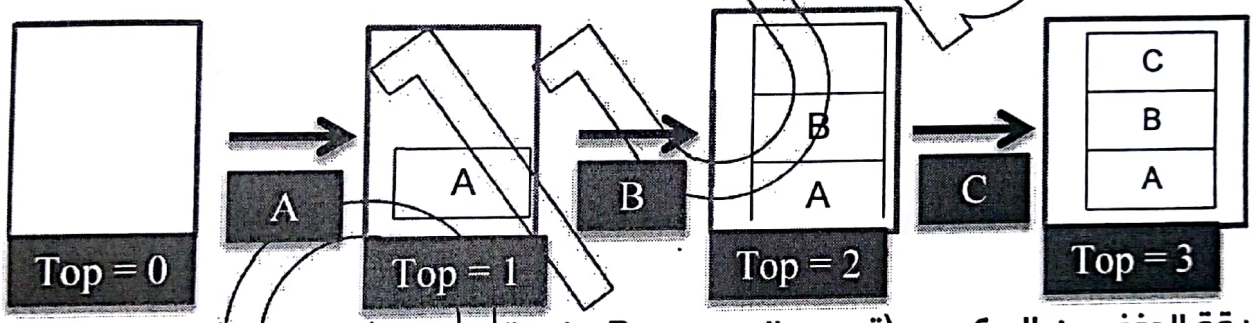
- المكدس $Stack$:

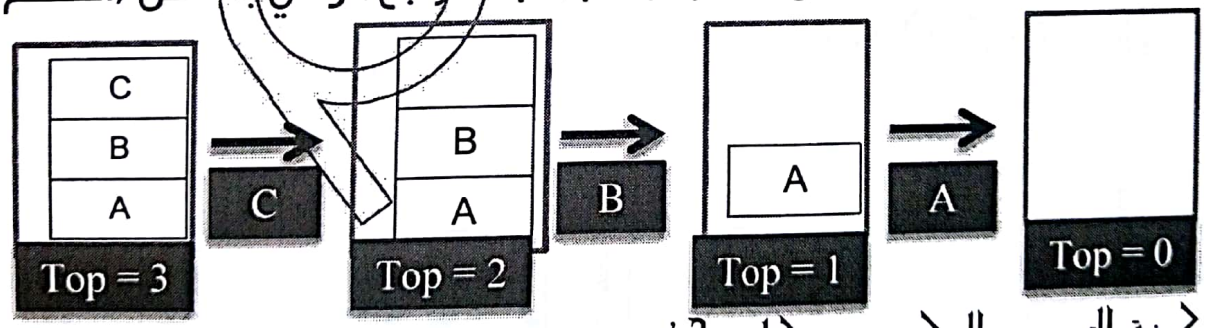
هو عبارة عن قائمة خطية ولكن مع بعض الشروط على الإضافة والحذف حيث تتم الإضافة والحذف من الرأس فقط أي يمكننا اعتباره حالة خاصة من القائمة الخطية .

ويمكن تعريفه على انه عبارة عن هيكل بياني يشبه الوعاء المفتوح من الأعلى فقط (قطرميز مكدوس) ويحوي المكدس دائماً على مؤشر يشير إلى أخر عنصر موجود في المكدس (ويرمز له Top) ، ويتميز المكدس بأن له الخاصية (LIFO) أي أن (Last In First Out) المقصود الداخل أولاً يخرج أخيراً .

شرح طريقة الإضافة في المكدس (تسمى عملية الدفع Push) وهي بالشكل (السهم للإضافة) :



شرح طريقة الحذف من المكدس (تسمى السحب Pop بالنسبة للمراجع) وهي بالشكل (السهم للحذف) :



مثال عن كيفية التعبير عن المكدس بشكل برمجي :

بما أن المكدس هو قائمة خطية فشكل السجل هو نفسه بالشكل :

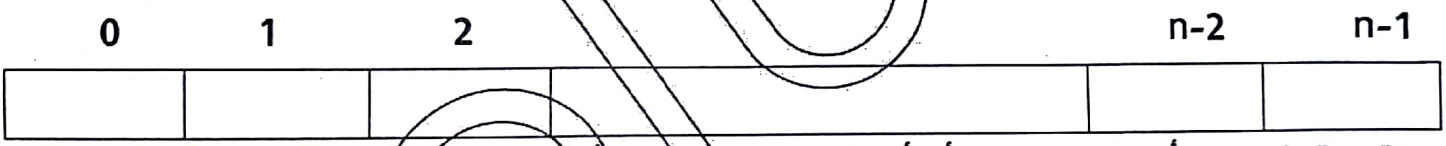
```
struct node{ int val;  stack *next;};
```

العمليات للإضافة والحذف قلنا أنه فقط من الرأس ، وهي ذاتها التي قمنا بها بالقائمة الخطية .

```
node *Top;    // أعلى المكدس
node *x= new stack;
x -> val = 5;
x -> next = NULL;
```

```
Top=x;    // إضافة العنصر الأول
x = new stack;
x -> val = 10;
x -> next = Top;    // جعل العنصر في البداية
Top =x;  // إضافته إلى رأس المكدس
stack *p = Top;  // وضعنا مؤشر جديد كي لا نفقد المؤشر الأساسي
// طباعة العناصر
while (p != NULL ){
cout << p -> val << " " ;  p = p -> next ; }
// حذف العنصر الأول
Top = Top -> next;
p = Top;  // أعدنا المؤشر الوسيط الى رأس القائمة
// طباعة العناصر بعد الحذف
while (p != NULL ){
cout << p -> val << " " ;  p = p -> next ; }
```

ويمكن أيضاً تمثيل المكدس على متجهة (لنسمها A فرضاً) بالشكل التالي :



والقيم تدخل من الأعلى ، وسنفرض أن أعلى المكدس هنا هو آخر عنصر في المصفوفة ، بالتالي سيكون بدل المؤشر سنستخدم دليل ، وسيكون الـ $Top = n - 1$ ، ليدل على أنه في الأعلى ، فسيكون دالة إدخال قيمة المتغير x إلى المكدس بالشكل التالي (سنعملها على شكل دالة) :

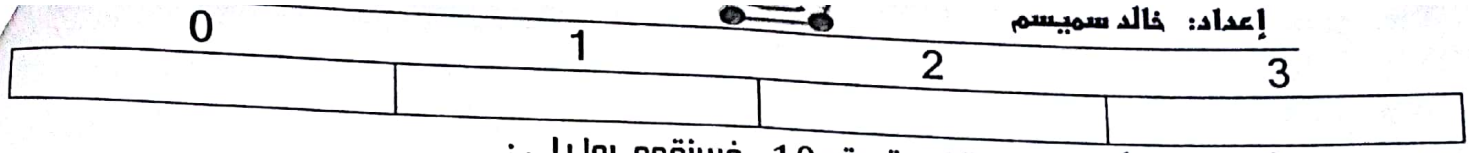
```
void push(int x){
    A[Top]=x;
    Top=Top-1;
}
```

هذه الحالة العامة ، ولكن بشكل أدق يجب علينا التأكد من أن الـ Top لا يتجاوز حدود المتجهة .

بالنسبة لدالة السحب pop فسيكون لها الشكل التالي :

```
anyType pop(){
Top=Top+1;
return A[Top]; }
```

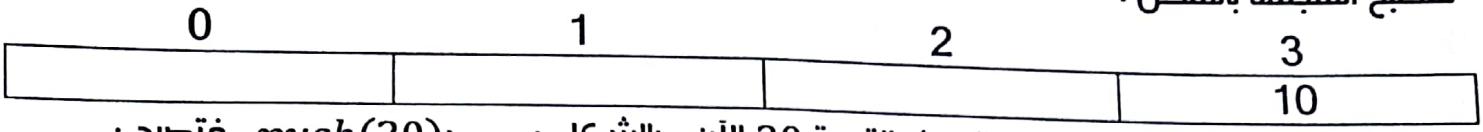
حيث أن : anyType هي نوع العناصر المخزنة بالمكدس ، الآن سنشرح ما سبق بمثال بسيط ، بفرض لدينا متجهة اسمها A مكونة من 4 عناصر من الأعداد الصحيحة ،بالتالي فإنه في البداية سيكون لدينا هنا $Top = 3$ ، (وسنشرح ذلك بفرض أن المتغير Top والمتجهة A هما متحولات عامة) .



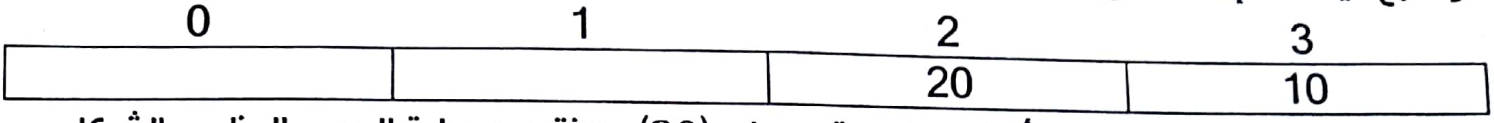
الآن وبفرض أننا سندخل أول عنصر ولتكن قيمته 10 ، فسنقوم بما يلي :

$$push(10);$$

فتصبح المتجهة بالشكل :

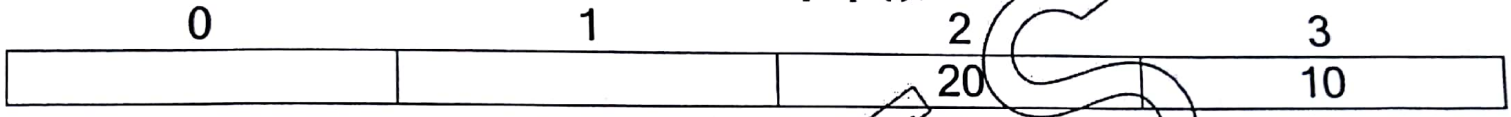


وتصبح قيمة الـ Top مساوية لـ 2 . لنضيف القيمة 20 الآن ، بالشكل : $push(20);$. فتصبح :



ويكون $\text{Top} = 1$ ، الآن بفرض أننا نريد الطباعة سيطبع (20) ، سنقوم بعملية السحب للعناصر بالشكل :

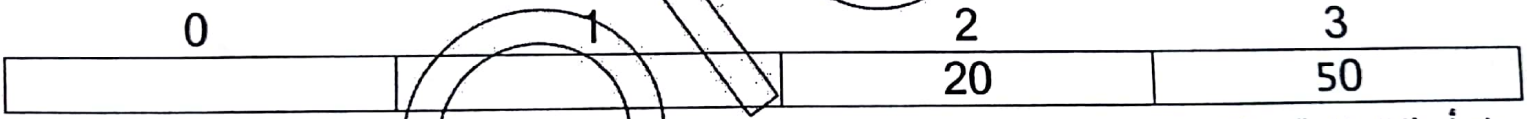
$$cout \ll pop();$$



هنا تم زيادة Top ليصبح $\text{Top} = 2$ ، ثم أعدنا قيمة $\text{A}[2]$. الآن لو قمنا بما يلي سيكون :

$$cout \ll pop();$$

بالتالي سيطبع هنا 10 ، حيث سيزيد بداية الـ Top ليصبح 3 ، ثم سيعيد $A[3]$ ، الآن لو أضفنا قيمة جديدة إلى المكدس ولتكن 50 ، سنقوم بما يلي : $push(50);$ ، وستصبح الـ $Top = 2$.والشكل:



ولو أردنا الطباعة ستكون :

$$cout \ll pop();$$

سيطبع هنا 50 ، وستعود قيمة الـ $Top = 3$.

وهذا هو عمل المكدس بالإضافة والحذف ، ولكن في المؤشرات في الطريقة الأولى يكون فيها الحجز ديناميكياً ويتم حذف الخلايا ، أما هنا فتبقى الخلية موجودة ولكن تتغير قيمتها مثلاً كما رأينا ، ما يهمنا هنا هو معرفة آلية الإضافة والحذف في المكدس أنها دوماً من رأس المكدس فقط أي قمنا بتحديد شروط الإضافة أو الحذف ، وأن دالة الحذف تقوم أولاً بإرجاع القيمة الموجودة في الرأس قبل حذفها ، ولا ننسى أنه يجب وضع شروط في دالة الإضافة والسحب لإلا تتجاوز حدود المتجهة إذا تعاملنا مع متجهة ، وإلى عدم ضياع المؤشر للرأس في حالة تعاملنا مع المؤشرات .

نعلم أنه في حال تم العمل على متجهة أنه يمكننا الوصول إلى أي عنصر فيها ، ولكن الشرح السابق كان لتبيان آلية عمل المكدس حيث لدينا شروط في الإضافة والسحب .

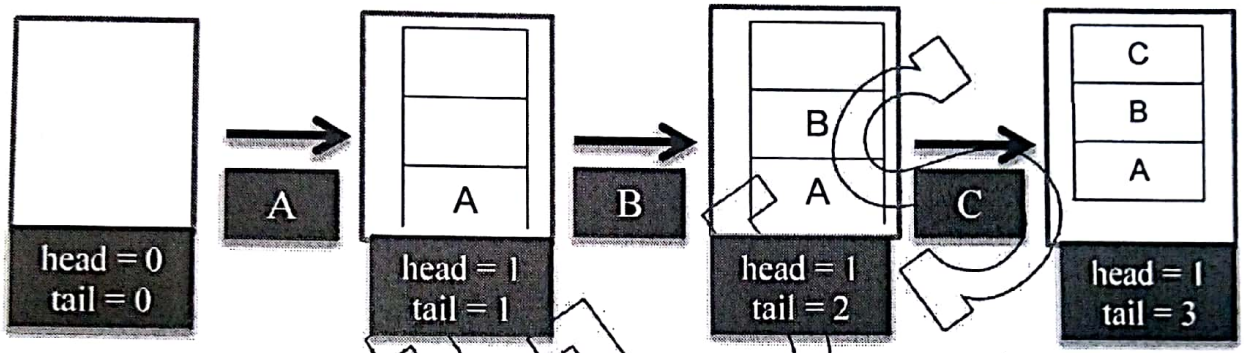
- الرتل (أو الطابور) $Queue$:

هو عبارة عن قائمة خطية حيث تشبه الوعاء المفتوح من الأعلى والأسفل بحيث أنها تحقق الخاصية (FIFO) أي أن (First in First out) أي أول عنصر يدخل هو أول عنصر يخرج، وهي تسمى ايضاً بطوابير الانتظار لأنها تشبه عند انتظار الأشخاص بالدور فأول شخص يدخل هو أول شخص يخرج (مثل فرن الخبز p:) بالتالي الإضافة تتم على ذيل (tail) (نهاية) القائمة، والحذف يتم من الرأس (head) (بداية) القائمة .

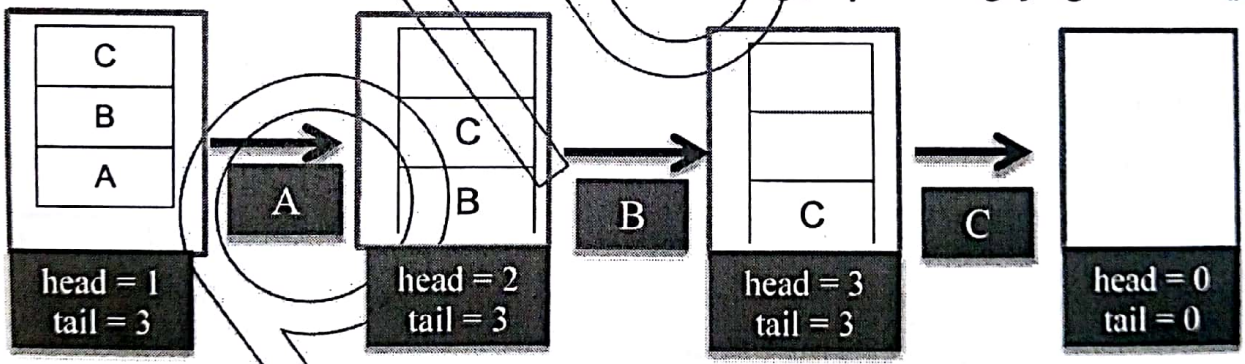
وهنا ايضاً في المحاضرة السابقة تحدثنا على آلية الحذف من البداية ، والإضافة من الخلف ، بالتالي هنا أيضاً أضفنا شروط على الحذف والإضافة ، ويحوي الرتل على مؤشرين مؤشر على الرأس ومؤشر على الذيل .

ويمكن ايضاً كما المكدس أن نمثل الرتل على شكل متجهة ، ويكون هنا إضافة دليل ثاني للعمل عليه.

شرح آلية الإضافة على الرتل (السهم دليل الإضافة) بفرض أن دليل المتجهة يبدأ بـ 1 :



شرح آلية الحذف من الرتل (السهم دليل الحذف) :



تسمى عمية الإضافة على الرتل بـ $enqueue$ ، بينما عملية الحذف تسمى $dequeue$ وإذا مثلنا الرتل على شكل متجهة أعداد صحيحة كمثال بالاسم A ، ولدينا الدليلين $tail = head = 0$ في البداية، وهذه المتغيرات عامة فإن هاتين الدالتين تكونان بالشكل (فرضنا دليل المتجهة يبدأ بـ0) :

```
void enqueue(int x){
    A[tail]=x;
    tail=tail+1;
}
int dequeue(){
    head=head+1;
    return A[head-1]; }
```

وطبعاً لا ننسى إضافة الشروط اللازمة لتتأكد من إذا كان الرتل فارغاً أو ممتلئاً (لأن عدد عناصر المتجهة محدود). الآن سنعطي مثال لشرح الآلية كما فعلنا في المكدس .

لنفرض لدينا في البداية المتجهة A من البعد 4 عناصر ، وليدنا بدايةً $tail = head = 0$ ، عندها :

$enqueue(10);$

0	1	2	3
10			

وهنا يصبح $tail = 1 , head = 0$ ، سنضيف عنصراً آخر : $enqueue(20);$

0	1	2	3
10	20		

وهنا يصبح $tail = 2, head = 0$ ، الآن لو أردنا الطباعة فنقوم بـ :

$cout \ll dequeue( );$

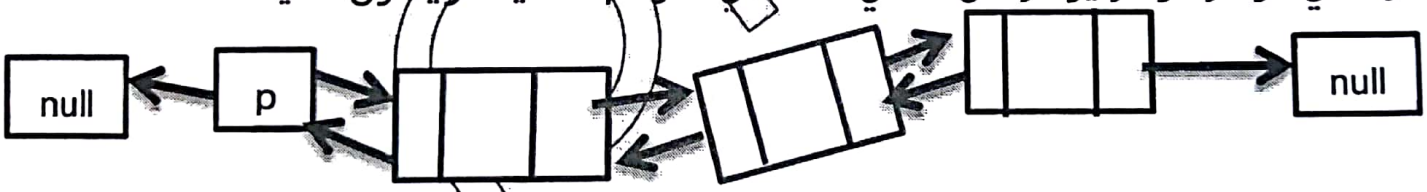
فهنا سيطبع 10 ، ومن ثم سيصبح : $tail = 2, head = 1$ ، لنضف عنصراً جديداً : $enqueu(50);$

0	1	2	3
10	20	50	

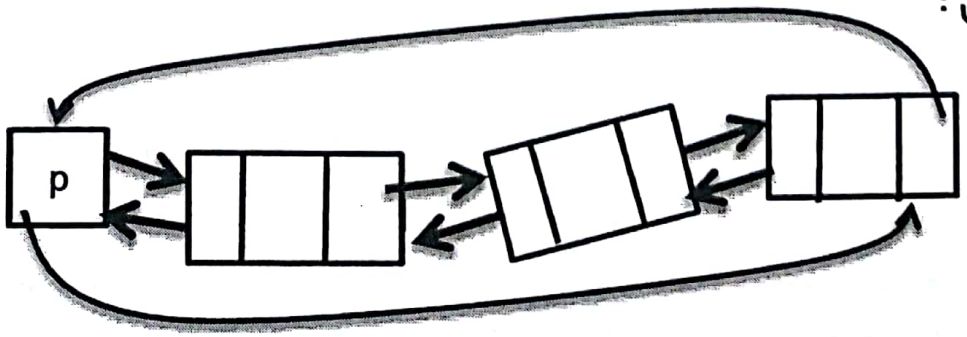
هنا يصبح لدينا $tail = 3, head = 1$ ، وهكذا مع البقية .

افترضنا هنا عكس ما افترضنا في المكدس ، أن الرأس هنا هو بداية المتجهة وليس أخر عنصر .

هذه كانت الأفكار الرئيسية التي قد ناقشها الدكتور ، ثم قام الدكتور بالحديث قوائم ثنائية الارتباط حيث تحوي مؤشرين ، وهي تشبه القوائم الخطية ولكن لها مؤشران ، في القوائم الخطية يوجد مؤشر وحيد يؤشر على العنصر التالي، بينما في القوائم ثنائية الارتباط يوجد مؤشران الأول على العنصر السابق ، والثاني على العنصر التالي ، وأخر مؤشر يؤشر على لا شيء كما في القوائم الخطية . ويكون تمثيلها :



وتحدث أيضاً الدكتور عن الرتل الدائري ، حيث هو يشبه القوائم ثنائية الارتباط ، ولكن المؤشر الذي يسبق الرأس يكون أخر عنصر ، والمؤشر الذي يدل على العنصر التالي للعنصر الأخير يكون على أول عنصر ، فيكون تمثيلها بالشكل :



وهكذا يتم التمثيل .

وتحدث أيضاً عن الرتل ذو أولوية ، حيث أننا نضيف خانة جديدة للرتل (حقل جديد للسجل) وهذا الحقل يمثل أولوية هذا العنصر ، وبالتالي فيقوم بإخراج العناصر بحسب أولويتها ، وصاحب الأولوية الأعلى يخرج أولاً (مثل فرن الخبز فالأولوية للذين يشترون ربطة واحدة أكثر من تلك للذين يريدون كمية كبيرة) وهكذا ..

إن بنى المعطيات الخطية كثيرة ولها استخدامات كثيرة وشائعة ، وكما رأينا في البداية فإن الإساس يعتمد على القائمة الخطية ثم بإضافة بعض الشروط ينتج لدينا بنية معطيات جديدة ، وتختلف كل بنية معطيات عن غيرها بالتعقيد الزمني للعمليات ، ولكل بنية استخداماتها ، وما يجب معرفته أن بنى المعطيات تُعطى في مقررات منفصلة في غير اختصاصات ولكن قام الدكتور بإعطائها لنا لنكون على علم في حال أكملنا في هذا المجال وقرأنا عنها يوماً ما أن يكون يدلنا علم مسبق بها ولو بلمحة بسيطة مساعدة لما بعد ومما يجب ذكره ايضاً أنه في لغات البرمجة كالـ $c++$ والـ $java$ يوجد مكتبات مصممة مسبقاً لبنى المعطيات هذه ولكن ما أخذناه هنا هو الأساس في كيفية بنائها ، ففي بعض الأحيان المكتبات الموجودة في اللغات لا تفي بالغرض بالمطلوب ، وربما نحتاج إلى تصميم بنية معطيات جديدة خاصة بالمسألة التي نعمل بها ، لذلك فإن ما أخذناه هو الأساس في بناء بنى المعطيات أو ما يسمى أيضاً بهياكل البيانات .

وقد قام الدكتور بذكر وتفصيل القائمة الخطية والرتل والمكدس وما تبقى قد تم ذكره بشكل بسيط فقط كفكرة عنه ، وقد طلب منا القيام ببحث ودراسة هذا الموضوع لوحدنا ، فحاولت أن أقوم بوضع شرح إضافي لما ركز عليه ، وذكر الشيء الذي ذكره للبقية (كقوائم ثنائية الارتباط ورتل ذو أولوية) وكطلب للتدريب حاول إنشاء قائمة خطية بالدوال المعرفة عليها ، ثم إنشاء مكدس والدوال المعرف عليه ، وأيضاً نفس الشي لإنشاء رتل ، وحاول الإنشاء باستخدام المؤشرات والمؤشرات لترسخ لمفاهيم أكثر ، وإن العمليات الشهيرة على بنى المعطيات هي البحث والطباعة والإضافة والحذف .