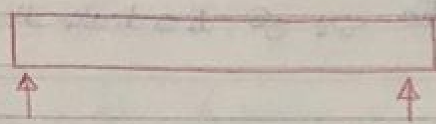
Queues

Queue is a Container datatype that stores items in "First in first out" order (FIFO)

ال Queue ما هو الا list معدلة وهى انها دايما بعمل insert فى الآخر



(مؤخرة الـqueue) Rear item (is most recent addition to queue)

Front item (مقدمة) (has waited longer than others)

لما ال user يجوز يسترجع من ال queue بياخد من الأول ولو عاوز يكتب بيكتب فى الآخر

ونلاحظ ان ال elements فى النص ليس لهم أى significant

* Difference between queues & lists

- queue is a restricted form of a list
- Additions to the queue must occur at the rear
- Deletions from the queues must occurs at the front

ال queues بتستخدم فى OS بشكل كبير

اول حاجه ال buffer مفيش operating system مفهوش buffer

عشان ال speed match

Subject:
Date:

Queuing theory

بندرس الـbehaviour بتاع الـqueue لما بيحصل حاجات بيطلب خدمة
زي لما نروح شئون الطلبة وفي البولي (يعني بالك لازم ندرس الـ(response time)
ده مهم من غيره الـqueue بتاعي بيبوظ & synchronizations

The difference between queues & lists

Queues	lists
removal بتعمل insert في أي مكان ومفيش	لازم insert في الآخر
Delete operation بس فيه	و remove من الأول

* الـ priority queue دي special case من الـ inorder list
و البيحدد الـ order هو الـ priority

Queue (list من الـ special type (لما هو بالظبط

- Static
  - array
- Dynamic
  - linked structure

very common (in embedded system)
as queue has fixed size
that depend on the buffer
size that is almost fixed

* No limitation on size

Note

Queue is different from stack as stack has only one place to input in it & read from it

APPlications

لو عندي Printer فيه buffer عشان يخزن الحاجة اللي عايزين نطبعها و فيه Counter في ال Computer عشان يبعت حاجة ال printer عشان يطبعها هياخد منه وقت قليل فالسرعة عالية جداً (electronic speed) بس ال Printer مش هيستقبل ال حاجة بنفس السرعة فعشان كده محتاج buffer وخلي بالك العملية اني ابعت الحاجة لل printer واني اطبع ده بيتم بسرعة ال printer وعشان مفيش وقت ال CPU و بيبعت لل IO device هو يبقى handle ال process دي

* عشان اخدم أكتر من user في نفس الوقت و عندي single resource فمحتاج استخدم queue

* CPU Scheduling: هل في tasks كتير شغالة a time
process scheduling

P1 → P2 → ... → PN

CPU

عندي اكتر من Process بتحصل في بقسم time slots بحيث كل task تاخد 1 timeslot و لما فياخد P1 في timeslot بتاعها لو خلصت تمشي لو مخلصتش تتحط في برجعها في اخر queue و هكذا

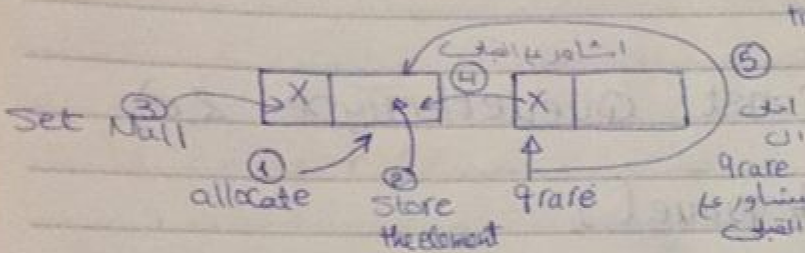
Interrupt | P3 | P1 | PN | ... | P2 | P1 |

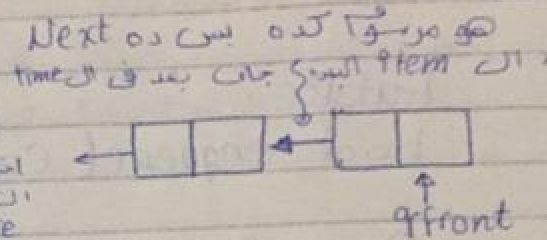
Task Schedule

ال Interrupt لو ال user عمل click ع أي مكان محتاج أوقف ال queue و اروح اشوف ال user كان عايز إيه

Queue design

General way to insert in queue

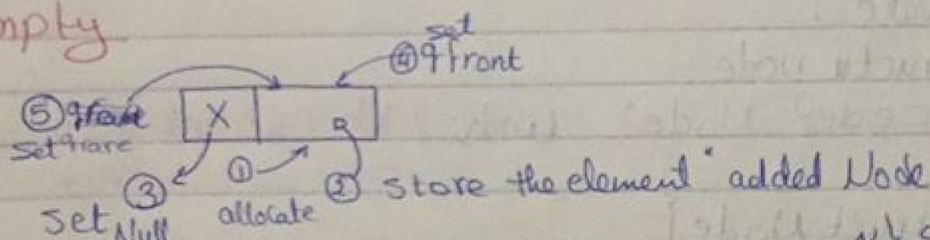




④ next == added node

ال order مهم جداً

Empty

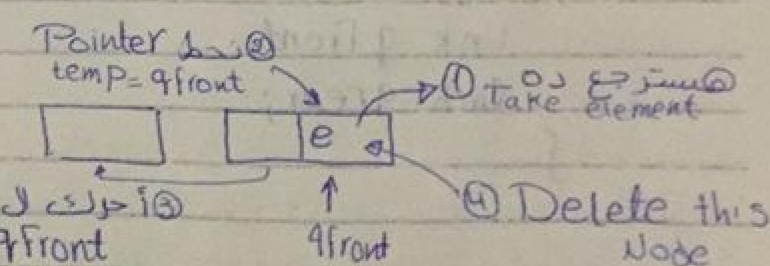


حتى بالشكل ده

General ~~&empty~~ emp

زي بعض في ماعدا رقم "4"

Dequeue



Notes next لو عملت ٤ قبل 3 ال

هيضيع مني

* لما ب read ال queue بشيل ال item اللي بقرأه

* الـ sequence مهم جداً

Subject:
Date:

Queues operation

Enqueue insert enqueue

Pre Condition

لازم فيه مكان احط فيه Queue جديدة
لازم الـ Queue ما تكون empty

Post Condition

القى لازم الـ (Queue Post = Queue after) = Queue before
بس بعلما الـ x تكون added في آخر الـ Queue (rear of queue)

~~enqueue~~ → insert in queue

dequeue → remove from the queue يرجع queue data type

← احنا في الـ queue لا نحتاج الـ Prev Pointer عشان انا مش برجع أنا عايز اعمل next بس يبقى الـ queue مش متجيبه نعملوا doubleLinkedList
← مفيش Queue عشان next و First طبيعة الـ Current في باحد دائماً منه أول و احد مش متاح

Subject:
Date:

Q.front ' Get the data wzt removing it

Pre-Condition

Queue is not empty

Post Condition

Qafter = Qbefore

Return.

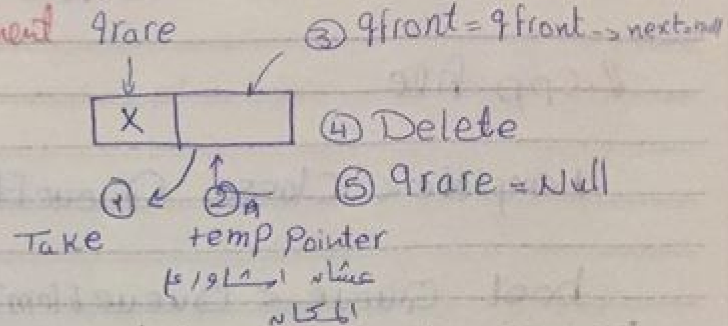
Return item stored @ a queue front

```
template < class QueueElem Type >

Queue Queue < QueueElemType>
 { front ( )

  assert ( ! is Empty ( ))
  Return q front -> Element;
    }
```

Special Case: queue has only one element



لو اكتفيت باني اعمل dequeue باول وبس
هيحصل inconsistency هيحصل لانه الـ
qrear بيشاور على مكان في الـ memory
فبالتالي لما الـ queue اتمسحت
فالا تنبه بتشاورله

ـه في الـ constructor بعمل initialize للـ queue بتاعي
وبعرف إنا empty و لا من الـ Pointers ← front
لـ rear
عن طريق اني اعمل check هل هما بيساوي Null ولا

Subject:
Date:

// .h file

```
template < Class QueueElemType >

classQueue {

Public:
bool enqueue( Const QueueElemType &e)

QueueElemTyp dequeue( );

QueueElemType front ( );

bool isEmpty ( );

Private :
struct node
Typedef Node* link; // No Current as nonext as it is
                                               Queue

struct Node{
  QueueElemType elem;
   link next;
   };
 link q front;
 link qrear;
}
```

enqueue

Pre Condition Queue is not empty

Post Condition $Q_{before} = Q_{after}$

// .cpp file

```
templete < Class  QueueElemType >

bool  Queue < QueueElemType >  // Before each member
                                   function
  enqueue ( QueueElemType  & e)
  {
    link  addedNode;
①   addedNode = new Node;
    if (addedNode == Null)
          return  false;
②   added Node → element = e;
③   added Node → next = Null;

    if (qfront == Null)
     qfront = addedNode;
④
    else
     qrear → next = added Node;

⑤   qrear = addedNode;

    return true;

  }
```

Note

ال front ماهي إلا ال next

~~dequeue~~ → ~~[illegible] consume~~

```
main( ){

  CQueue <int> iQueue;
   int i,j
  for (i=0; i<4; i++)
      Cin >> J;
   flag = iQueue.enqueue(J)

while(! iQueue.isEmpty( )) {

                J = iQueue.dequeue( );
        Cout << J;
  }

⊕  J = iQueue.dequeue( );
      Cout >> J;
   J = iQueue.front( );
      Cout >> J;
   J = iQueue.dequeue( );
      Cout >> J;
   enqueue(10);
```

طبع وامسح

الموجود في ال queue

10 ← front → 10

5 → 5

6 → 6

7 ← rear → 7

لو عندك في ال Queue

10
5
6
15

~~10~~ → 10
~~5~~ → 5 → 5
6 → 6
15 → 15 → 15
10 → 10

فلما ينفذ ⊕ هيروح يجيب ال element اللي في queue وهي هتمسح لما تجيبها من ال q يعني هاخد ما من العنصر وتروح تطبعها

بعد كده راح تاخد ال queue عشان ينفذ ال front بس هنا ال front بتجيب ال Data من غير ما تمسحها فتخرج "5" وبعد يتم عمل dequeue فال J خدت ال value "5" لا اتمسحت من ال queue وبعد يتم enqueue(10) فزود "10" في آخر ال queue