1. Array Implementation of Stack

#include <stdio.h>

int stack[100],i,j,choice=0,n,top=-1;

void push();

void pop();

void show();

void main ()

{

printf("Enter the number of elements in the stack ");

scanf("%d",&n);

printf("\*\*\*\*\*\*\*\*\*Stack operations using array\*\*\*\*\*\*\*\*\*");

printf("\n----------------------------------------------\n");

while(choice != 4)

{

printf("Chose one from the below options...\n");

printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");

printf("\n Enter your choice \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

show();

break;

}

case 4:

{

printf("Exiting....");

break;

}

default:

{

printf("Please Enter valid choice ");

}

};

}

}

void push () {

int val;

if (top == n )

printf("\n Overflow");

else {

printf("Enter the value?");

scanf("%d",&val);

top = top +1;

stack[top] = val; } }

void pop () {

if(top == -1)

printf("Underflow");

else

top = top -1; }

void show() {

for (i=top;i>=0;i--) {

printf("%d\n",stack[i]); }

if(top == -1) {

printf("Stack is empty");}}

OUTPUT:

Enter the number of elements in the stack 4

\*\*\*\*\*\*\*\*\*Stack operations using array\*\*\*\*\*\*\*\*\*

----------------------------------------------

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

1

Enter the value?50

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

2

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

3

Stack is emptyChose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

4

Exiting....

2. Linked List Implementation of Stack

#include <stdio.h>

#include <stdlib.h>

void push();

void pop();

void display();

struct node

{

int val;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice=0;

printf("\n\*\*\*\*\*\*\*\*\*Stack operations using linked list\*\*\*\*\*\*\*\*\*\n");

printf("\n----------------------------------------------\n");

while(choice != 4)

{

printf("\n\nChose one from the below options...\n");

printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");

printf("\n Enter your choice \n");

scanf("%d",&choice);

switch(choice) {

case 1: {

push();

break; }

case 2: {

pop();

break; }

case 3: {

display();

break; }

case 4: {

printf("Exiting....");

break; }

default: {

printf("Please Enter valid choice "); }

}; }}

void push ()

{

int val;

struct node \*ptr = (struct node\*)malloc(

sizeof(struct node));

if(ptr == NULL)

{

printf("not able to push the element");

}

else

{

printf("Enter the value");

scanf("%d",&val);

if(head==NULL)

{

ptr->val = val;

ptr -> next = NULL;

head=ptr;

}

else

{

ptr->val = val;

ptr->next = head;

head=ptr;

}

printf("Item pushed");

}

}

void pop()

{

int item;

struct node \*ptr;

if (head == NULL)

{

printf("Underflow");

}

else

{

item = head->val;

ptr = head;

head = head->next;

free(ptr);

printf("Item popped");

}

}

void display()

{

int i;

struct node \*ptr;

ptr=head;

if(ptr == NULL)

{

printf("Stack is empty\n");

}

else

{

printf("Printing Stack elements \n");

while(ptr!=NULL)

{

printf("%d\n",ptr->val);

ptr = ptr->next;

}

}

}

OUTPUT:

\*\*\*\*\*\*\*\*\*Stack operations using linked list\*\*\*\*\*\*\*\*\*

----------------------------------------------

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

3

Stack is empty

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

1

Enter the value50

Item pushed

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

1

Enter the value60

Item pushed

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

2

Item popped

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

4

Exiting....

3. Implementation of Infix to Postfix conversion and Evaluation of Postfix

Expression.

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define SIZE 30

char stack[SIZE];

int top=0;

char infix[SIZE];

char postfix[SIZE];

int p=0; void in2po();

void push(char);

void pop(char);

int main()

{

printf("infix: ");

gets(infix);

infix[strlen(infix)]='.';

in2po(); printf("postfix: ");

puts(postfix); getch();

}

void push(char c)

{

if(top!=SIZE) {

stack[top]=c;

top++;

} else {

printf("Stack Overflow\n");

getch();

exit(EXIT\_FAILURE);

}

}

void pop(char c)

{

if(top<0) {

printf("Stack empty\n");

getch(); exit(EXIT\_FAILURE);

} else {

switch(c) {

case '^': while( top>0 && stack[top-1]=='^' ) {

postfix[p]=stack[top-1];

top--; p++;

} break;

case '/':

case '\*':

while( top>0 && (stack[top-1]=='^' || stack[top-1]=='/') ) {

postfix[p]=stack[top-1];

top--;

p++;

} break;

case ')': while( top>0 && stack[top-1]!='(' ) {

postfix[p]=stack[top-1];

top--; p++;

} top--;

break;

case '.': while( top>0 ) {

postfix[p]=stack[top-1];

top--; p++;

} break;

}

}

}

void in2po() {

int i=0;

char c;

while(i<strlen(infix))

{

c = infix[i];

switch(c) {

case '+':

case '-':

case '(' : push(c); break;

case '.': case ')': pop(c);

break;

case '/':

case '^':

case '\*':

pop(c);

push(c);

break;

default: postfix[p]=c;

p++;

}

i++;

}

}

OUTPUT:

infix: a+b+c

postfix: abc++

4. Array Implementation of Queue

#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

void insert();

void delete();

void display();

int front = -1, rear = -1;

int queue[maxsize];

void main ()

{

int choice;

while(choice != 4)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n=================================================================\n");

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

printf("\nEnter your choice ?");

scanf("%d",&choice);

switch(choice) {

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nEnter valid choice??\n"); } } }

void insert()

{

int item;

printf("\nEnter the element\n");

scanf("\n%d",&item);

if(rear == maxsize-1)

{

printf("\nOVERFLOW\n");

return;

}

if(front == -1 && rear == -1)

{

front = 0;

rear = 0;

}

else

{

rear = rear+1;

}

queue[rear] = item;

printf("\nValue inserted ");

}

void delete() {

int item;

if (front == -1 || front > rear) {

printf("\nUNDERFLOW\n");

return; }

else {

item = queue[front];

if(front == rear) {

front = -1;

rear = -1 ; }

else {

front = front + 1; }

printf("\nvalue deleted "); } }

void display()

{

int i;

if(rear == -1)

{

printf("\nEmpty queue\n");

}

else

{ printf("\nprinting values .....\n");

for(i=front;i<=rear;i++)

{

printf("\n%d\n",queue[i]);

}

}

}

OUTPUT:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?1

Enter the element 50

Value inserted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?1

Enter the element

60

Value inserted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?30

Enter valid choice??

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?3

printing values .....

50

60

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?1

Enter the element

70

Value inserted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?2

value deleted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?1

Enter the element

80

Value inserted

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?3

printing values .....

50

60

70

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?4

5. Linked List Implementation of Queue

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*front;

struct node \*rear;

void insert();

void delete();

void display();

void main ()

{

int choice;

while(choice != 4)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n=================================================================\n");

printf("\n1.insert an element\n2.Delete an element\n3.Display thequeue\n4.Exit\n");

printf("\nEnter your choice ?");

scanf("%d",& choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nEnter valid choice??\n");

}

}

}

void insert()

{

struct node \*ptr;

int item;

ptr = (struct node \*) malloc (sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

return;

}

else

{

printf("\nEnter value?\n");

scanf("%d",&item);

ptr -> data = item;

if(front == NULL)

{

front = ptr;

rear = ptr;

front -> next = NULL;

rear -> next = NULL;

}

else

{

rear -> next = ptr;

rear = ptr;

rear->next = NULL;

}

}

}

void delete ()

{

struct node \*ptr;

if(front == NULL)

{

printf("\nUNDERFLOW\n");

return;

}

else

{

ptr = front;

front = front -> next;

free(ptr);

}

}

void display()

{

struct node \*ptr;

ptr = front;

if(front == NULL)

{

printf("\nEmpty queue\n");

}

else

{

printf("\nprinting values .....\n");

while(ptr != NULL)

{

printf("\n%d\n",ptr -> data);

ptr = ptr -> next;

}

}

}

OUTPUT:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?1

Enter value?

15

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?1

Enter value?

11

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?1

Enter value?

23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?1

Enter value?

34

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?1

Enter value?

34

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?3

printing values .....

23

34

34

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?1

Enter value?

22

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?3

printing values .....

34

34

22

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=================================================================

1.insert an element

2.Delete an element

3.Display thequeue

4.Exit

Enter your choice ?4

6. Array Implementation of Circular Queue

#include <stdio.h>

# define max 6

int queue[max]; // array declaration

int front=-1;

int rear=-1;

void enqueue(int element)

{

if(front==-1 && rear==-1)

// condition to check queue is empty

{

front=0;

rear=0;

queue[rear]=element;

}

else if((rear+1)%max==front)

// condition to check queue is full

{

printf("Queue is overflow..");

}

else

{

rear=(rear+1)%max;

// rear is incremented

queue[rear]=element;

// assigning a value to the queue at the rear position.

}

}

int dequeue()

{

if((front==-1) && (rear==-1))

// condition to check queue is empty

{

printf("\nQueue is underflow..");

}

else if(front==rear)

{

printf("\nThe dequeued element is %d", queue[front]);

front=-1;

rear=-1;

}

else

{

printf("\nThe dequeued element is %d", queue[front]);

front=(front+1)%max;

}

void display()

{

int i=front;

if(front==-1 && rear==-1)

{

printf("\n Queue is empty..");

}

else

{

printf("\nElements in a Queue are :");

while(i<=rear)

{

printf("%d,", queue[i]);

i=(i+1)%max;

}

}

}

int main()

{

int choice=1,x; // variables declaration

while(choice<4 && choice!=0) // while loop

{

printf("\n Press 1: Insert an element");

printf("\nPress 2: Delete an element");

printf("\nPress 3: Display the element");

printf("\nEnter your choice");

scanf("%d", &choice);

switch(choice)

{

case 1:

printf("Enter the element which is to be inserted");

scanf("%d", &x);

enqueue(x);

break;

case 2:

dequeue();

break;

case 3:

display();

}

}

return 0;

}

OUTPUT:

Press 1: Insert an element

PRess 2: Delete an element

Press 3: Display an element

Enter your choice

1

Enter the element which is to be inserted

10

Press 1: Insert an element

PRess 2: Delete an element

Press 3: Display an element

Enter your choice

1

Enter the element which is to be inserted

20

Press 1: Insert an element

PRess 2: Delete an element

Press 3: Display an element

Enter your choice

1

Enter the element which is to be inserted

30

Press 1: Insert an element

PRess 2: Delete an element

Press 3: Display an element

Enter your choice

3

Elements ina Queue are:10, 20, 30,

Press 1: Insert an element

PRess 2: Delete an element

Press 3: Display an element

Enter your choice

2

The dequeued element is 10

7. Linked List Implementation of Circular Queue

#include <stdio.h>

// Declaration of struct type node

struct node

{

int data;

struct node \*next;

};

struct node \*front=-1;

struct node \*rear=-1;

// function to insert the element in the Queue

void enqueue(int x)

{

struct node \*newnode; // declaration of pointer of struct node type.

newnode=(struct node \*)malloc(sizeof(struct node));

// allocating the memory to the newnode

newnode->data=x;

newnode->next=0;

if(rear==-1) // checking whether the Queue is empty or not.

{

front=rear=newnode;

rear->next=front;

}

else

{

rear->next=newnode;

rear=newnode;

rear->next=front;

}

}

// function to delete the element from the queue

void dequeue()

{

struct node \*temp;

// declaration of pointer of node type

temp=front;

if((front==-1)&&(rear==-1))

// checking whether the queue

// is empty or not

{

printf("\nQueue is empty");

}

else if(front==rear)

// checking whether the single element is left in the queue

{

front=rear=-1;

free(temp);

}

else

{

front=front->next;

rear->next=front;

free(temp);

}

}

// function to get the front of the queue

int peek()

{

if((front==-1) &&(rear==-1))

{

printf("\nQueue is empty");

}

else

{

printf("\nThe front element is %d", front->data);

}

}

// function to display all the elements of the queue

void display()

{

struct node \*temp;

temp=front;

printf("\n The elements in a Queue are : ");

if((front==-1) && (rear==-1))

{

printf("Queue is empty");

}

else

{

while(temp->next!=front)

{

printf("%d,", temp->data);

temp=temp->next;

}

printf("%d", temp->data);

}

}

void main()

{

enqueue(34);

enqueue(10);

enqueue(23);

display();

dequeue();

peek();

}

OUPUT:

The elements in a Queue are : 34,10,23

The front element is 10