Pointers &LinkedList

م. سعيد سريحيني

RB Informatics; 21/11/2021
الخوارزميات

مقدمة

تعرفنا في المحاضرة السابقة عن العودية وكيفية عملها وحساب تعقيدها ونأتي اليوم لنستكفي مقررنا معاً عن بنية معطيات جديدة الا وهي LinkedList سنرى أهميتها وما الفرق بينها وبين المصفوفة ولكن قبل البدء بالحديث عنها سنتعلم عن المؤشرات وأهميتها واستخدامها ضمن هذه البنية.

- المؤشرات (pointers): وهي عبارة عن متحولات ذات نمط معين تحوي عنوان متغير آخر في الذاكرة ويمكن بواسطتها معرفة مكان تخزينه وتسمح لنا بالوصول إليه.

- في الـ C++ نُعرِّف المؤشر بالشكل التالي:

```
int x = 5 ;
int *y = &x ;
```

هنا الـ y تخزن الـ address لـ x في الذاكرة

- أما في الـ Java:

فإن كل object هو عبارة عن مؤشر يؤشر على مكان متحولاته التي يحتويها في الذاكرة.

```
Person p1;
```

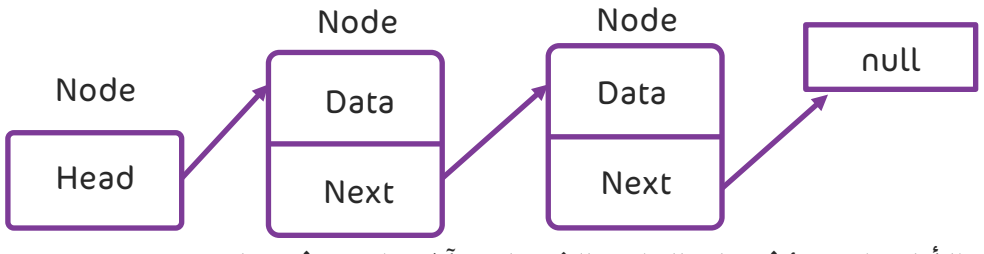
p1 هو مؤشر ولكن هنا لا يؤشر أي شيء لأن قيمته null

```
p1 = new Person();
```

هنا التعليمة new Person () تنشئ مكاناً في الذاكرة لهذا الـ object ويتغير عنوانه في الذاكرة إلى p1.

- الفكرة الأساسية من الـpointers هي تقليل حجم الذاكرة المستخدم فلا نريد عند تعريف الـobject أن نحجز مكان في الذاكرة له مباشرة لذلك تم إعتماد هذه الفكرة.

- linked list هي بنية معطيات خطية عناصرها لا تخزن بشكل متتابع في الذاكرة بل بشكل عشوائي وترتبط بمؤشرات ويكون كل عنصر عبارة عن عقدة تحوي قيمة ومؤشر يؤشر على العقدة التي تليها، العقدة الأولى تسمى head وتؤشر على العقدة الثانية والعقدة الأخيرة لا تؤشر على شيء او بالأحرى تؤشر على null.



هنا أول عنصر يؤشر على العنصر الذي يليه وآخر عنصر يشير على null

- ما الفرق بين الـ linked list والـ array؟

وجه المقارنة	Array	Linked list
Size	غير قابل للتعديل و الزيادة	قابل للتعديل والزيادة
كيفية التخزين	تحجز بشكل متتابع في الذاكرة	تحجز بشكل عشوائي في الذاكرة
access	نستطيع الوصول إلى أي عنصر بـ O(1)	للوصول إلى عنصر ما يجب المرور على كافة العناصر قبله
Insertion	لكي نضيف عنصر بين عنصرين علينا تحريك كل العناصر قبل الـ index الذي يليه ثم الإضافة O(n)	نستطيع إضافة عنصر بين عنصرين بـ O(1)

- خوارزمية الـ Binary search غير فعالة في حال كانت الـ linked list مرتبة لأنه كما قلنا أنَّه للوصول إلى عنصر ما يجب المرور على كافة العناصر قبله وهذا يؤدي إلى زيادة التعقيد بشكل كبير بل تصبح الـ linear search أكثر كفاءة.

- هنا يمكننا تعريف class وتمرير القيم فيه كي نغير الـ Data Type

هنا الـ Data هي القيمة التي ستخزن ضمن العقدة

Next هو المؤشر الذي سيؤشر على العقدة التالية

```
Class Node {
  int data;
  Node next;
  Node( int d ){
    data = d; } }
```

لطباعة عناصر الـ linked list:

سنبدأ بالمرور من الـ Head وطباعة ما تحويه

ثم الانتقال للعنصر التالي عبر المؤشر وطباعة ما يحتويه وهكذا...

```
void travers ( Node head ){
  Node p = head;
  while( p != null ){
  print(p.data);
  p = p.next;  } }
```

طباعة عناصر الـ linked list بشكل عكسي :

لكي نطبعها بشكل عكسي نستخدم العودية.

هنا في هذا التابع سيبقى يستدعي نفسه عودياً حتى تصبح قيمة الـ p تساوي null

عندها سيبدأ باستكمال عمله في كل استدعاء فيطبع الـ linked list بشكل عكسي.

```
void travers ( Node p ){
  if(p == null)
   return;
  travers (p.next);
  print(p.x); }
```

النهاية...

سنكون يوماً ما نُريد لا الرحلةُ ابتدأت

ولا الدرب انتهى.... ❤