NULL)
head=ptr;
else
{
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=ptr;
}
}
```

Deleting nodes from the link list

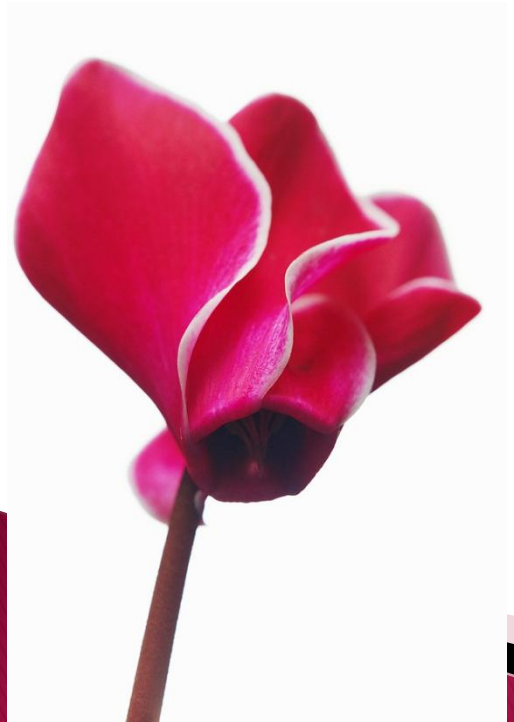
Deleting a node from the linked list has 3 instances just like insertion

1. Deleting the first node of the link list
2. Deleting the last node of the link list
3. Deleting the specified node within the link list

Function to delete the first node

```
void delete_at_begin()
{
node *temp;
int item;
            if(head==NULL)
            {
            printf("list is empty");
            }
            else
            {
            temp=head;
            if(temp->next==NULL)
            {
             item=temp->info;
             head=NULL;
            }
```

```
 else
        {
         item=temp->info;
         head=temp->next;
        }
}
printf("\n Node deleted is %d", item);
free(temp);
```



Function to delete the last node

```
void delete_at_end ( )
{
node *temp,*temp1;
int item;
        if(head==NULL)
        {
        printf("list is empty");
        }
        else
        {
        temp=head;
        if(temp->next==NULL)
        {
          item=temp->info;
          head=NULL;
        }
```

```
else
        {
        temp1=head->next;
        while(temp1->next!=NULL)
        {
        temp=temp->next;
        temp1=temp1->next;
        }
        temp->next=NULL;
        item=temp1->info;
        }
        }

printf("\n node deleted is %d", item);
free(temp1);
}
```

Function to delete node at specific location

```
void delete_at_specific ( )
{
node *temp,*temp1;
int item , pos , count;
                    if(head==NULL)
                    {
                    printf("list is empty");
                    }
                    else
                    {
                    temp=head,temp1=head->next;
                    if(temp->next==NULL)
                    {
                    item=temp->info;
                    head=NULL;
                    }
```

```
else
                    {
                    printf("enter position of node to be deleted\n");
                    scanf("%d",&pos);
                    int count=1;
                    while(count<pos-1)
                    {
                    temp=temp->next;
                    temp1=temp1->next;
                    count++;
                    }
                    temp->next=temp1->next;
                    item=temp1->info;
                    }
                    }
printf("\n node deleted is %d",item);
free(temp1);
}
```