**Object:-** Write a program to create, enqueue elements, dequeue elements and traverse elements of queue using switch.

**Description:-**

1. **Queue-** Likestack , Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of queue of consumers for a resource where the consumers that came first is served first.

The difference between stacks and queues is in removing. In a stack we remove the item the most recently added, in a queue, we remove the item the least recently added.

**Operations on Queue:-** Mainly the following four operations are performed on queue:

**Enqueue:** Add an item to the queue. If the queue is full, then it is said to be an overflow condition.

**Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an underflow condition.

**Front:** Get the front item from queue.

**Rear:** Get the last item from queue.

**Applications of Queue:**- Queue is used when things don’t have to be processed immediately, but have to be processed in First In First Out order like Breadth first search. This property of queue makes it also useful in following kind of scenarios:

1. When a resource is shared among multiple consumers. Examples include CPU scheduling.
2. When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, Pipes, File IO, etc.
3. **Switch statement:-**

Switch case statements are a substitute for long if statements that compare a variable to several integral values

* The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.
* Switch is a control statement that allows a value to change control of execution.

**Syntax:**

Switch (n)

{

case 1: //code to be executed if n=1;

break;

case 2: //code to be executed if n=2;

break;

default: //code to be executed if n doesn’t match any cases

}

Program:-

#include<stdio.h> //header file

#include<conio.h> //header file

int queue[5]; //array declaration

int maxsize=5,front=-1,rear=-1; // declaration

void enqueue(); //function declaration

void dequeue(); //function declaration

void traverse(); //function declaration

int main()

{

int choice,i; //declaration

clrscr();

printf("queue operation\n");

printf("1. For enqueue\n 2. For dequeue\n 3.For traverse\n 4. For exit");

do //do-while loop

{

printf("\n enter the choice:");

scanf("%d",&choice);

switch(choice) //switch statement

{

case 1: enqueue(); //function call

break;

case 2: dequeue(); //function call

break;

case 3: traverse(); //function call madhav

break;

case 4: exit();

break;

default: printf("you entered wrong choice");

}

}

while(choice!=4);

return 0;

}

void enqueue() //function definition

{

int element; //declaration

if (rear==maxsize-1) //if statement

{

printf("queue is overflow");

}

printf("enter the element:");

scanf("%d",&element);

if(front==-1) //if statement

{

front=front+1; //increment

rear=rear+1; //increment

queue[rear]=element;

}

else //else statement

{

rear=rear+1; //increment

queue[rear]=element;

}

}

void dequeue() //function definition

{

if(front==-1) //if statement

{

printf("queue is underflow");

}

printf("\n deleted element is: %d",queue[front]);

if(front==rear) //if statement

{

queue[front]='/0';

front=-1;

rear=-1;

}

else //else statement

{

queue[front]='/0';

front=front+1; //increment

}

}

void traverse() //function call

{

int i;

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

{

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

}

}

