Data Structures

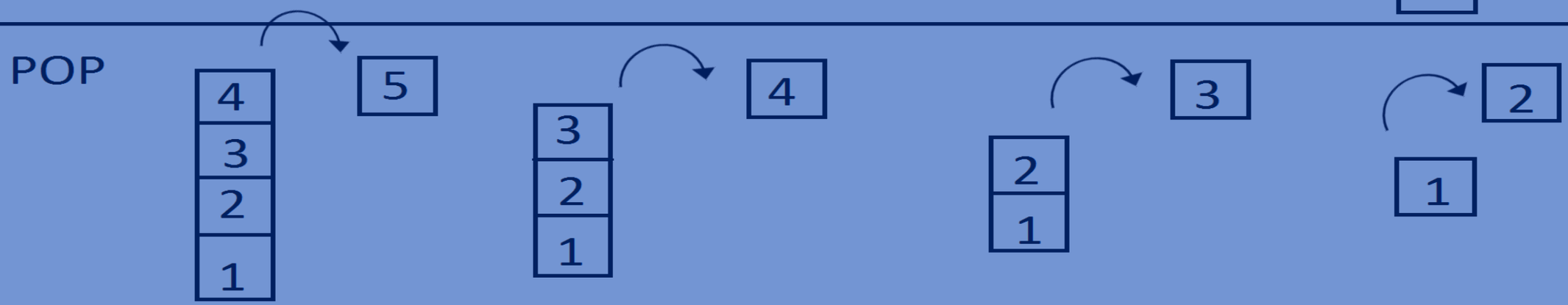
Implementing Stack and Queue using linked List

STACK:

- A stack is a container of objects that are inserted and removed according

To last-in first-out principle.

- Supports two operations push and pop.
- Push inserts the new elements at the top of the existing list.
- Pop removes the top most element from the list(most recently added item).



Implementation:

- **Function for pushing elements into stack.**

```
lin_list *push(lin_list *head,int data){
    lin_list *temp=(lin_list*)malloc(sizeof(lin_list));
    temp->data=data;
    temp->next=NULL;
    temp->next=head;
    head=temp;
    return head;
}
```

- **Function for poping of elements from stack.**

```
lin_list *pop(lin_list *head){
    lin_list *temp=head;
    head=head->next;
    printf("Deleted the node containing data %d\n",temp->data);
    free(temp);
    return head;
}
```

Whole program:

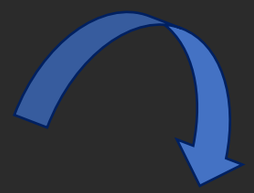
```c
#include<stdio.h>
#include<stdlib.h>
//cresting a node.
typedef struct lin_list{
    int data;
    struct lin_list *next;
}lin_list;

lin_list *push(lin_list *head,int data){
    lin_list *temp=(lin_list*)malloc(sizeof(lin_list));
    temp->data=data;
    temp->next=NULL;
    temp->next=head;
    head=temp;
    return head;
}

lin_list *pop(lin_list *head){
    lin_list *temp=head;
    head=head->next;
```

```
    printf("Deleted the node containing data %d\n",temp->data);
    free(temp);
    return head;
}

void PrintElements(lin_list *head){
    while(head!=NULL){
        printf("%d ",head->data);
        head=head->next;
    }
}
int main(){
    lin_list *head=NULL;
    head=push(head,1);
    head=push(head,2);
    head=push(head,3);
    head=push(head,4);
    PrintElements(head);printf("\n");
    head=pop(head);
    PrintElements(head);printf("\n");
    head=pop(head);
    PrintElements(head);
    return 0;
}
```

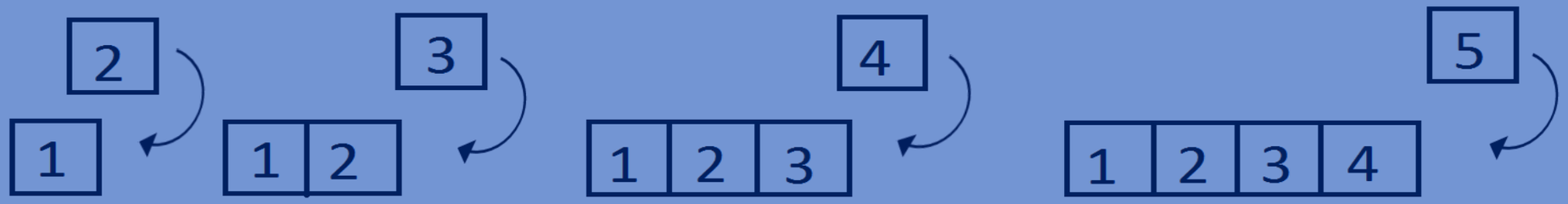


```
4 3 2 1
Deleted the node containing data 4
3 2 1
Deleted the node containing data 3
2 1
```

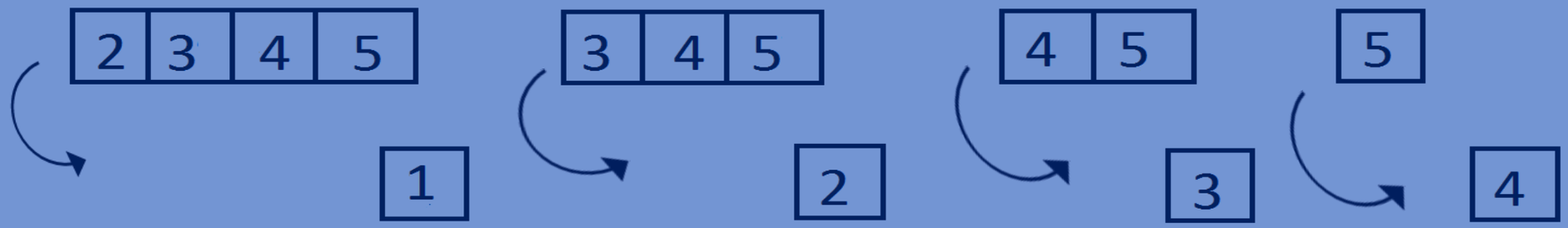
Queue:

- A queue is a container of objects that are inserted and removed according to the first-in first-out principle.
- Supports operations enqueue and dequeue.
- enqueue Inserts the new elements at the end of the list.
- dequeue removes the top most element from the list.

enqueue



dequeue



Implementation:

- **Function for enqueue.**

```
lin_list *enqueue(lin_list *head,int data){
    lin_list *newnode=(lin_list*)malloc(sizeof(lin_list));
    newnode->data=data;
    newnode->next=NULL;
    lin_list *temp=head;
    if(head==NULL){
        head=newnode;
    }
    else {
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newnode;
    }
    return head;
}
```

➢ Function for dequeue.

```
lin_list *dequeue(lin_list *head){
    lin_list *temp=head;
    head=head->next;
    printf("deleted the node containing data %d",temp->data);printf("\n");
    free(temp);
    return head;
}
```

Whole program:

```
#include<stdio.h>
#include<stdlib.h>
//creating a node.
typedef struct lin_list{
    int data;
    struct lin_list *next;
}lin_list;

lin_list *enqueue(lin_list *head,int data){
    lin_list *newnode=(lin_list*)malloc(sizeof(lin_list));
    newnode->data=data;
```

```
    newnode->next=NULL;
    lin_list *temp=head;
    if(head==NULL){
        head=newnode;
    }
    else {
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newnode;
    }
    return head;
}

lin_list *dequeue(lin_list *head){
    lin_list *temp=head;
    head=head->next;
    printf("deleted the node containing data %d\n",temp->data);
    free(temp);
    return head;
}

void PrintElements(lin_list *head){
    while(head!=NULL){
        printf("%d ",head->data);
        head=head->next;
    }
```

```
}
int main(){
    lin_list *head=NULL;
    head=enqueue(head,1);
    head=enqueue(head,2);
    head=enqueue(head,3);
    head=enqueue(head,4);
    PrintElements(head);printf("\n");
    head=dequeue(head);
    PrintElements(head);printf("\n");
    head=dequeue(head);
    PrintElements(head);
    return 0;
}
```

Output:

1 2 3 4

deleted the node containing data 1

2 3 4

deleted the node containing data 2

3 4