**A Mini Project Report**

**on**

**ELEVATOR SIMULATION**

**Course: Data Structures Lab**

**Sem: III Sec: CSE-B**

**By**

**K.ANISHA(1602-18-733-068)**

**K.ANISHKA(1602-18-733-069)**

**G.MEGHANA(1602-18-733-086)**



**Department of Computer Science & Engineering**

**Vasavi College of Engineering (Autonomous)**

**Ibrahimbagh, Hyderabad-31**

**2019**

**TABLE OF CONTENTS**

**S.no. Topic name Pg.no.**

**1. Project Abstract 4**

**2. Description 5-6**

**3. Code 7-17**

**4. Output 18-25**

**5. Constraints &Test Cases 26-34**

**6. References 35**

**PROJECT -ABSTRACT**

**ELEVATOR**

An elevator is a type of vertical transportation device that moves people or goods within floors of a building.

Our project deals with the simulation of an elevator. Firstly, our program will display the present floor where the elevator is stationed. Then, when someone want to access the elevator, they need to choose if they want to go to the top floors or bottom floors with respect to their current floor position. It will also display the message “Overload” when the number of people exceeds the maximum capacity of the elevator.

Our project will display the floor numbers and the user can choose accordingly. The elevator moves based on the preference order. So, it basically follows the principle ‘’First In First Out’’. Our project deals with the use of circular queues. Two pointers front and rear are declared and the elevator moves up and down accordingly.

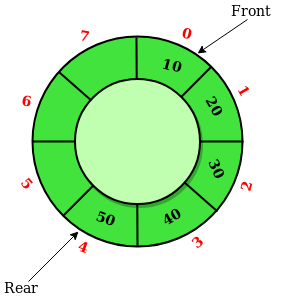
As the elevator moves from floor to floor, it displays the current floor number reached.

When the elevator arrives at floor chosen, the message “Floor reached” is displayed and that particular floor becomes the current lift position. This is our approach to create a basic simulation of an elevator.

**DESCRIPTION**

Our project deals with the simulation of an elevator. Firstly, it displays the floor numbers in the apartment along with the open doors button, close doors button and the Emergency button. The maximum weight the elevator can hold is 1000kgs. It follows the principle of “**First- In-First-Out**”. People at different floors of the apartment can access the lift. The arrows are displayed to show the movement of the lift as it moves up and down. When the elevator reaches a particular floor, “-5” is displayed. The floor reached by the elevator is displayed with 2 stars beside the floor number. The doors of the elevator start closing after 10000 milliseconds. If one wants to keep the doors open for a longer time compared to that of the threshold, they can press the open doors button. Also, if they want to close the doors before the scheduled time, they can press the close doors button. In case of emergency, the user is prompted to enter “1”. If “1” is entered, the user the prompted to enter the floor number where he wants to get down. The elevator stops at the selected floor and the program is terminated.

The data structure used for this simulation is **“Circular Queues”.** We selected this data structure as insertion and deletion can take place at any point in the circular queue.

It is a linear data structure in which the operations are conducted on the basis of FIFO or First In First Out principle. The last position is connected to the first position to make acircle. It is also known as Ring Buffer.

Insertion into a circular queue requires the use of two pointers i.e. front and rear. Initially, both of them are equal to -1. Elements are added at the rear end and deletion of elements takes place at the front end. As elements are inserted into the queue, the front and rear values are updated accordingly.

How it works?

A circular buffer first starts empty and of some predefined length. For example, this is a 7-element buffer:

[Circular buffer - empty.svg](https://en.wikipedia.org/wiki/File:Circular_buffer_-_empty.svg)

Assume that a 1 is written into the middle of the buffer (exact starting location does not matter in a circular buffer

Circular buffer - XX1XXXX.svg

Then assume that two more elements are added — 2 & 3 — which get appended after the 1:

Circular buffer - XX123XX.svg

If two elements are then removed from the buffer, the oldest values inside the buffer are removed. The two elements removed, in this case, are 1 & 2, leaving the buffer with just a 3:

Circular buffer - XXXX3XX.svg

If the buffer has 7 elements then it is completely full:

Circular buffer - 6789345.svg

A consequence of the circular buffer is that when it is full and a subsequent write is performed, then it starts overwriting the oldest data. In this case, two more elements — A & B — are added and they *overwrite* the 3 & 4:

Circular buffer - 6789AB5.svg

Alternatively, the routines that manage the buffer could prevent overwriting the data and return an error or raise an [exception](https://en.wikipedia.org/wiki/Exception_handling). Whether or not data is overwritten is up to the semantics of the buffer routines or the application using the circular buffer.

Finally, if two elements are now removed then what would be returned is **not** 3 & 4 but 5 & 6 because A & B overwrote the 3 & the 4 yielding the buffer with:

Circular buffer - X789ABX.svg

### Advantages of Circular Queues

1. It takes up less memory than the linear queue.

2. A new item can be inserted in the location from where a previous item is deleted.

3. Infinite number of elements can be added continuously but deletion must be used.

Time complexity of enQueue() i.e. insert operation into a circular queue is O(1) as there is no loop in any of the operation.

The libraries included are-> stdio.h, stdlib.h ,string.h ,time.h

**CODE->**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**#include<time.h>**

**int front=-1,rear=-1,a[30],b,c=1,o=0,g,h,desired\_floor,n,temp;**

**char s[4];**

**int m=0;**

**struct que{**

**char data;**

**struct que \*next;**

**};**

**struct que \*front1=NULL;**

**struct que \*rear1=NULL;**

**void arrowup();**

**void arrowdown();**

**int full(int);**

**void pattern(int);**

**void delete(int,int);**

**void insert(int,int);**

**void search(int,int);**

**void display(int);**

**int movement(int,int);**

**void emergency();**

**void main()**

**{**

**int i,j,k=0,t,currentfloor=0,e,d,y,x,z,q,u,l;**

**printf("The number of floors in the apartment are-> 16\n");**

**for(i=1;i<=4;i++)**

**{**

**for(j=1;j<=4;j++)**

**{**

**printf("%d ",k);**

**k++;**

**}**

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

**}**

**printf("<|> for opening\t >|< for closing\t EMERGENCY\n");**

**printf("The maximum load the lift can weigh is =1000 kgs\n");**

**printf("Current floor number is-> 0\n");**

**for(i=0;i<16;i++)**

**{**

**insert(16,i);**

**}**

**while(c==1)**

**{**

**scanf("%d",&c);**

**if(c!=1)**

**{**

**break;**

**}**

**printf("Floor at which people want to access the lift->");**

**scanf("%d",&h);**

**if(h==16)**

**{**

**printf("Pls enter valid floor number");**

**scanf("%d",&h);**

**}**

**e=currentfloor;**

**x=e;**

**currentfloor=h;**

**q=currentfloor;**

**while(q>e)**

**{**

**printf(" %d\n",x++);**

**arrowup();**

**q--;**

**}**

**while(q<e)**

**{**

**printf(" %d\n",x--);**

**arrowdown();**

**q++;**

**}**

**search(n,h);**

**display(n);**

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

**pattern(h);**

**delete(n,h);**

**printf("Doors Opening\n ");**

**printf("People enter the elevator\n");**

**printf("Which floor do u want to got to->?");**

**scanf("%d",&t);**

**currentfloor=movement(h,t);**

**}**

**printf("The elevator is at %d floor.Now, it is at rest\n",currentfloor);}**

**int full(int n)**

**{**

**if(front==0 && rear==n-1)**

**{**

**printf("Queue is full");**

**return 0;**

**}**

**if(front==rear+1)**

**{**

**printf("Queue is full");**

**return 0;**

**}**

**return 1;**

**}**

**void insert(int n,int d)**

**{ int r;**

**r=full(n);**

**if(r==0)**

**{**

**return;**

**}**

**else**

**{**

**if(front==-1)**

**{**

**front++;**

**}**

**rear=(rear+1)%n;**

**a[rear]=d;**

**}**

**}**

**void display(int n)**

**{**

**if (rear >= front)**

**{**

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

**printf("%d ",a[i]);**

**}**

**else**

**{**

**for (int i = front; i < n; i++)**

**printf("%d ", a[i]);**

**for (int i = 0; i <= rear; i++)**

**printf("%d ", a[i]);**

**}**

**}**

**void search(int n,int b)**

**{**

**int p;**

**if (rear >= front)**

**{**

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

**{if(a[i]==b)**

**a[i]=-5;}**

**}**

**else**

**{**

**for (int i = front; i < n; i++)**

**{if(a[i]==b)**

**a[i]=-5;}**

**for (int i = 0; i <= rear; i++)**

**{if(a[i]==b)**

**a[i]=-5;}**

**}**

**}**

**void delete(int n,int b)**

**{**

**if (rear >= front)**

**{**

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

**{if(a[i]==-5)**

**a[i]=b;}**

**}**

**else**

**{**

**for (int i = front; i < n; i++)**

**{if(a[i]==-5)**

**a[i]=b;}**

**for (int i = 0; i <= rear; i++)**

**{if(a[i]==-5)**

**a[i]=b;}**

**}**

**}**

**void pattern(int l)**

**{**

**int i,j,k=0;**

**for(i=1;i<=4;i++)**

**{**

**for(j=1;j<=4;j++)**

**{**

**if(k==l)**

**{**

**printf("\*\*");**

**}**

**printf("%d ",k);**

**k++;**

**}**

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

**}**

**}**

**void arrowup()**

**{**

**int i=5;**

**printf(" ^\n");**

**for(i=0;i<3;i++)**

**{printf(" |\n");}**

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

**}**

**void arrowdown()**

**{**

**int i=5;**

**for(i=0;i<3;i++)**

**{printf(" |\n");}**

**printf(" v");**

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

**}**

**int movement(int currentfloor,int desired\_floor)**

**{**

**int e,q,x,start\_t,end\_t,d,y,g;**

**e=currentfloor;**

**x=e;**

**currentfloor=desired\_floor;**

**q=currentfloor;**

**printf("Doors open and people enter the elevator->\n");**

**printf("Sum of weights of people in the elevator");**

**scanf("%d",&g);**

**if(g>1000)**

**{**

**printf("Overload!!!\n");**

**}**

**start\_t = clock();**

**for(y=0;y<10000000;y++){}**

**end\_t=clock();**

**printf("%d\n",end\_t);**

**if(end\_t<10000)**

**{**

**scanf("%s",s);**

**d=0;**

**if(s[d]=='>')**

**{printf("Doors closing\n");}**

**}**

**if(end\_t>10000)**

**{**

**printf("Doors are about to close->Did everyone get in?\n");**

**scanf("%s",s);**

**d=0;**

**if(s[d]=='>')**

**{printf("Doors closing\n");}**

**else{**

**printf("Doors are still open\n");**

**printf("Doors Closing\n");**

**}}**

**while(q>e)**

**{**

**printf(" %d\n",x++);**

**arrowup();**

**emergency();**

**q--;**

**}**

**while(q<e)**

**{**

**printf(" %d\n",x--);**

**arrowdown();**

**emergency();**

**q++;**

**}**

**search(16,desired\_floor);**

**display(16);**

**pattern(desired\_floor);**

**delete(16,desired\_floor);**

**printf("\n Current floor is %d\n",desired\_floor);**

**delete(16,desired\_floor);**

**return desired\_floor;**

**}**

**void emergency()**

**{**

**int l,p;**

**printf("\nIs there any case of emergency-? if yes,choose Emergency by pressing on 1\n");**

**scanf("%d",&l);**

**if(l==1)**

**{**

**printf("As it is an emergency,pls enter the floor where u want to get off the elevator->\n");**

**scanf("%d",&p);**

**pattern(p);**

**printf("Doors opening\n");**

**printf("Elevator is at %d\n",p);**

**printf("Pls Get down... \n");**

**exit(0);**

**}**

**}**

**OUTPUT->**

**The number of floors in the apartment are-> 16**

**0 1 2 3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**<|> for opening >|< for closing EMERGENCY**

**The maximum load the lift can weigh is =1000 kgs**

**Current floor number is-> 0**

**1**

**Floor at which people want to access the lift->3**

**0**

**^**

**|**

**|**

**|**

**1**

**^**

**|**

**|**

**|**

**2**

**^**

**|**

**|**

**|**

**0 1 2 -5 4 5 6 7 8 9 10 11 12 13 14 15**

**0 1 2 \*\*3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**Doors Opening**

**People enter the elevator**

**Which floor do u want to got to->?6**

**Doors open and people enter the elevator->**

**Sum of weights of people in the elevator 456**

**30394**

**Doors are about to close->Did everyone get in?**

**<|>**

**Doors are still open**

**Doors Closing**

**3**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**4**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**5**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**0 1 2 3 4 5 -5 7 8 9 10 11 12 13 14 15 0 1 2 3**

**4 5 \*\*6 7**

**8 9 10 11**

**12 13 14 15**

**Current floor is 6**

**1**

**Floor at which people want to access the lift->2**

**6**

**|**

**|**

**|**

**v**

**5**

**|**

**|**

**|**

**v**

**4**

**|**

**|**

**|**

**v**

**3**

**|**

**|**

**|**

**v**

**0 1 -5 3 4 5 6 7 8 9 10 11 12 13 14 15**

**0 1 \*\*2 3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**Doors Opening**

**People enter the elevator**

**Which floor do u want to got to->?8**

**Doors open and people enter the elevator->**

**Sum of weights of people in the elevator 1234**

**Overload!!!**

**59262**

**Doors are about to close->Did everyone get in?**

**>|<**

**Doors closing**

**2**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**3**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**4**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**5**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**6**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**7**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**0 1 2 3 4 5 6 7 -5 9 10 11 12 13 14 15 0 1 2 3**

**4 5 6 7**

**\*\*8 9 10 11**

**12 13 14 15**

**Current floor is 8**

**1**

**Floor at which people want to access the lift->7**

**8**

**|**

**|**

**|**

**v**

**0 1 2 3 4 5 6 -5 8 9 10 11 12 13 14 15**

**0 1 2 3**

**4 5 6 \*\*7**

**8 9 10 11**

**12 13 14 15**

**Doors Opening**

**People enter the elevator**

**Which floor do u want to got to->?9**

**Doors open and people enter the elevator->**

**Sum of weights of people in the elevator 45**

**90871**

**Doors are about to close->Did everyone get in?**

**<|>**

**Doors are still open**

**Doors Closing**

7

^

|

|

|

**Is there any case of emergency-? if yes, choose Emergency by pressing on 1**

**1**

**As it is an emergency, pls enter the floor where u want to get off the elevator->**

**8**

**0 1 2 3**

**4 5 6 7**

**\*\*8 9 10 11**

**12 13 14 15**

**Doors opening**

**Elevator is at 8**

**Pls Get down...**

**CONSTRAINTS**

1. The number of floors in the apartment is 16.
2. The lift can be accessed one floor at a time i.e. only one button is active inside the elevator display.
3. The time after which the doors of the lift start closing is 10000 milliseconds.

**TEST CASES**

Case 1: The doors of the elevator start closing at 10,000 milliseconds. If the time limit exceeds 10,000 and if the close button is pressed, doors get closed. If the open doors button is pressed, the doors remain open. If the time is less than 10,000 milliseconds, the doors can be closed manually by pressing the close doors button.

Code output->

**The number of floors in the apartment are-> 16**

**0 1 2 3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**<|> for opening >|< for closing EMERGENCY**

**The maximum load the lift can weigh is =1000 kgs**

**Current floor number is-> 0**

**1**

**Floor at which people want to access the lift->3**

**0**

**^**

**|**

**|**

**|**

**1**

**^**

**|**

**|**

**|**

**2**

**^**

**|**

**|**

**|**

**0 1 2 -5 4 5 6 7 8 9 10 11 12 13 14 15**

**0 1 2 \*\*3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**Doors Opening**

**People enter the elevator**

**Which floor do u want to got to->?8**

**Doors open and people enter the elevator->**

**Sum of weights of people in the elevator 90**

**42091**

**Doors are about to close->Did everyone get in?**

**<|>**

**Doors are still open**

**Doors Closing**

**3**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**4**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**5**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**6**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**7**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes,choose Emergency by pressing on 1**

**0**

**0 1 2 3 4 5 6 7 -5 9 10 11 12 13 14 15 0 1 2 3**

**4 5 6 7**

**\*\*8 9 10 11**

**12 13 14 15**

**Current floor is 8**

Case 2: When number of people getting in the lift is greater than the capacity of the lift, we have displayed the message “OVERLOAD!!!” and the program continues when the condition is either true or false.

Code output->

**The number of floors in the apartment are-> 16**

**0 1 2 3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**<|> for opening >|< for closing EMERGENCY**

**The maximum load the lift can weigh is =1000 kgs**

**Current floor number is-> 0**

**1**

**Floor at which people want to access the lift->3**

**0**

**^**

**|**

**|**

**|**

**1**

**^**

**|**

**|**

**|**

**2**

**^**

**|**

**|**

**|**

**0 1 2 -5 4 5 6 7 8 9 10 11 12 13 14 15**

**0 1 2 \*\*3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**Doors Opening**

**People enter the elevator**

**Which floor do u want to got to->?6**

**Doors open and people enter the elevator->**

**Sum of weights of people in the elevator 1230**

**Overload!!!**

**28365**

**Doors are about to close->Did everyone get in?**

**<|>**

**Doors are still open**

**Doors Closing**

**3**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes, choose Emergency by pressing on 1**

**0**

**4**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes, choose Emergency by pressing on 1**

**0**

**5**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes, choose Emergency by pressing on 1**

**0**

**0 1 2 3 4 5 -5 7 8 9 10 11 12 13 14 15 0 1 2 3**

**4 5 \*\*6 7**

**8 9 10 11**

**12 13 14 15**

**Current floor is 6**

**0**

**The elevator is at 6 floor. Now, it is at rest.**

Case 3: We have also given the option of emergency in case of any emergency situations like power cut, fire accidents, etc. The user is provided the option of emergency so that when the situation calls it can be activated and the user is given the option to which floor he wants to go.

Code Output->

**The number of floors in the apartment are-> 16**

**0 1 2 3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**<|> for opening >|< for closing EMERGENCY**

**The maximum load the lift can weigh is =1000 kgs**

**Current floor number is-> 0**

**1**

**Floor at which people want to access the lift->3**

**0**

**^**

**|**

**|**

**|**

**1**

**^**

**|**

**|**

**|**

**2**

**^**

**|**

**|**

**|**

**0 1 2 -5 4 5 6 7 8 9 10 11 12 13 14 15**

**0 1 2 \*\*3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**Doors Opening**

**People enter the elevator**

**Which floor do u want to got to->?6**

**Doors open and people enter the elevator->**

**Sum of weights of people in the elevator 890**

**28735**

**Doors are about to close->Did everyone get in?**

**<|>**

**Doors are still open**

**Doors Closing**

**3**

**^**

**|**

**|**

**|**

**Is there any case of emergency-? if yes, choose Emergency by pressing on 1**

**1**

**As it is an emergency, pls enter the floor where u want to get off the elevator->**

**3**

**0 1 2 \*\*3**

**4 5 6 7**

**8 9 10 11**

**12 13 14 15**

**Doors opening**

**Elevator is at 3**

**Pls Get down...**

**REFERENCES->**

|  |
| --- |
| https://mail.google.com/mail/u/0/images/cleardot.gif  https://mail.google.com/mail/u/0/images/cleardot.gif |
|

* <https://www.quora.com/What-is-the-benefit-of-a-circular-queue>
* <https://www.geeksforgeeks.org/circular-queue-set-1-introduction-array-implementation/>
* <https://en.wikipedia.org/wiki/Circular_buffer>
* <https://www.tutorialspoint.com/c_standard_library/c_function_clock.htm>

TEXT BOOK REFERED: ELLIS HORROWITZ ,SATRAJ SAHANI,SUSAN ANDERSON FORD ,“ *FUNDAMENTALS OF DATA STRUCTURES IN C”,2nd EDITION(2008),UNIVERSITY PRESS.*