|  |
| --- |
|  |
| // | C program to implement circular queue using SLL |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  |  |
|  | struct node |
|  | { |
|  | int info; |
|  | struct node \*link; |
|  | } \*rear = NULL; |
|  |  |
|  | void insert(int item); |
|  | int del(); |
|  | void display(); |
|  | int isEmpty(); |
|  | int peek(); |
|  |  |
|  | int main() |
|  | { |
|  | int choice, item; |
|  | while (1) |
|  | { |
|  | printf("\n1.Insert\n"); |
|  | printf("2.Delete\n"); |
|  | printf("3.Peek\n"); |
|  | printf("4.Display\n"); |
|  | printf("5.Quit\n"); |
|  | printf("\nEnter your choice : "); |
|  | scanf("%d", &choice); |
|  |  |
|  | switch (choice) |
|  | { |
|  | case 1: |
|  | printf("\nEnter the element for insertion : "); |
|  | scanf("%d", &item); |
|  | insert(item); |
|  | break; |
|  | case 2: |
|  | printf("\nDeleted element is %d\n", del()); |
|  | break; |
|  | case 3: |
|  | printf("\nItem at the front of queue is %d\n", peek()); |
|  | break; |
|  | case 4: |
|  | display(); |
|  | break; |
|  | case 5: |
|  | exit(1); |
|  | default: |
|  | printf("\nWrong choice\n"); |
|  | } /\*End of switch\*/ |
|  | } /\*End of while\*/ |
|  | } /\*End of main()\*/ |
|  |  |
|  | void insert(int item) |
|  | { |
|  | struct node \*tmp; |
|  | tmp = (struct node \*)malloc(sizeof(struct node)); |
|  | tmp->info = item; |
|  | if (tmp == NULL) |
|  | { |
|  | printf("\nMemory not available\n"); |
|  | return; |
|  | } |
|  |  |
|  | if (isEmpty()) /\*If queue is empty \*/ |
|  | { |
|  | rear = tmp; |
|  | tmp->link = rear; |
|  | } |
|  | else |
|  | { |
|  | tmp->link = rear->link; |
|  | rear->link = tmp; |
|  | rear = tmp; |
|  | } |
|  | } /\*End of insert()\*/ |
|  |  |
|  | int del() |
|  | { |
|  | int item; |
|  | struct node \*tmp; |
|  | if (isEmpty()) |
|  | { |
|  | printf("\nQueue underflow\n"); |
|  | exit(1); |
|  | } |
|  | if (rear->link == rear) /\*If only one element\*/ |
|  | { |
|  | tmp = rear; |
|  | rear = NULL; |
|  | } |
|  | else |
|  | { |
|  | tmp = rear->link; |
|  | rear->link = rear->link->link; |
|  | } |
|  | item = tmp->info; |
|  | free(tmp); |
|  | return item; |
|  | } /\*End of del()\*/ |
|  |  |
|  | int peek() |
|  | { |
|  | if (isEmpty()) |
|  | { |
|  | printf("\nQueue underflow\n"); |
|  | exit(1); |
|  | } |
|  | return rear->link->info; |
|  | } /\* End of peek() \*/ |
|  |  |
|  | int isEmpty() |
|  | { |
|  | if (rear == NULL) |
|  | return 1; |
|  | else |
|  | return 0; |
|  | } /\*End of isEmpty()\*/ |
|  |  |
|  | void display() |
|  | { |
|  | struct node \*p; |
|  | if (isEmpty()) |
|  | { |
|  | printf("\nQueue is empty\n"); |
|  | return; |
|  | } |
|  | printf("\nQueue is :\n"); |
|  | p = rear->link; |
|  | do |
|  | { |
|  | printf("%d ", p->info); |
|  | p = p->link; |
|  | } while (p != rear->link); |
|  | printf("\n"); |
|  | } |