|  |  |  |
| --- | --- | --- |
|  |  | #include <stdio.h> |
|  |  | void enque(int,int []); |
|  |  | void display(int []); |
|  |  | void deque(int []); |
|  |  | void isempty(int[]); |
|  |  | void isfull(int[]); |
|  |  | int beg=0; |
|  |  | int rear=-1; |
|  |  | int main() |
|  |  | { |
|  |  | int a[10]; |
|  |  | enque(5,a); |
|  |  | enque(6,a); |
|  |  | enque(7,a); |
|  |  | deque(a); |
|  |  | deque(a); |
|  |  | isempty(a); |
|  |  | isfull(a); |
|  |  |  |
|  |  | return 0; |
|  |  | } |
|  |  | void enque(int n,int a[]){ |
|  |  | if(rear!=10){ |
|  |  | rear++; |
|  |  | a[rear]=n; |
|  |  | printf("Queue\n"); |
|  |  | display(a); |
|  |  | } |
|  |  | else{ |
|  |  | printf("Overflow\n"); |
|  |  | } |
|  |  | } |
|  |  | void display(int a[]) |
|  |  | { |
|  |  | if(beg!=-1) |
|  |  | { |
|  |  | for(int i=beg;i>=beg && i<=rear;i++){ |
|  |  |  |
|  |  | printf("%d\n",a[i]); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | } |
|  |  | void deque(int a[]) |
|  |  | { |
|  |  | if(beg!=-1) |
|  |  | { |
|  |  | beg++; |
|  |  | printf("Queue\n"); |
|  |  | display(a); |
|  |  | } |
|  |  | } |
|  |  | void isempty(int a[]) |
|  |  | { |
|  |  | if(beg==-1) |
|  |  | { |
|  |  | printf("empty\n"); |
|  |  | } |
|  |  | else |
|  |  | printf("not empty\n"); |
|  |  | } |
|  |  | void isfull(int a[]) |
|  |  | { |
|  |  | if(beg==10) |
|  |  | { |
|  |  | printf("full\n"); |
|  |  | } |
|  |  | else |
|  |  | printf("not full\n"); |
|  |  | } |

90 changes: 90 additions & 0 deletions

90 [Day 4/Stack ADT.c](https://github.com/SrikanthSaravanan12/Training/commit/99c26d24c6f45b1b82f7fa5689875a254beabff8#diff-caf9eb2f910656416472f9b501c548368ac6c9449b9830114b6ea32cd7dd814c)

|  |  | @@ -0,0 +1,90 @@ |
| --- | --- | --- |
|  |  | #include <stdio.h> |
|  |  | void push(int,int []); |
|  |  | void display(int []); |
|  |  | void pop(int []); |
|  |  | void isempty(int[]); |
|  |  | void isfull(int[]); |
|  |  | int top=-1; |
|  |  | int main() |
|  |  | { |
|  |  | int a[10]; |
|  |  | push(5,a); |
|  |  | push(6,a); |
|  |  | push(7,a); |
|  |  | pop(a); |
|  |  | pop(a); |
|  |  | isempty(a); |
|  |  | isfull(a); |
|  |  |  |
|  |  | return 0; |
|  |  | } |
|  |  | void push(int n,int a[]){ |
|  |  | if(top!=10){ |
|  |  | top++; |
|  |  | a[top]=n; |
|  |  | printf("stack\n"); |
|  |  | display(a); |
|  |  | } |
|  |  | else{ |
|  |  | printf("Overflow\n"); |
|  |  | } |
|  |  | } |
|  |  | void display(int a[]) |
|  |  | { |
|  |  | if(top!=-1) |
|  |  | { |
|  |  | for(int i=top;i>=0;i--){ |
|  |  |  |
|  |  | printf("%d\n",a[i]); |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | } |
|  |  | void pop(int a[]) |
|  |  | { |
|  |  | if(top!=-1) |
|  |  | { |
|  |  | top--; |
|  |  | printf("stack\n"); |
|  |  | display(a); |
|  |  | } |
|  |  | } |
|  |  | void isempty(int a[]) |
|  |  | { |
|  |  | if(top==-1) |
|  |  | { |
|  |  | printf("empty\n"); |
|  |  | } |
|  |  | else |
|  |  | printf("not empty\n"); |
|  |  | } |
|  |  | void isfull(int a[]) |
|  |  | { |
|  |  | if(top==10) |
|  |  | { |
|  |  | printf("full\n"); |
|  |  | } |
|  |  | else |
|  |  | printf("not full\n"); |
|  |  | } |
|  |  |  |