Southern Technical University
Technical Institute / Qurna
Dep. of Computer Systems Techniques

Second class
Subject : Data Structures
Lecturer : Israa Mahmood Hayder
Lecture no.1

الطابور (Queue)

-الاسبوع الرابع عشر-

الرابع عشر	الطابور Queue. - تمثيل الطابور باستخدام المصفوفة. - الطابور الموصولlinked queue.

B// Rationale (مبررات الوحدة) :-

A queue is like the checkout line at the supermarket where the first customer is at the front of the line, the second customer is next in line, and so on until you reach the last customer who is at the back of the line. In the real world, queues are used in programs that process transactions. A transaction is a set of information such as an order form. Transaction information is received by a program and then placed in a simple queue waiting to be processed by another part of the program.

C// Central Ideas (الفكرة المركزية):-

- Define the Queue
- The Business of Queues
- Insertion operation algorithm
- Deletion operation algorithm

D// Objectives (أهداف الوحدة):-

After studying this unit, the student will be able to:-

- Define the Queue and its pointers
- Represent the Queue in C++

- Write Insertion operation algorithm and Program
- Write Deletion operation algorithm and Program

1- A Queue

A queue is like the checkout line at the supermarket where the first customer is at the front of the line, the second customer is next in line, and so on until you reach the last customer who is at the back of the line. Customers check out of the supermarket in the order they arrive in the line. That is, the first customer is the first one to check out. This is referred to as first in, first out (fifo).

The same concept applies to a queue in your program. A *queue* is a sequential organization of data. Data is accessible using fifo. That is, the first data in the queue is the first data that is accessible by your program. In this unit, you will explore the simplest type of queue, a fixed size, first in, first out queue using an array and the Circular Queue.

2-The Business of Queues

Queues are very important in business applications that require items to be processed in the order they are received. The supermarket checkout line is a queue that most of us have experienced, but you won't be creating a supermarket checkout line in a program unless the program is designed to simulate a checkout line.

In the real world, queues are used in programs that process transactions. A transaction is a set of information such as an order form. Transaction information is received by a program and then placed in a simple queue waiting to be processed by another part of the program.

Let's return to a supermarket to see how this works. The cash register is a computer that runs a transaction program that, among other things, processes the bar code on each product scanned at the checkout counter.

One of the first steps to processing the bar code is to look up the price. There could be 20 or more cash registers in a busy supermarket all trying to look up prices at the same time. However, the computer can process only one bar code at a time. The program that look ups prices manages the demand by using a simple queue in which each new request is placed at the back of the queue, and the program looks up the bar code that is at the front of the queue.

Many other applications use a simple queue to maintain the order in which to process items. These include programs that process stock and bond trades and those that process students registering for a course. Queues are also used within a computer to manage printing.

Data organized by a queue may be stored in an array. The queue determines the array element that is at the front and back of the queue. The array is not the queue. Likewise, the queue is not the array. Both are two separate things. This is an important concept to grasp and one that may be difficult to understand at first.

Definition of Queue :

The queue is a linear list that permits the insertion to be performed at one end and the deletion on the other end , such a linear list is frequently referred to as **First In – First Out** (**FIFO**) list .

F ← 0

R ← 0

} الحالة الابتدائية

The vector representation of the queue requires pointers **F** and **R** , which denote the position of it's **Front** and **Rear** Respectively.

3- Linear Implementation of operations on Queue:

For efficient processing of queues, we shall therefore need two indices so that we can keep track of both the front and the rear of the queue without moving any entries. To append an entry to the queue, we simply increase the rear by one and put the entry in that position. To serve an entry, we take it from the position at the front and then increase the front by one. This method, however, still has a major defect: Both the front and rear indices are increased but never decreased.

The Array and the Queue in C++

Data organized by a queue may be stored in an array. The queue determines the array element that is at the front and back of the queue. The array is not the queue. Likewise, the queue is not the array. Both are two separate things. This is an important concept to grasp and one that may be difficult to understand at first.

The C++ queue program is organized into three files: the queue.h file, the queue.cpp file, and the queueProgram.cpp file. The queue.h file, shown next, sets the default size of the array and defines the Queue class. The Queue class declares size, front, and back attributes that store the array size and the index of the front and back of the queue. The Queue class also declares a pointer that will point to the array. In addition to these, the Queue class defines a set of member functions that manipulate the queues. These are explained later in this section.

```
//queue.h
#define DEFAULT_SIZE 8
class Queue{
  private:
    const int size;
    int front;
    int back;
    int* values;
  public:
    Queue(int size = Default_Size);
    virtual ~Queue();
    bool isFull();
    bool isEmpty();
    void enqueue(int);
    int dequeue();
};
```

The queue.cpp file contains the implementation of the member functions for the Queue class. There are six member functions defined in this file: Queue(), ~Queue(), isFull(), isEmpty(), enqueue(), and dequeue().

The Queue() member function is a constructor, which is passed the size of the array when an instance of the Queue class is declared. If the constructor is called with no parameters, then the default size is used; otherwise, the value passed to the constructor is used. The value of the array size is assigned to the attribute size by the first statement within the constructor.

The second statement uses the new operator to declare an array of integers whose size is determined by the size passed to the constructor. The new operator returns a pointer to the array, which is assigned to the values pointer. The last two statements in the constructor initialize the front and back attributes to zero.

The ~Queue() member function is the destructor and uses the delete operator to remove the array from memory when the instance of the Queue class goes out of scope.

The isFull() member function determines if there is room available in the queue by comparing the calculated value of the back of the queue with the value of the front of the queue,. Notice that the expression that calculates the back of the queue is very similar to the expression in the enqueue process (see the “Enqueue” section of this chapter), and both produce the same result. The queue is full when the back index is 1 behind the front. Placing another element in the queue would overwrite the front element and corrupt the queue. The modulus operator is used again to make this a circular queue, so when you’re at element 7 on the back, the next element to look at is element 0.

Figure 5-4: The isFull() member function determines if there is room to place another item on the back of the queue.

Quiz1

1- What is a queue?

2- Why is the isFull() member method called?

3- Why is the isEmpty() member method called?

4- What happens to the data stored on the array when the data is removed from the queue?

Operations On Queues :-

1- Insertion

If R ≥ N then write "**Error Queue is Over Flow** "

Else

R ← R+1

q(R) ← item

end if

if F=0 then F ←1 **(F إذا كان العنصر المضاف هو أول عنصر نحد ث)**

end if

2- Empty the queue

F ← 0
R ← 0

3- Deletion :

If F=0 then write "**Error Queue is Under Flow** "

Else

item ← q(F)

if F=R then R ←F←0 **(إذا كان أخر عنصر في الطابور)**

else

F ← F+1

end if

end if

Quiz2: -١- ite algorithm to insert an item to the Queue

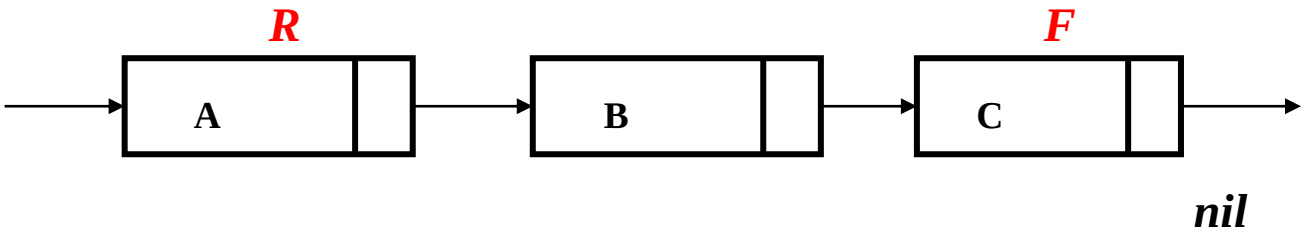
2- Write algorithm to delete an item from the Queue, 3- clear the Queue.

Queues Using Linked Lists

The queue in was created using an array to store data. As you'll recall, the array is separate from the queue. Data is assigned to elements of the array. The queue itself consists of two variables called front and back. Each points to the array element that is at the front of the queue or at the back of the queue. When data is removed from the front of the queue, the program changes the value of the front variable to point to the next array element. However, the data removed from the queue remains assigned to the array. That is, data isn't removed from memory.

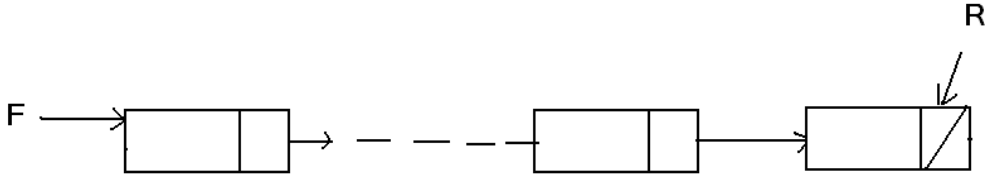
There is a serious problem with using arrays to store data for queues: you must know the size of the array when you write the program. An array can store only a specific maximum number of elements at any point in time, similar to an architect designing a specific space for a box office that can accommodate a maximum number of fans at any point in time.

However, there is a difference between exceeding the number of array elements and overflowing the space around the box office: unlike the stadium, there is no parking lot for fans to gather in while waiting to get in the queue to purchase tickets inside a computer.



Linked Queue :-

Like single list it's start is the front and rear points to last element .



Operation on linked Queue:-

1- Insertion :-

Algorithm :-

```
If R= nil then new ( R) ;
    read(R.data);
    R.next =nil;
    F=R ;
Else
   New (P) ;
   R.next = p ;
   R = P ;
End if'
End insert;
```

2- Deletion :-

Algorithm :-

```
if    F =nil  then "under flow"
else
{
   P=F;
   F = F.next;
   item← F.data;
   delete (p);
  }
If F= nil then R= nil ;
End delete
```

Quiz3:

1) Delete an element from linked queue
2) Write the algorithm to push an item to the linked Queue

References:

1- Data Structures Demystified, by Jim Keogh and Ken Davidson, ISBN:0072253592, McGraw-Hill/Osborne © 2004.

2- هياكل البيانات / الطبعة الثانية، تاليف د.عصام الصفار،اصدارات السفير للنشر/ بغداد، ٢٠٠١

4- الحقيبة التعليمية لمادة هياكل البيانات ، اعداد نفارت يوسف الياس