|  |  |
| --- | --- |
|  | #define SIZE 5 /\* Size of Circular Queue \*/  int CQ[SIZE],f=-1,r=-1; */\* Global declarations \*/*    void CQinsert(int elem)  { */\* Function for Insert operation \*/*  if( CQfull()) printf("**\n\n** Overflow!!!!**\n\n**");  else  {  if(f==-1)f=0;  r=(r+1) % SIZE;  CQ[r]=elem;  }  }  int CQdelete()  { */\* Function for Delete operation \*/*  int elem;  if(CQempty()){ printf("**\n\n**Underflow!!!!**\n\n**");  return(-1); }  else  {  elem=CQ[f];  if(f==r){ f=-1; r=-1;} */\* Q has only one element ? \*/*  else  f=(f+1) % SIZE;  return(elem);  }  }  int CQfull()  { */\* Function to Check Circular Queue Full \*/*  */\* proverava se pre povećanja r\*/*  if( (f==r+1) || (f == 0 && r== SIZE-1)) return 1;  return 0;  }    int CQempty()  { */\* Function to Check Circular Queue Empty \*/*  if(f== -1) return 1;  return 0;  }    void display()  { */\* Function to display status of Circular Queue \*/*  int i;  if(CQempty()) printf(" **\n** Empty Queue**\n**");  else  {  printf("Front[%d]->",f);  for(i=f;i!=r;i=(i+1)%SIZE)  printf("%d ",CQ[i]);  printf("%d ",CQ[i]);  printf("<-[%d]Rear",r);  }  }    void main()  { */\* Main Program \*/*  int opn,elem;  do  {  clrscr();  printf("**\n** ### Circular Queue Operations ### **\n\n**");  printf("**\n** Press 1-Insert, 2-Delete,3-Display,4-Exit**\n**");  printf("**\n** Your option ? ");  scanf("%d",&opn);  switch(opn)  {  case 1: printf("**\n\n**Read the element to be Inserted ?");  scanf("%d",&elem);  CQinsert(elem); **break**;  case 2: elem=CQdelete();  if( elem != -1)  printf("**\n\n**Deleted Element is %d **\n**",elem);  **break**;  case 3: printf("**\n\n**Status of Circular Queue**\n\n**");  display(); **break**;  case 4: printf("**\n\n** Terminating **\n\n**"); **break**;  default: printf("**\n\n**Invalid Option !!! Try Again !! **\n\n**");  **break**;  }  printf("**\n\n\n\n** Press a Key to Continue . . . ");  getch();  }while(opn != 4);  } |