بنى المعطيات الخطية

المحاضرة السادسة
م. لمى السبع

بنى المعطيات المتقدمة

سوف نتعرف في الدروس القادمة على أهم بنى معطيات متقدمة و التي هي :

- بنى المعطيات الخطية :
  - الأشعة Arrays
  - السلاسل الخطية Linked Lists
  - المكدسات Stacks
  - الأرتال Queues
- البنى الشجرية Trees
- البيانات Graphs

بنى المعطيات الخطية :

تُعد البنى الخطية أكثر البنى استعمالاً لتنظيم المعطيات , لأنها تسمح بتنظيم المعطيات التي نريد معالجتها تنظيماً تسلسلياً . من أهم بنى المعطيات الخطية لدينا :

الأشعة - Arrays

من المعروف أن الشعاع صف أحادي البعد حيث نذكر إنه يمكن الوصول إلى أي عنصر من شعاع وصولاً مباشراً .

السلاسل الخطية - Linked Lists

يمكن تمثيل السلاسل الخطية إما باستخدام الأشعة أو بالمؤشرات . حيث يمكننا في الحالتين تطبيق العمليات التالية عليها : إيجاد طول السلسلة , قراءة السلسلة (تصاعدياً و تنازلياً) , بحث عن عنصر , حشر عنصر و حذف عنصر .

1- تمثيل السلاسل الخطية باستخدام الأشعة:

و يتم ذلك بتعريف سجل مؤلف من حقلين : حقل طول السلسلة و حقل صف عناصر السلسلة . *مثال*:

```
const lmax=100;
struct listType {
        int  length;
        elem val[lmax];     //  elem = أي نمط معطيات معروف
};
```

سلبيات تمثيل السلاسل باستخدام الأشعة :

إن الطول الأعظم للسلسلة محدود , فلا يمكن إضافة عناصر أكثر من طول السلسلة . لذلك يجب أخذ هذه الفكرة بعين الاعتبار أثناء تعريف السلسلة و حجز طول زائد حتى لا يحدث خطأ . بالإضافة إلى ذلك فأن عمليات الحشر و الحذف تسبب عدداً كبيراً من الإزاحات في الذاكرة .

ملاحظات : `L.length=0` ⇔ `L` فارغة

`L.val[i]` = العنصر ذا الرقم `i` من السلسلة `L`

حذف عنصر x من الموقع k في السلسلة الخطية	إضافة عنصر x في الموقع k من السلسلة الخطية

`listType delet(listType L,int k)`
`{`
`  if ((k<L.length)&&(k>=0))`
`  {`
`  for (int i=k;i<L.length-1;i++)`
`  L.val[i]=L.val[i+1];`
`  --L.length;`
`  }`
`  return L;`
`}`
	`listType insert(listType L,elem x,int k)`
	`{ if ((L.length<lmax)&&(k<=L.length))`
	`  {`
	`  for (int i=L.length-1;i>=k;i--)`
	`  L.val[i+1]=L.val[i];`
	`  L.val[k]=x;`
	`  ++L.length;`
	`  }`
	`  return L;`
	`}`

تمثيل السلاسل الخطية باستخدام المؤشرات

- لحل هذه المشاكل سوف نستخدم التخصيص الديناميكي للذاكرة. نوع المعطيات الإساسي المستخدم في مثل هذه البرامج هو نوع معين من السجل struct ، حيث ينتمي أحد حقول هذا السجل الى نوع المعطيات "مؤشر" أما باقي الحقول عادية حسب الطلب.

```
Struct elm
    int x;
   elm *next;
} ;
elm *p ;
```

هكذا نكون قد عرفنا على نوع معطيات جديد وهو السجل :

❖حقله الاول يدعى x وهو عدد صحيح ،

❖الحقل الثاني يدعى next وهو مؤشر يشير الى سجل آخر من نفس النوع elm .

❖و صرحنا عن p على أنه مؤشر وظيفته التأشير الى عناوين فيها فقط قيم من نوع ذلك السجل .

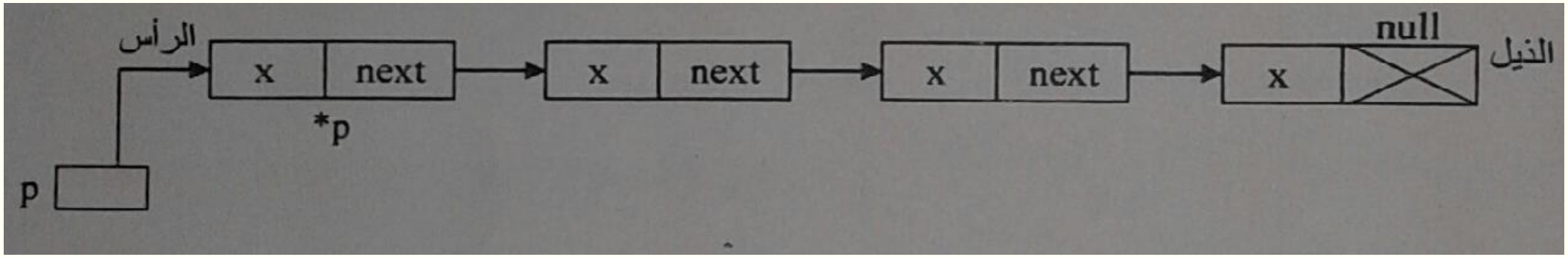
- اذاً p يؤشر الى p* حيث أن p* هو سجل (وذلك السجل بدوره مؤشر الى سجل .....).
- للوصول لحقول أي سجل نقوم بذكر اسم السجل أولاً ثم نقطة ثم اسم الحقل مثال :
  - للوصول لقيمة المتحول x نكتب p.x*
  - وللوصول للمؤشر next نكتب p.next*
- ويمكن الوصول لهذه الحقول عن طريق المؤشر P مثال:
  - للوصول لقيمة المتحول x نكتب p-> x
  - وللوصول للمؤشر next نكتب p->next

تعريف القائمة المترابطة

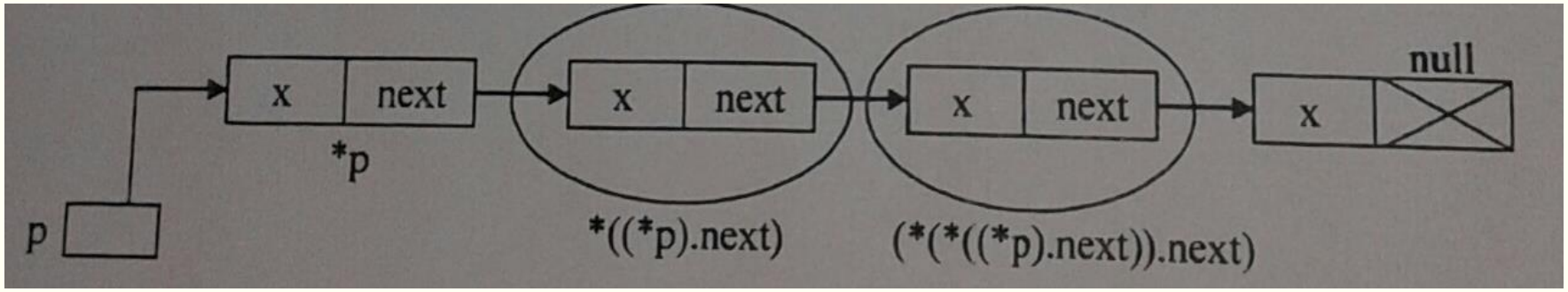
- بما أن كل سجل elm يستطيع أن يشير الى سجلelm آخر، فيمكننا بناء سلسلة طويلة من هذه البنى وذلك بربط هذه البنى بعضها ببعض بواسطة الحقل next .لذلك يمكننا تعريف اللائحة المترابطة **بأنها سلسلة من العناصر، رُبط فيها كل عنصر بالعنصر الذي يليه و هكذا....حيث يدعى كل عنصر من هذه السلسلة بعقدة node .**

- يشار الى العقدة الأولى في القائمة المترابطة بمؤشر يشير الى elm ، حيث أن هذا المؤشر عبارة عن مؤشر مستقل وليس حقلاً من السجل !!

- ان الحقل next الموجود في العقدة الأولى (أي في السجل الأول) سوف يشير الى العقدة الثانية ... أما الحقل next الموجود في العقدة الأخيرة من القائمة المترابطة فيجب أن يشير الى لا شيء أي يساوي null حتى يخبرنا عن انتهاء القائمة. تدعى العقدة الأولى " رأس السلسلة" أما العقدة الأخيرة "ذيل السلسلة ".

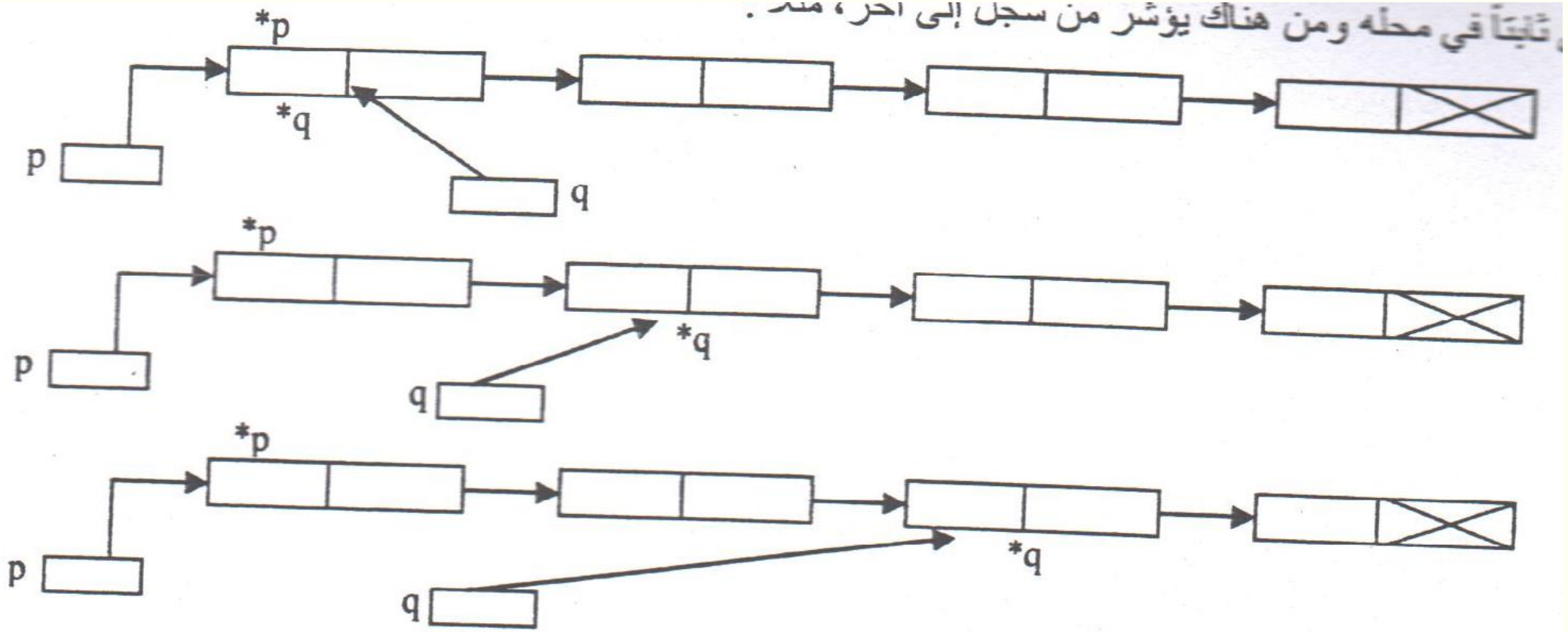

الرأس
x
next
x
next
x
next
x
null
الذيل
*p
p

- من أجل التعامل مع القائمة المترابطة يجب بالضرورة معرفة عنوان أول سجل من القائمة الذي يؤشر عليه مؤشر مستقل ما، أما بقية السجلات نوصل اليها على التتالي من سجل الى أخر حيث يلزمنا أحياناً مؤشرات مساعدة مستقلة للتنقل بين السجلات (عوضاً عن أن تقفز من سجل الى آخر) .
- يجب دوماً معرفة "من يؤشر و الى من"، مثلاً:



- **<u>هنا نلاحظ ما يلي :</u>**
- P يؤشر الى p * حيث p * عبارة عن سجل فيه الحقل next يؤشر الى السجل الثاني . ولكن next ليس متحولاً مستقلاً حتى نقول أنه يؤشر بل نقول أن:
- (*p).next يؤشر الى (*((*p).next)*.
- *(*((*p).next)).next)يؤشرالى(*((*p).next)).next
  اذاً القاعدة هي دوماً : المؤشر يؤشر الى محتوياته.
-
- نلاحظ أن الكتابة السابقة معقدة بعض الشيء ، خاصة اذا كان عدد السجلات كبيراً لذلك يفضل استخدام مؤشر مساعد و ليكن q حيث يبقى ثابتاً في محله و من هناك يؤشر من سجل الى آخر، **مثلاً:**

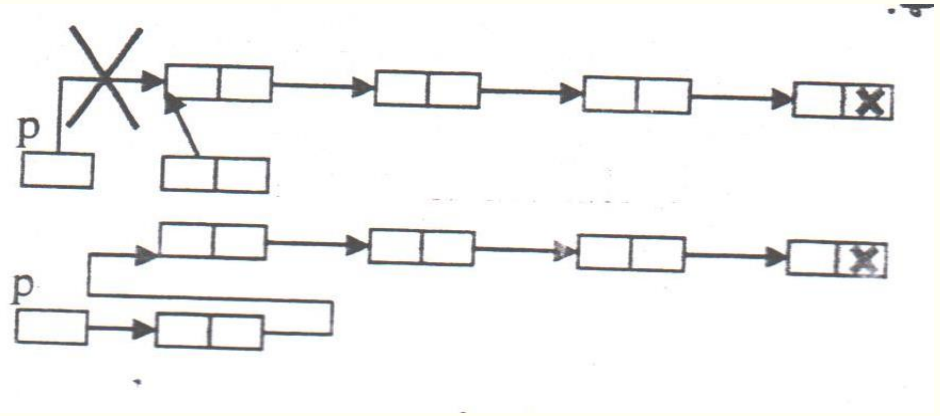
ثابتاً في محله ومن هناك يؤشر من سجل إلى آخر، مثلاً .



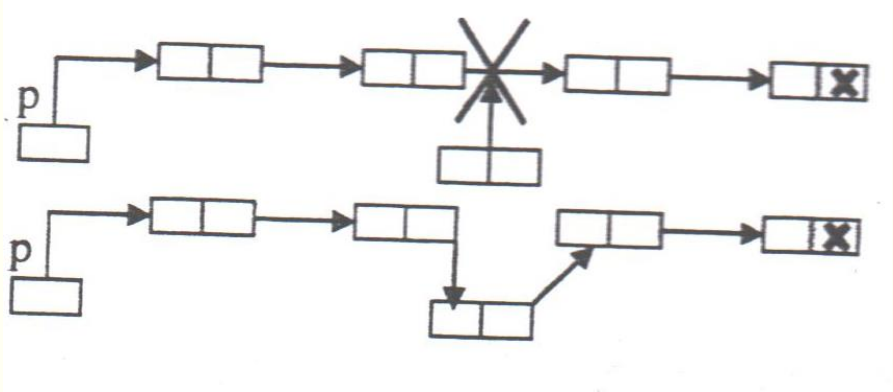
العمليات على القوائم المترابطة

❖ **الحشر:** نميز هنا ثلاث حالات

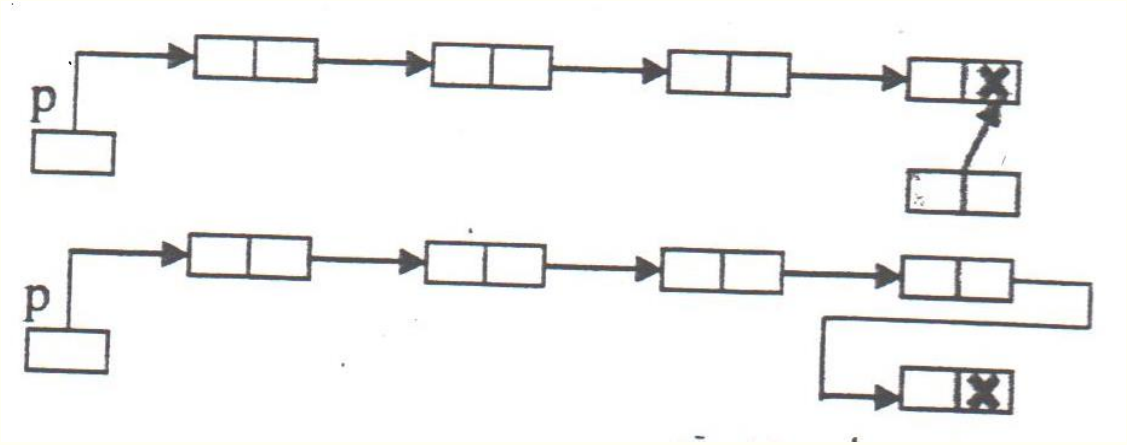
- حشر سجل في بداية سلسلة :



- حشر سجل في وسط السلسلة:



- حشر سجل في نهاية السلسلة :

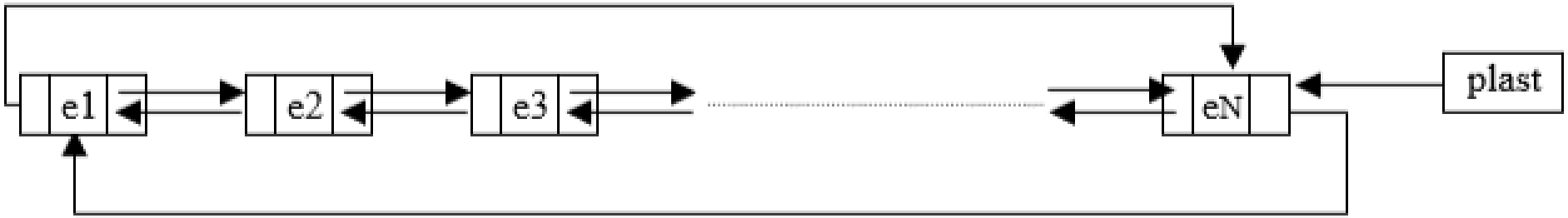


❖ **<u>الحذف:</u>** نميز حالتين و هما حذف أول سجل و حذف سجل من أي مكان آخر.

-

السلاسل الدائرية :

بدلاً من أن يأخذ المؤشر في آخر عنصر في السلسلة القيمة nil نجعل هذا المؤشر يؤشر إلى أول عنصر في السلسلة . و نحدد السلسلة بمؤشر إلى آخر عنصر فيها , كما في الشكل التالي :

السلاسل مضاعفة الارتباط :

نجعل فيها مؤشر العنصر الأخير أن يؤشر إلى أول عنصر , كما أننا نجعل مؤشر العنصر الأول أن يؤشر إلى آخر عنصر .