بسم الله الرحمن الرحيم

مقدمة

الحمد لله الذي تقدست عن الأشباح ذاته، وتنزهت عن مشابهة الأمثال صفته، واحد لا من قلة، وموجود لا من علة، بالبر معروف، وبالإحسان موصوف، معروف بلا غاية، وموصوف بلا نهاية، أول بلا إبتداء وأخر بلا إنتهاء، لا تنسب إليه البنون ولا توهنه السنون، ولا تفنيه تداول الأوقات، كل المخلوقات تحت قهر عظمته، أمره بين الكاف والنون، بذكره أنس المخلصون، وبرؤيته تقر العيون، وبتوحيده أبتهج الموحدون، هدى أهل طاعته إلى صراط مستقيم، وأباح أهل محبته جنة النعيم، وعلم عدد أنفاس مخلوقاته في علمه القديم، جعل لكل شيئ قدرا، وأحاط بكل شيئ علما، وغفر ذنوب المذنبين كرما وحلما، ليس كمثله شيء وهو السميع العليم، والصلاة والسلام على سيد الأنام، مخرج البشرية من الظلام، وهاديهم إلى أفضل الطرق وأتمها على الدوام، محمد خير ولد عدنان، صل الله عليه وعلى أله وصحبه وسلم وعلينا معهم عدد ما أحاط به علمه وخطه قلمه إلى يوم الدين، وبعد

إن الله خلق الخلق في هذا الكون وجعل الإنسان سيد هذا الكون، والإنسان هو خليفة الله في أرضه، وهو المفضل على سائر المخلوقات، حيث منحه الله العقل وميزه بالعقل على سائر خلقه، وطالبه بعمارة الكون، وجعله قادرا على تسيير واستغلال كثير من سنن الكون التي منها تسير الجبال، وجري الأنهار على وجه الأرض الخ، وبما أن سنن هذا الكون تسير وفق نظام محكم دقيق، فلكي تستغل هذه السنن لا بد وأن يوجد نظام محكم دقيق يكون قادر على الاستفادة من هذه السنن، ولقد كان حدث اختراع الحاسوب من الأشياء العظيمة في تاريخ البشرية، ولكن الحدث الأعظم منه هو إستغلال الحاسوب في كل مجالات الحياة، منها الطب والهندسة والكيمياء .....الخ، حتى قيل أن الحاسوب أصبح اليوم مرتبط بكل العلوم بشقيها التجريبية والإنسانية، وذلك لإرتباط الحاسوب بالمعلومات، والمعلومات يحتاجها الإنسان في كل مجالات الحياة.

ولكي يستفيد الإنسان من قدرات الحاسوب، لا مناص من عمل مخطط للحاسوب (خارطة عمل) كي يسير وفقا له، هذا المخطط عبارة عن مجموعة من الخطوات أو الأوامر التي من خلالها يتم إنجاز المهمة المطلوبة من الحاسوب إنجازها، فإذا كانت الخطة المرسومة محكمة، فإن الهدف المنشود سوف يتحقق بإذن الله، أما إذا كانت الخطة غير محكمة، فإن الهدف المنشود من هذه الخطة لن يتحقق، وكل مخطط يحتاج إلى معلومات خاصة، هذه المعلومات تختلف من نشاط لأخر ومن مؤسسة لأخرى، وبالتالي لا بد من هياكل تفصل بين المعلومات كي يتم الوصول إليها بسهولة ويسر.

أهمية هياكل البيانات

إن هذا العصر هو عصر المعلومات، ويتم الحصول على المعلومات من معالجة البيانات، ولكن البيانات التي تعتبر المواد الأولية للمعلومات تختلف من نشاط لأخر ومن مؤسسة لأخرى، وبالتالي لا بد من الفصل بين البيانات في هياكل كي يتم الوصول إليها بسهولة ويسر، وبأقل تكاليف، مثل التعامل مع الطابور في عمليات الحذف أو الإضافة، ولذالك تكمن أهمية كتاب هياكل البيانات الذي سوف نحاول فيه ان نوضح وضع المعلومات أو البيانات في هياكل تساعد المستخدم للوصول إليها بسهولة ويسر، وتمكنه من معالجتها بطريقة ما، مثل التعامل مع الطابور والمكدس الخ.

إن بناء هياكل البيانات Data Structure هو الموضوع الرئيسي لهذا الكتاب الذي سوف نحاول فيه شرح وتبسيط موضوع هياكل البيانات Data Structure، و التي تسمى أحيانا تراكيب البيانات، وذالك لما له من أهمية قصوى، وخاصة في الواقع العملي وفي تنمية القدرات البرمجية.

المتطلبات السابقة للإستفادة من هذا الكتاب

مطلوب منك أيها القار الكريم أن تكون ملما بالبرمجة بلغة C++ ولديك القدرة على التعامل مع لغة C++ ، كما أنه مطلوب منك أن تكون لديك همة عالية لكي تستفيد من هذا الكتاب.

هذا ونسأل الله العلي العظيم أن يسدد خطانا و يغفر زلاتنا وأن يوفقنا لما فيه الخير والنجاح.

الفقير إلى عفو ربه

د / صالح نعمان عبدالله العسلي

المراجع التي يمكن أن يستعين بها الطالب:

1- Data Structure using C & C ++
2- Data Structure, Algorithms, and Applications in C++
3- Algorithm and data structure

الهدف العام من هذا الكتاب

الهدف العام من هذا الكتاب هو جعل الطالب قادر على الأتي:

1. **معرفة أنواع مختلفة من كهياكل البيانات**
2. **حجز و تحرير مواقع في الذاكرة أثناء التنفيذ**
3. **تكوين وإستخدام أنواع مختلفة من كهياكل البيانات**
4. **تحليل أنواع مختلفة من كهياكل البيانات**

وسوف يتم الوصول إلى الهدف العام بإذن الله عن طريق مجموعة من الأهداف موجودة في بداية كل وحدة من هذا الكتاب.

محتوى الكتاب

يحتوي هذا الكتاب على الأتي:

معرفة وبناء بعض الهياكل الأساسية للبيانات Data Structure مثل (الجداول) المصفوفات **الأحادية والثنائية، المؤشرات ، الدوال ، قضايا الترتيب والبحث عن العناصر، بالإضافة إلى**

, Structures, dynamic memory allocation linked lists, Stacks, queues, circular linked lists, double linked lists and binary trees.

الجزأ الأول

الفصل الأول

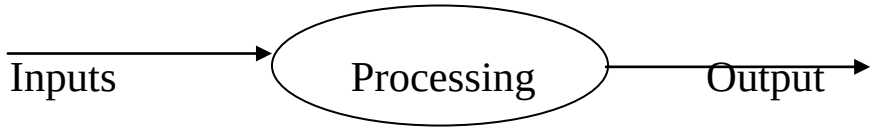
مقدمة

سوف ندرس في هذا الجزأ هيكلة البيانات Data Structure، و التي تسمى أحيانا تراكيب البيانات، وذالك لما لها من أهمية قصوى وخاصة في الواقع العملي وفي تنمية القدرات في البرمجة.

في البداية سوف نحاول تعريف عنوان دراستنا (هياكل البيانات) أو تراكيب البيانات.

البيانات هي المواد الأولية التي عند معالجتها بطريقة ما يتم الحصول على المعلومات، أي أن البيانات تكون في صورتها الأولية قلية الفائدة، وعند معالجتها قد تحصل الفائدة المرجوة منها.

إن البيانات قد تكون بالصورة الأولية، وقد تكون مركبة، وعند ما تكون مركبة فقد تكون أولية لبعض و غير أولية لبعض آخر، أي أن البيانات من حيث التركيب والبساطة هي عملية نسبية، فعلى سبيل المثال عندما نتعامل مع الأعداد الصحيحة نقول integer يكون هذا التعريف بسيط للمبرمجين بلغة C++ ولكنه مركب عند المبرمجين بلغة التجميع assembly language أو لغة الالة Machine language، وعلى كل حال فالذي يهمنا قوله هنا، أن البيانات قد يحصل لها نوع من المعالجة، عند إذ نحصل منها على معلومات.



ولكن هذه المعلومات ممكن أن تكون مبعثرة أو غير مرتبة و يصعب الوصول إليها، وبالتالي تكون عديمة الفائدة أو قليلة الفائدة، وفي عصرنا الحاضر عصر المعلومات، يجب أن تكون المعلومات في وضع ما يمكن من الوصول إليها بسهولة ويسر، وبأقل تكاليف، من حيث الوقت والجهد.

أما كلمة هياكل، فهي جمع هيكل وهو التركيب وفق خطة، والمقصود هنا وضع المعلومات في هيكل على النحو الملائم.

إن وضع المعلومات أو البيانات في هيكل على شكل طابور، على سبيل المثال، سوف يسهل التعامل مع الطابور في عمليات الحذف أو الإضافة، و يقلل من عمليات البحث عن المعلومة إذا كانت تخضع لنظام الطابور المتمثل في first in first out FIFO.

وعلى هذا، فإن شاء الله سوف نقوم بدراسة، وتوضيح، وبناء بعض الهياكل الأساسية، التي من خلالها يمكن أن تحفظ بعض المعلومات، بحيث يسهل التعامل معها، ومن هذه الهياكل على سبيل المثال: الجداول(المصفوفات) arrays , Structures, linked lists, Stacks, queues, trees, الخ.

ونسأل الله أن يوفقنا لما فيه الخير والسداد إنه على ما يشاء قدير.

الباب الأول المصفوفات الأحادية والثنائية

الهدف من هذا الباب هو الأتي:

القدرة على تكوين الجداول / المصفوفات الأحادية والثنائية

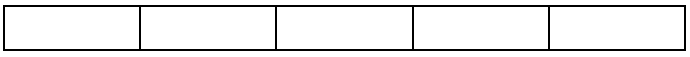
القدرة على إستخدام الجداول / المصفوفات الأحادية والثنائية كهياكل للبيانات

1 الجداول / المصفوفات وتسمى Tables / Matrixes/Arrays

تعتبر الجداول أو Arrays من أسهل هياكل البيانات بناء ومعالجة، ويوجد منها نوعين:

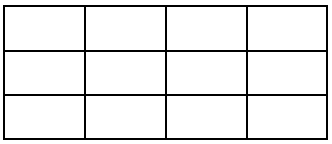
1. 1 الجداول الأحادية One dimension Arrays وهي التي تتكون من صف واحد وعدة أعمدة، أو من عمود واحد وعدة صفوف كما في الشكل 1 .1

شكل a. 1 .1عمود واحد وأربعة صفوف



شكل 1.b 1. صف واحد وخمسة أعمدة

1. 2 الجداول الثنائية Two dimensions Arrays وهي التي تتكون من عدة صفوف وعدة أعمدة: كما في الشكل 1.2 ثلاثة صفوف وأربعة أعمدة



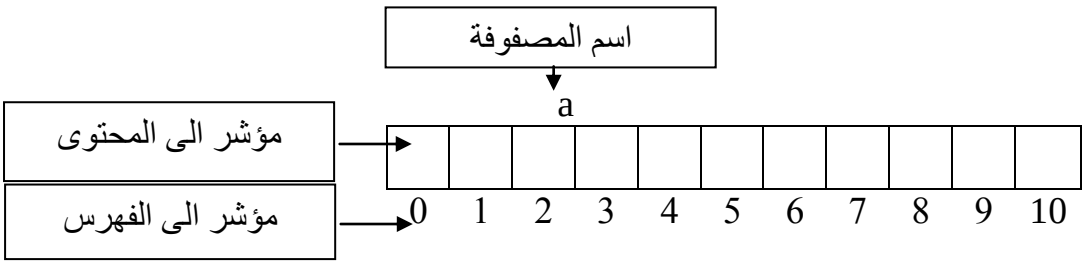
شكل 1.2

1. 3 تعريف الجدول (المصفوفة):

الجدول (المصفوفة) عبارة عن مجموعة محدودة من مواقع متجاورة في ذاكرة الحاسوب، يمكن أن يحتوين على مجموعة محدودة من البيانات المتشابهة في النوع.

المواقع المتجاورة تسمى خانات الجدول أو خانات المصفوفة، كل تلك المواقع أو الخانات يحملن اسما واحدا، هو اسم الجدول أو المصفوفة، و يحدد ذالك الإسم من قبل المبرمج.

كل جدول أو مصفوفة يتكون من خلايا يمكن أن يحتوين على بيانات متشابهة، كما أن كل جدول يحتوي على مجموعة محدودة من الفهارس تساوي عدد خلايا الجدول، كما هو موضح في الشكل 1.3.



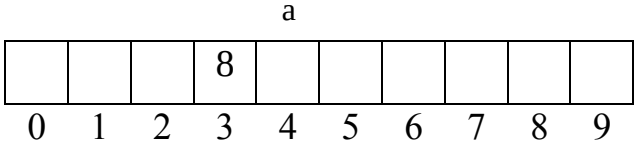
شكل 1.3 يوضح المواقع والفهارس لخلايا الجدول

1. 3.1 . التعامل مع الجداول (المصفوفات) الأحادية

للتعامل مع الجداول (المصفوفات) الأحادية، يكفي أن يتم ذكر اسم الجدول (المصفوفة) و رقم العمود فقط إذا كانت مكونة من صف واحد وعدة أعمدة، أو ذكر اسم الجدول (المصفوفة) و ذكر رقم الصف فقط، إذا كانت مكونة من عمود واحد وعدة صفوف.

في الشكل 1.3، الجدول (المصفوفة) a مكون من صف وأحدى عشر عمودا، فإذا أردنا أن نضع القيمة 8 في الخانة رقم 3، فإننا نذكر اسم الجدول (المصفوفة) ورقم الخانة على النحو ;8 =[3]a

(إذ أن ذكر اسم الجدول (المصفوفة) فقط لا يكفي لإستخدامه) ثم نضع فيه ما نشاء، كما في الشكل 1.4



شكل 1.4

عند التعامل مع الجداول يجب أن ننبه إلى الأتي:

1) هذه الجداول تحمل بيانات من نوع واحد، أي أن محتوياتها كلها يجب أن تكون متشابهة من حيث النوع، كما هو واضح من التعريف السابق.
2) حجم الجدول (المصفوفة) يجب أن يحدد في البداية عند تعريفه، كما أن الفراغات الخاص بالجدول و التي لم تستخدم بعد، تضل محجوزة طوال فترة التنفيذ، سواء تم إستخدام هذه الفراغات من قبل المستخدم، أم لم يتم إستخدامهن.
3) لا يستطيع المستخدم تجاوز الحجم المحدود للجدول.
4) خانات الجدول (المصفوفة) يحجز لها مواقع متجاورة في الذاكرة من قبل الCompiler ونظام التشغيل، حسب السعة المطلوبة، وأن سعة كل موقع تعتمد على نوع المعلومات التي حجزت من أجله.

فمثلاً: char < int< float< double

4. 1 التعامل مع الجداول (المصفوفة) الأحادية بلغة C++

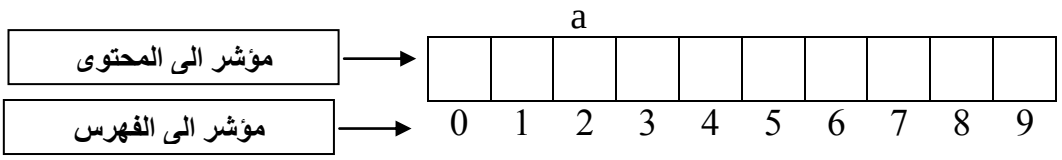
لكي يتم تعريف الجدول (المصفوفة) الأحادية بلغة C++ نحتاج الأتي

Type variable-name [size];

حيث type هو نوع البيانات التي سوف تحفظ في الجدول (المصفوفة)، أما variable-name فهو اسم الجدول (المصفوفة)، أما Size هو حجم الجدول (المصفوفة) المراد حجزه.

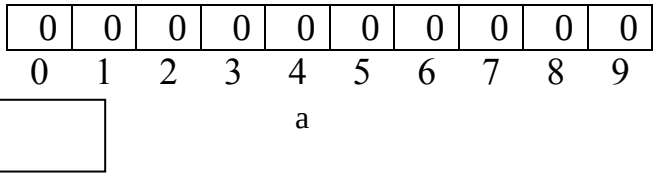
ونود أن ننبه هنا أن الحجم يجب أن يحدد مسبقاً وأن يكون عددا صحيحا، وأن النظام يعطي الرقم صفر لأول خانة يتم حجزها.

مثال1 : اكتب بلغة C++ التعريف لتكوين جدول يتكون من صف واحد وعشرة أعمدة يحتوى على معلومات من نوع int : فعلى ذالك يكون التعريف بلغة C أو C++ هو على النحو ; int a[10]



مثال2 فإذا طلب منا أن نضع في كل خانات هذا الجدول (المصفوفة) القيمة صفرا، فما علينا إلا كتابة الأتي:

```
التعريف  int a[10],I;
for(I=0;I<10;I++)
  a [I]= 0;
```

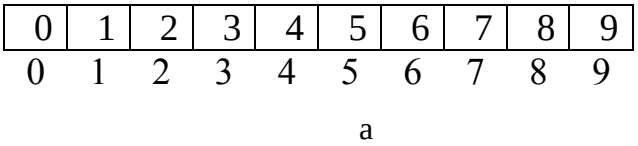


هذه الحلقة سوف تضع القيمة صفرا في كل خانات الجدول (المصفوفة) a.

مثال3 فإذا أردنا أن نضع رقم الخانة كقيمة في الخانة نفسها فيكون الحل هو

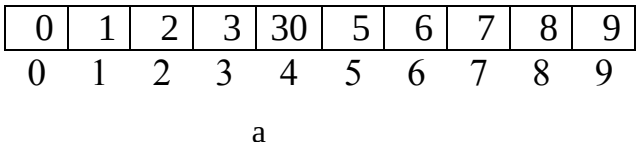
```
for(I=0;I<10;I++)
  a [I]= I;
```

فبعد تنفيذ الحلقة (for loop) يكون الجدول (المصفوفة) على هذا الشكل.



ولكن بعد ذالك، إذا أردنا وضع 30 في الخانة الرابعة من الجدول (المصفوفة) a فنكتب هذا الأمر:

a[4] = 30;



ومن هذا يظهر أنه بالإمكان التعامل مع خانات الجدول (المصفوفة) واحدةً واحدةً، ولكن هذه الطريقة غير مجدية، خاصة عندما يكون طول الجدول (المصفوفة) كبير، ولهذا نلجأ إلى إستخدام الحلقات loops للتعامل مع الجداول كما رأينا سابقاً، طبعا هذا إذا أردنا التعامل مع عدد كبير من خانات الجدول (المصفوفة)، أما إذا كان العدد قليلا فإن التعامل المباشر يكون أفضل.

وعلى ذالك فنلاحظ أن إستخدام الجداول سريع وسهل، ويحتاج الجدول الأحادي إلى loop واحد للتعامل مع كل خاناته.

مثال4

اكتب برنامجا يعرف مصفوفة أحادية مكونة من 10صفوف من نوع int ثم يضع 30 في الخانة الأولى من المصفوفة، ثم يزيد 10 في كل خانة، ثم يطبع محتوى المصفوفة.

الحل: يتم كتابة البرنامج التالي :

```
main( )
{
  int  a[10],I,k=30;//      التعريف
  for(I=0;I<10;I++)
  {
    a [I]= k;
     cout<<a[i]<<" ";
      k = k+10;
   }
 return 0;
}
```

فبعد تنفيذ البرنامج السابق، يتم تكوين مصفوفة ويكون محتوى المصفوفة في الذاكرة على شكل 1.5.

a

30	40	50	60	70	80	90	100	110	120
0	1	2	3	4	5	6	7	8	9

شكل 1.5

1.4. 1 طباعة محتوى الجدول (المصفوفة)

لطباعة محتوى جدول أو مصفوفة ما، فيتم طباعة محتوى كل خانة من خانات الجدول على حدة (خانة خانة).

فمثلا إذا أردنا طباعة محتوى الجدول (المصفوفة) a شكل 1.5 **ابتداء من الخانة الأولى إلى النهاية، فيتم كتابة هذه الحلقة (for loop) وسط البرنامج.**

a

30	40	50	60	70	80	90	100	110	120
0	1	2	3	4	5	6	7	8	9

```
for(I=0;I<10;I++)
 cout<<a[I];
```

Iمؤشرا الى index

فسوف يظهر لنا محتوى الخانات فقط على هذا النحو شكل 1.6:

30 40 50 60 70 80 90 100 110 120

شكل 1.6

أما إذا أردنا طباعة محتوى الجدول (المصفوفة) a من الخانة الرابعة إلى النهاية فيتم كتابة هذه الحلقة (for loop) وسط البرنامج.

```
for(I=3;I<10;I++)
 cout<<a[I];
```

مؤشر الى الخانة الرابعة

فسوف يظهر لنا شكل 1.7

60 70 80 90 100 110 120

شكل 1.7

ولكن إذا أردنا طباعة محتوى الجدول a من الخانة الاخيرة إلى الخانة الأولى فيتم كتابة هذه الحلقة (for loop) وسط البرنامج.

```
for(I=9;I>-1;I--)
 cout<<a[I];
```

مؤشر الى الخانة الاخيرة

فسوف يظهر لنا شكل 1.8

120 110 100 90 80 70 60 50 40 30

شكل 1.8

وإذا أردنا وضع القيمة 30 في كل الخانات الفردية من المصفوفة السابقة a فيتم كتابة التالي:

30	40	50	60	70	80	90	100	110	120
0	1	2	3	4	5	6	7	8	9

```
for(I=1;I<10;I+=2)
 a[I]=30;
```

فتكون النتيجة على النحو التالي شكل 1.9

30	30	50	30	70	30	90	30	110	30
0	1	2	3	4	5	6	7	8	9

شكل 1.9

ونلاحظ أن التغير حصل فقط على محتوى الخانات الفردية من المصفوفة، أما محتوى الخانات الزوجية من المصفوفة a فلم يحصل لها أي تغير.

و إذا أردنا طباعة محتوى الخانات الزوجية فقط من الجدول (المصفوفة) a شكل 1.9

من الخانة الأولى إلى النهاية فيتم كتابة هذه الحلقة (for loop) وسط البرنامج.

```
for(I=0;I<10;I=I+2)
 cout<<a[I];
```

معدل الزيادة للمتغير I

فسوف يظهر لنا شكل 1.10

30 50 70 90 110

شكل 1.10

وكذالك إذا أردنا ايضا طباعة محتوى الخانات الفردية فقط من الجدول (المصفوفة) شكل 1.9

من الخانة الأولى إلى النهاية فيتم كتابة هذه الحلقة (for loop) وسط البرنامج.

```
for(I=1;I<10;I=I+2)
 cout<<a[I];
```

فسوف يظهر لنا شكل 1.11 أي ان معدل

الزيادة للمتغير I بمقدار اثنين

30 30 30 30 30

شكل 1.11

مثال5

اكتب برنامجا يعرف مصفوفة أحادية مكونة من 10صفوف من نوع int ثم يضع 30 في الخانة الأولى من المصفوفة، ثم يزيد 10 في كل خانة، ثم يطبع محتوى المصفوفة بنفس الترتيب، و يطبع محتوى المصفوفة ايضا بعكس الترتيب.

الحل: يتم كتابة البرنامج التالي :

```
#include<iostream.h>
main( )
{
  int  a[10],I,k=30;//      التعريف
  for(I=0;I<10;I++)
  {
    a [I]= k;
     cout<<a[i]<<" ";
      k = k+10;
   }
 cout<<endl;
for(I=9;I>-1;I--)
  cout<<a[i]<<" ";
 return 0;
}
```

لطباعة محتوى المصفوفة بنفس الترتيب.

لطباعة محتوى المصفوفة بعكس الترتيب.

فبعد تنفيذ البرنامج السابق يتم تكوين مصفوفة في الذاكرة ويكون محتوى المصفوفة كما في شكل 1.12.

a

30	40	50	60	70	80	90	100	110	120
0	1	2	3	4	5	6	7	8	9

شكل 1.12.a

مثال6

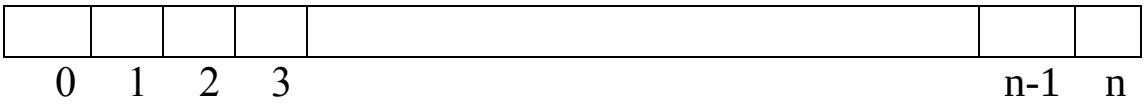
اكتب برنامجا يعرف مصفوفة أحادية مكونة من 20 موقعا من نوع int ثم يضع 55 في الخانة الأولى من المصفوفة، ثم يزيد 5 في كل خانة، ثم يطبع محتوى المصفوفة بنفس الترتيب، كل 5 عناصر في صف، و يطبع ايضا محتوى المصفوفة بعكس الترتيب كل 6 عناصر في صف.

الحل: يتم كتابة البرنامج التالي :

```
#include<iostream.h>
main( )
{
  int  a[20],I,k=55,C=0//      التعريف
  for(I=0;I<20;I++)
  {
    a [I]= k;
     cout<<a[i]<<" ";
      k = k+5;
   if(( i+1)%5 ==0)
     cout<<endl;
    }
 cout<<endl;
for(I=19;I>-1;I--)
{
 C++;
  cout<<a[i]<<" ";
  if( C%6 ==0)
     cout<<endl;
 }
 return 0;
}
```

2.4.1 تمثيل الجداول الأحادية في الذاكرة.

تمثل الجداول الأحادية في الذاكرة سواء كانت مكونة من عدة أعمدة وصفاً واحداً، أو مكونة من عدة صفوف وعموداً واحداً، شكل 1.12 يوضح ذالك.:



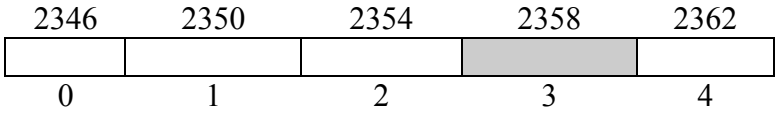
شكل 1.12

حيث يتم تخصيص الخانة الأخيرة، في معظم الأنظمة، لتحتوي على عدد الصفوف المكونة للجدول، وعادة تحتوي على بداية ونهاية أرقام الخانات، أما بقية الصفوف فإنها تحتوي على بيانات الجدول (المصفوفة).

عند التعامل مع الجدول (المصفوفة) فإن مترجم النظام يضع مؤشرا إلى عنوان أول خانة مرفقاً مع اسم الجدول (المصفوفة) ثم يزيد هذا العنوان (عنوان أول خانة) بمقدار 2bytes إذا كان نوع البيانات char أو 4bytes إذا كان نوع البيانات int وهكذا.

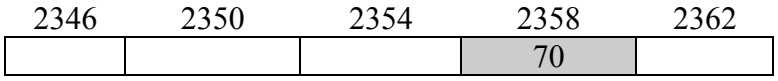
فإذا فرضنا أن لدينا جدول مكون من 5 خانات واسمه a وهو من نوع int ;[5] int a

وكان عنوان أو خانة في هذا الجدول (المصفوفة) 2346 فيكون عنوان الخانة الثانية هو 2350 ، شكل 1.13 يوضح ذالك.:



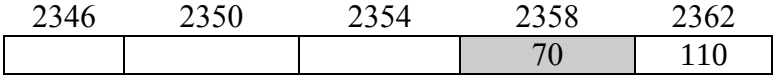
شكل 1.13

فإذا نفذنا هذا الأمر: ;70 =[3]a فإن القيمة 70 سوف تأخذ مجراها إلى الخانة الثالثة وسوف تستقر هناك، شكل 1.14 يوضح ذالك.:



شكل 1.14

فإذا نفذنا هذا الأمر: ;110 =[4]a فإن القيمة 110 سوف تأخذ مجراها إلى الخانة الرابعة وسوف تستقر هناك. شكل 1.15 يوضح ذالك.:



شكل 1.15

3.4. 1 أمثلة على المصفوفات الأحادية

س$_1$ - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقأما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في صف) ثم يطبع في صف جديد قيمة وموقع أصغر عناصر المصفوفة.

الحل

```
#include<iostream.h>
#include<stdlib.h>
int main ( )
 {
  int a[100],i,p=0 , min;
  a[0]=rand()%1000;
  min=a[0];
  cout<<a[i]<<”  “;
  for(i=1;i<100;i++)
  {
    a[i]=rand()%1000;
    cout<<a[i]<<”  “;
    if(a[i]< min)
    {
     min= a[i];
     p=i;
    }
         if((i+1)%8==0)
        cout<<endl;
 }
        cout<<endl<<" the minimum is “<<min<<"  the positon   is “<<p;
                return 0;
  }
```

س$_2$ - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 150 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقأما عشوائية، ثم يطبع محتويات المصفوفة (كل 6 عناصر في صف) ثم يطبع في صف جديد قيمة وموقع أكبر عناصر المصفوفة.

الحل

```
#include<iostream.h>
#include<stdlib.h>
int main ( )
```

```
{
 int a[150],i,p=0 , max;
 a[0]=rand()%1000;
 max=a[0];
 cout<<a[i]<<"  ";
 for(i=1;i<150;i++)
 {
   a[i]=rand()%1000;
   cout<<a[i]<<"  ";
   if(a[i]> max)
   {
    max = a[i];
    p=i;
   }
        if((i+1)%6==0)
       cout<<endl;
}
       cout<<endl<<" the maximum is "<<max <<"  the positon   is "<<p;
              return 0;
 }
```

س3 - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً ، من نوع int ، ثم يولد في هذه المصفوفة أرقأما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في صف) ثم يطبع في صف جديد قيمة وموقع ثاني أصغر عنصر في المصفوفة.

الحل

```
#include<iostream.h>
#include<stdlib.h>
int main ( )
 {
  int a[100],i,p1,p2=0 , m1,m2;
  a[0]=rand()%1000;
  a[1]=rand()%1000;
 if(a[0]< a[1])
    {
     m1= a[0]; p1=0;
     m2= a[1]  ; p2=1;
    }
 else
{
     m1= a[1]; p1=1;
```

```
    m2= a[0]  ; p2=0;
   }
   cout<<a[0]<<" "<<a[1]<<" ";
 for(i=2;i<100;i++)
 {
  a[i]=rand()%1000;
  cout<<a[i]<<" ";
  if(a[i]< m1)
  {
      m2= m1  ; p2=p1;
     m1= a[i]; p1=i;
  }
 else
   if(a[i]< m2)
    {
      m2= a[i];   ; p2=i;
     }
       if((i+1)%8==0)
       cout<<endl;
 }
       cout<<endl<<" the second minimum is "<<m2<<"  the positon   is "<<p2;
 return 0;
 }
```

س4 - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً ، من نوع int ، ثم يولد في هذه المصفوفة أرقأما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في صف) ثم يطبع محتويات المصفوفة بعكس الترتيب (كل 7 عناصر في صف).

الحل

```
#include<iostream.h>
#include<stdlib.h>
int main ( )
 {
  int a[100],i,c=0;
  for(i=0;i<100;i++)
  {
    a[i]=rand()%1000;
    cout<<a[i]<<" ";
    if((i+1)%8==0)
         cout<<endl;
  }
for(i=99;i>-1;i--)
  {
```

```
   cout<<a[i]<<" ";
   if((c+1)%7==0)
        cout<<endl;
   c++;
  }
return 0;
  }
```

س5 - اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 150 موقعاً ، من نوع int ، ثم يولد في الموقع الأول و الموقع الثاني من هذه المصفوفة أرقأما عشوائية، ثم يضع في الموقع الثالث مجموع محتوى الموقع الأول و الموقع الثاني، وهكذا في كل مواقع المصفوفة، ثم يطبع محتويات المصفوفة (كل 6 عناصر في صف).

الحل

```
#include<iostream.h>
#include<stdlib.h>
int main ( )
 {
  int a[150],i,sum=0;
  for(i=0;i<100;i=i+2)
  {
    a[i]=rand()%1000;
    cout<<a[i]<<" ";
    a[i+1]= rand()%1000;
    cout<<a[i+1]<<" ";
    sum= a[i] + a[i+1];
   cout<<sum<<" ";
   if((i+1)%4==0)
        cout<<endl;
 }
        return 0;
  }
```

اسئلة

س$_1$ - اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 180 موقعاً ، من نوع int ، ثم يولد في الموقع الأول و الموقع الثاني من هذه المصفوفة أرقاما عشوائية، ثم يضع في الموقع الثالث مجموع محتوى الموقع الأول و الموقع الثاني، وهكذا في كل مواقع المصفوفة، ثم يطبع محتويات المصفوفة (كل 9 عناصر في صف).

س$_2$ – مستخدما الدوال: اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 200 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في صف) ثم يستدعى دالة لطباعة محتوى المصفوفة الأعداد الزوجية والفردية كل على حدة (كل 7 عناصر في صف).

س$_3$– مستخدما الدوال: اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 200 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يستدعى دالة لطباعة محتويات المصفوفة (كل 8 عناصر في صف) ثم يستدعى دالة للطباعة في سطر جديد عدد الأعداد الزوجية والفردية في المصفوفة.

س$_4$ – مستخدما الدوال: اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 200 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في صف) ثم يستدعى دالة لطباعة محتوى المصفوفة مفروزة بناء على قيمة الخانة الاخيرة من جهة اليمين. (كل 7 عناصر في صف)

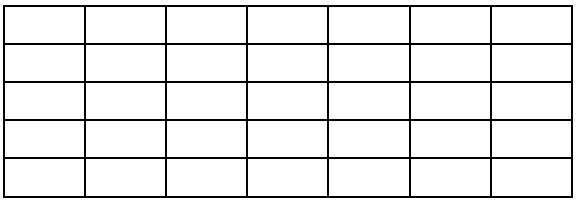
س$_5$– مستخدما الدوال: اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 150 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يستدعى دالة الطباعة لطباعة محتويات المصفوفة (كل 7 عناصر في صف) ثم يستدعى دالة لترتيب عناصر المصفوفة تصاعديا، ثم يستدعى دالة الطباعة لطباعة محتوى المصفوفة (كل 7 عناصر في صف)، ثم يستدعى دالة لترتيب عناصر المصفوفة تنازليا، ثم يستدعى دالة الطباعة لطباعة محتوى المصفوفة (كل 7 عناصر في صف).

س$_6$ – مستخدما الدوال: (بدون إستخدام if) اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 160 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يستدعى دالة لطباعة محتوى المصفوفة (كل 8 عناصر في صف).

س$_7$ – مستخدما الدوال: اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 50 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يستدعى دالة الطباعة لطباعة محتويات المصفوفة (كل 5 عناصر في صف) ثم يستدعى دالة لتزيح عناصر المصفوفة دائرا نحو اليسار بمقدار 5، ثم يستدعى دالة الطباعة لطباعة محتوى المصفوفة (كل 5 عناصر في صف).

1. 5 الجداول أو المصفوفات الثنائية (ثنائية الأبعاد)

تم تعريف الجداول الثنائية بأنها التي تتكون من أكثر من صف وأكثر من عمود. والجداول الثنائية لها نفس خواص الجداول الأحادية من حيث أن محتوياتها يجب أن تتشابه من حيث النوع، ويجب أن يحدد مسبقاً حجمها(عدد صفوفها، وعدد أعمدتها) ونوعها. . والشكل 1.16 يوضح ذالك.:



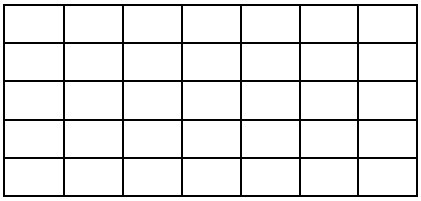
شكل 1.16 جدول ثنائي يتكون من خمسة صفوف وسبعة

1. 5 .1 التعامل مع الجداول الثنائية

تعرف هذه الجداول الثنائية في لغة C++ على النحو التالي: ;Data-type table-name[size1][size2] حيث كلمة Data-type تعني نوع البيانات في الجدول (المصفوفة) و كلمة table-name تعني اسم الجدول (المصفوفة)، و size1 هو عدد الصفوف في الجدول (المصفوفة) ،ولا بد أن يكون عددا صحيحا، size2 هو عدد الأعمدة في الجدول (المصفوفة)، و يكون عددا صحيحا أيضاً.

فإذا أردنا تعريف الجدول (المصفوفة) ، والشكل 1.17 بلغة C++ فيكون التعريف هو :;int a[5][7]

في هذا التعريف: اسم المصفوفة a، ونوع البيانات التي سوف تحتجزها int، وعدد صفوفها 5، وعدد أعمدتها 7.

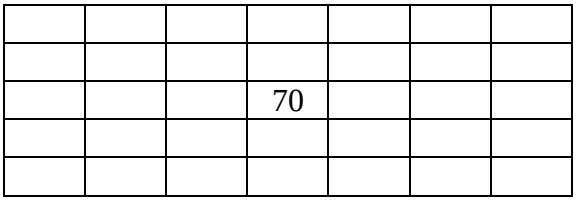


شكل 1.17 جدول ثنائي يتكون من خمسة صفوف وسبعة

1. 5 .2 وضع قيم في الجداول الثنائية

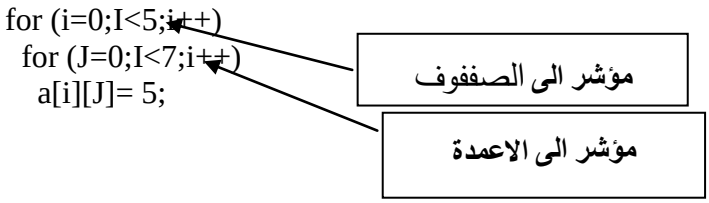
فإذا أردنا وضع القيمة 70 في الصف الثالث العمود الرابع من خانات الجدول (المصفوفة) a شكل 1.18 فيتم كتابة الأمر:

a[2][3] = 70;



شكل 1.18 جدول ثنائي يتكون من خمسة صفوف وسبعة

وإذا أردنا أن نضع في كل خانات الجدول (المصفوفة) a القيمة 5، فيتم كتابة الأوامر التالية:



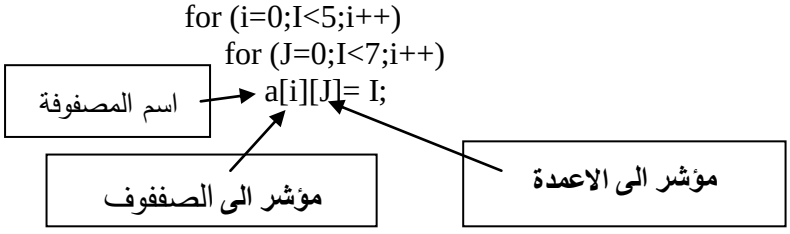
عند تنفيذ الأوامر السابقة، فإن قيم خانات الجدول (المصفوفة) شكل 1.19 كلهن سوف يحملن القيمة 5.

5	5	5	5	5	5	5
5	5	5	5	5	5	5
5	5	5	5	5	5	5
5	5	5	5	5	5	5
5	5	5	5	5	5	5

شكل 1.19 جدول ثنائي يتكون من خمسة صفوف وسبعة

من هنا يتضح أن معالجة الجدول (المصفوفة) الثنائية لا بد وأن يشار إلى كل عنصر من عناصر المصفوفة عن طريق ذكر اسم المصفوفة ثم ذكر الصف و العمود، ولا بد وأن تحتوي على (two loops) حلقتين، كي تغطى كل خانات الجدول (المصفوفة)، ويكون مدى كل حلقة هو إما عدد الصفوف أو عدد الأعمدة للجدول.

وإذا أردنا أن نضع في كل صف من صفوف الجدول (المصفوفة السابقة) a قيمة الصف، فيتم كتابة الأوامر التالية:



عند تنفيذ الأوامر السابقة، فإن قيم خانات كل صف من صفوف الجدول (المصفوفة) شكل 1.20 سوف يحملن قيمة الصف.

0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4

شكل 1.20 جدول ثنائي يتكون من خمسة صفوف وسبعة أعمدة

1. 5 .3 امثلة على إستخدام الجدول (المصفوفة) الثنائية:

مثال 1: اكتب برنامجاً يعرف جدولاً ثنائياً a من نوع int، يتكون من خمسة صفوف وسبعة أعمدة، ثم يولد فيه قيم عشوائية ما عدى الصف الأخير، فيضع في كل موقع منه مجموع عناصر العمود الذى فيه.

الحل

50	61	69	13	6	8	10
80	18	31	87	82	75	15
56	89	45	97	54	12	5
97	25	54	89	36	91	12
283	193	199	286	178	186	42

a جدول ثنائي يتكون من خمسة صفوف وسبعة أعمدة

هنا سوف يتم المرور على عناصر

المصفوفة عمودا عمودا

هنا سوف يتم المرور على عناصر

المصفوفة صفا صفا

```
#include<stdlib.h>
#include<iostream.h>
 void main( )
{
 int a[5][7],I,j,sum=0;
 for(I=0;I<7;I++)
 {
  for(j=0;j<4;j++)
  {
   a[j][I]=rand()%1000;
   sum=sum+ a[j][I];
   }
   a[j][I]=sum;
   sum=0;
  }
 for(I=0;I<5;I++)
 {
  for(j=0;j<7;j++)
    cout a[I][j];
   cout<<endl;
  }
}
```

مثال -2 على إستخدام الجدول/المصفوفة الثنائية:

اكتب برنامجاً يعرف جدولاً ثنائياً يتكون من خمسة صفوف وسبعة أعمدة من نوع int، ، ثم يولد فيه قيم عشوائية ما عدى العمود الاخير، فيضع في كل موقع منه مجموع عناصر الصف الذى فيه.

الحل

50	61	69	13	6	8	207
80	18	31	87	82	75	373
56	89	45	97	54	12	353
97	25	54	89	36	91	392
47	78	10	85	21	68	309

هنا سوف يتم المرور على

عناصر المصفوفة صفا صفا

مثال -3 على استخدام الجدول (المصفوفة) الثنائي:

اكتب برنامجاً يعرف جدولاً ثنائياً يتكون من ستة صفوف وسبعة أعمدة، من نوع int، ثم يولد فيه قيم عشوائية ما عدى الصفين الاخيرين، فيضع في كل موقع من الصف قبل الاخير مجموع عناصر العمود الذى فوقه، ثم يضع في كل موقع من الصف الاخير

ضعف قيمة عناصر العمود الذى فوقه، ثم يطبع محتوى المصفوفة بحيث تكون عناصر كل صف في سطر.

الحل

```
#include<stdlib.h>
#include<iostream.h>
void main( )
{
 int a[5][7],I,j,sum=0;
 for(I=0;I<5;I++)
 {
  for(j=0;j<6;j++)
   {
    a[I][j]=rand()%1000;
    sum=sum+ a[I][j];
    cout<< a[I][j]<< “  “;
    }
   a[I][j]=sum;
   cout<< a[I][j];
   sum=0;
   cout<< endl;
 }
}
```

```
#include<stdlib.h>
#include<iostream.h>
void main( )
{
  int a[6][7],I,j,sum=0;
  for(I=0;I<7;I++)
  {
    for(j=0;j<4;j++)
    {
      a[j][I]=rand()%1000;
      sum=sum+ a[j][I];
     }
    a[j][I]=sum; a[j+1][I]=sum*2;
    sum=0;
  }
  for(I=0;I<6;I++)
  {
   for(j=0;j<7;j++)
       cout<<a[I][j]<<" ";
      cout<<endl;
   }
 return ;
}
```

50	61	69	13	6	8	10
80	18	31	87	82	75	15
56	89	45	97	54	12	5
97	25	54	89	36	91	12
283	193	199	286	178	186	42
566	386	398	572	356	372	84

4.5. 1 أمثلة على المصفوفة الثنائية

س1 اكتب بلغة C++ برنامجا يضع في مصفوفة ثنائية تحتوي على القيم التالية ثم يطبع محتويات المصفوفة بحيث تكون عناصر كل صف في سطر .

3	6	9	12	15	18	21	24	27
27	30	33	36	39	42	45	48	51
51	54	57	60	63	66	69	72	75
75	78	81	84	87	90	93	96	99

```
#include <iostream.h>
main( )
 {
  int a[4][9],i,j,k=3;
for(i=0;i<4;i++)
{
 for(j=0;j<9;j++)
 {a[i][j]=k;
  cout<<k<<" ";
  k=k+3;
}
k=k-3;     cout<<endl;
}
retrun 0;
}
```

س2 اكتب بلغة C++ برنامجا يضع في مصفوفة ثنائية تحتوي على القيم التالية ثم يطبع محتويات المصفوفة بحيث تكون عناصر كل صف في سطر .

3	6	9	12	15	90	21	24	27
27	30	33	36	39	90	45	48	51
51	54	57	60	63	90	67	72	75
75	78	81	84	87	90	93	96	99
99	102	105	108	111	90	117	120	123
123	126	129	132	135	90	141	144	147

```
  #include <iostream.h>
  main( )
   {
  int a[6][9],i,j,k=3;
  for(i=0;i<6;i++)
  {
   for(j=0;j<9;j++)
   {  a[i][j]=k;  k=k+3;   }
  k=k-3; }
  for(i=0;i<6;i++)
   a[i][5]=90;
  for(i=0;i<6;i++)
    {
     for(j=0;j<9;j++)
         cout<<a[i][j]<<" ";
     cout<<endl; }
  retrun 0;
  }
```

س3 اكتب بلغة C++ برنامجا يضع في مصفوفة ثنائية تحتوي على القيم التالية ثم يطبع محتويات المصفوفة بحيث تكون عناصر كل صف في سطر .

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**
4	**8**	**12**	**16**	**20**	**24**	**28**	**32**	**36**
8	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**
16	**32**	**48**	**64**	**80**	**96**	**112**	**128**	**144**
32	**64**	**96**	**128**	**160**	**192**	**224**	**256**	**288**

```
#include <iostream.h>
main( )
 {
  int a[6][9],i,j;
  for(i=0;i<9;i++)
 {
 a[0][i]=i+1;
 cout<<a[0][i]<<" ";
  }
cout<<endl;
for(i=1;i<6;i++)
{
 for(j=0;j<9;j++)
  {
  a[i][j]= a[i-1][j]*2;
   cout<<a[i][j]<<"  ";
  }
    cout<<endl;
 }
retrun 0;
 }
```

س4 اكتب بلغة C++ برنامجا يضع في مصفوفة ثنائية تحتوي على القيم التالية ثم يطبع محتويات المصفوفة بحيث تكون عناصر كل صف في سطر .

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
4	8	12	16	20	24	28	32	36
8	16	24	32	40	48	56	64	72
16	32	48	64	80	96	112	128	144
32	64	96	128	160	60	224	256	288

```
#include <iostream.h>
main( )
 {
  int a[6][9],i,j;
  for(i=0;i<9;i++)
 {
 a[0][i]=i+1;
 cout<<a[0][i]<<" ";
  }
cout<<endl;
for(i=1;i<6;i++)
  for(j=0;j<9;j++)
    a[i][j]= a[i-1][j]*2;
 a[5][5]=60;
for(i=1;i<6;i++)
{
 for(j=0;j<9;j++)
   cout<<a[i][j]<<"  ";
    cout<<endl;
 }
retrun 0;
  }
```

س5 اكتب بلغة C++ برنامجا يضع في مصفوفة ثنائية تحتوي على القيم التالية ثم يطبع محتويات المصفوفة بحيث تكون عناصر كل صف في سطر .

1	**6**	**11**	**16**	**21**	**26**	**31**	**36**	**41**	**46**
2	**7**	**12**	**17**	**22**	**27**	**32**	**37**	**42**	**47**
3	**8**	**13**	**18**	**23**	**28**	**33**	**38**	**43**	**48**
4	**9**	**14**	**19**	**24**	**29**	**34**	**39**	**44**	**49**
5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
6	**11**	**16**	**21**	**26**	**31**	**36**	**41**	**46**	**51**

```
#include <iostream.h>
main( )
 {
int a[6][10],i,j,k=1;
for(i=0;i<6;i++)
{
 for(j=0;j<10;j++)
{
a[i][j]= k;
 cout<<k<<" ";
 k=k+5;
 }
   k=i+1;
 cout<<endl;
 }
retrun 0;
 }
```

س6 اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية
كما هو موضح، ثم يطبع محتوياتها عناصر كل صف في سطر

1	**6**	**11**	**16**	**21**	**26**	**31**	**36**	**41**	**46**	**51**	**1**
2	**7**	**12**	**17**	**22**	**27**	**32**	**37**	**42**	**47**	**52**	**2**
3	**8**	**13**	**18**	**23**	**28**	**33**	**38**	**43**	**48**	**53**	**3**
4	**9**	**14**	**19**	**24**	**29**	**34**	**39**	**44**	**49**	**54**	**4**
5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**5**
6	**11**	**16**	**21**	**26**	**31**	**36**	**41**	**46**	**51**	**56**	**6**

```
#include <iostream.h>
main( )
 {
int a[6][12],i,j,k=1;
for(i=0;i<6;i++)
{
 for(j=0;j<11;j++)
 {
  a[i][j]= k;
  cout<<k<<" ";
  k=k+5;
 }
k=i+1;
a[i][j]=k;  k++;
cout<<endl;
 }
retrun 0;
 }
```

6.1 تمثيل الجداول الثنائية في الذاكرة.

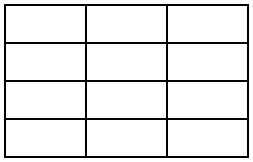
إحدى طرق تمثيل الجدول (المصفوفة) الثنائي هي Row-major representation

في هذه الطريقة، أول صف من الجدول (المصفوفة) يحتوي على قيمة الحدين الأعلى والأدنى للصفوف، و الصف الثاني يحتوي على قيمة الحدين الأعلى والأدنى للأعمدة.

إن العنوان المعطى من قبل النظام للجدول يكون عنوان أو خانة في أول صف وأول عمود، أما عنوان الخانة الثانية فيتم حسابه بزيادة سعة نوع البيان الذي يحتويه الجدول (المصفوفة) وسوف نوضحه في المثال التالي:

إذا تم تعريف الجدول (المصفوفة) شكل 1.21 على هذا النحو ;[3][4]int a وكان عنوان أول خانة هو 2562 فما هو عنوان آخر خانة في الجدول ؟

شكل 1.21 a مصفوفة ثنائية تتكون من أربعة صفوف وثلاثة أعمدة



الحل

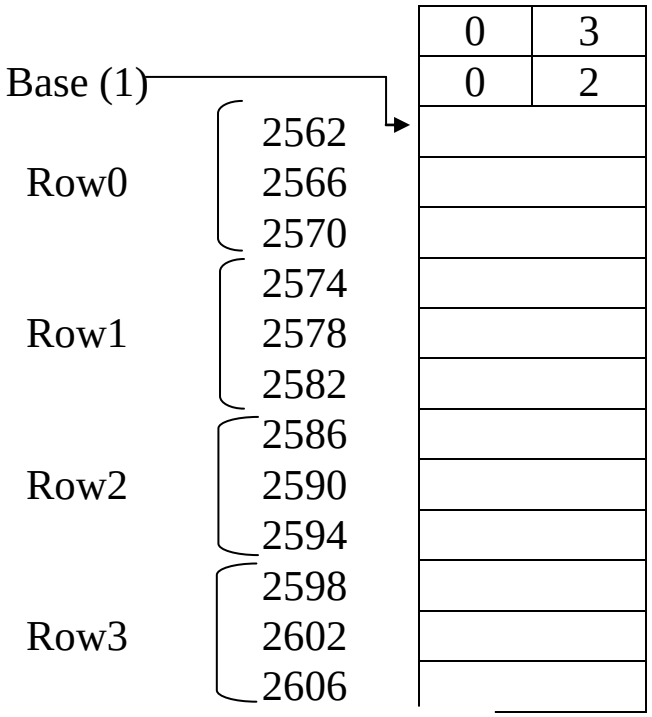
الجدول (المصفوفة) a تتكون من أربعة صفوف وثلاثة أعمدة، فإذا كان عنوان الخانة [0][0]a هو 2562.

إذا يكون عدد خانات الجدول (المصفوفة) 12=4*3 خانة.

بما أن عنوان الخانة الأولى هو 2562.

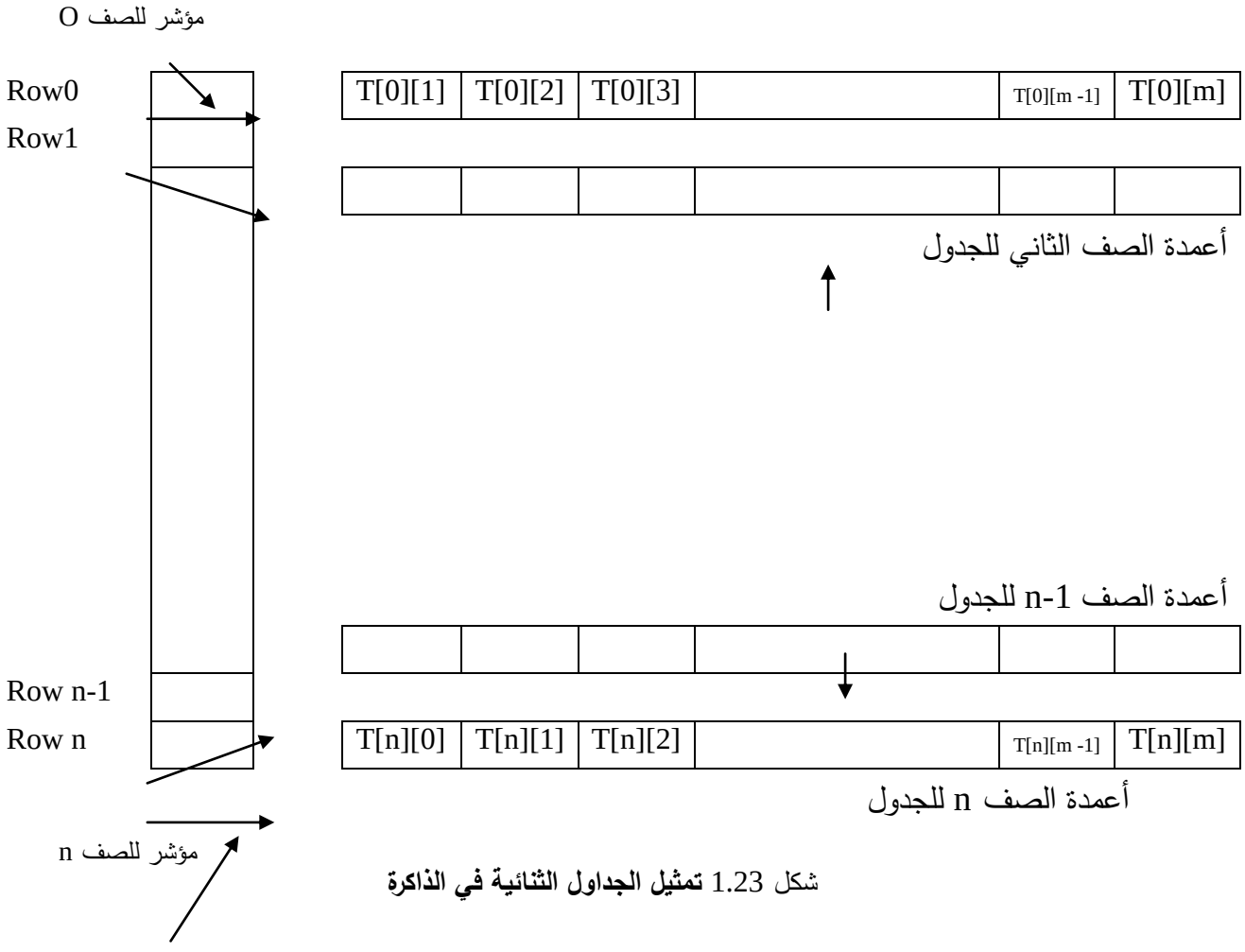
٪ فيكون عنوان آخر خانة = 11*4+2562 = 2606

في هذه الطريقة يمثل الجدول (المصفوفة) على شكل تتابع من bytes في الذاكرة ويخصص 2bytes في البداية تحتوى على عدد كل من الصفوف والأعمدة على التوالي، و الشكل 1.22 يوضح تمثيل الجداول الثنائية في الذاكرة.



شكل 1.22 **تمثيل الجداول الثنائية في الذاكرة**

ويمكن تمثيل الجداول الثنائية في الذاكرة على هذا النحو شكل 1.23 يوضح ذالك.



شكل 1.23 **تمثيل الجداول الثنائية في الذاكرة**

7.1 عيوب إستخدام الجداول:

سبق وأن أشرنا إلى مميزات إستخدام الجداول أما عيوبها فتمثل بالأتي:

1. يجب تحديد حجمها مسبقا، وهنا توجد بعض الصعوبة في بعض التطبيقات، حيث أن حجم البيانات قد لا يكون واضحا أثناء كتابة البرنامج أو النظام، وعندها يتم اللجؤ إلى حجز كمية إضافية من الذاكرة قد لا تستغل.
2. في الجدول (المصفوفة) الواحدة يجب أن تكون البيانات من نفس النوع، وهذا في الواقع العملي غير صحيح، لأننا نحتاج إلى تعدد البيانات من حيث النوع.
3. عند الحذف يبقى الموقع الذي حذفت منه المعلومة محجوزا حتى ولو لم يتم إستخدامه.
4. عند الإزاحة إلى الأمام أو إلى الخلف، يتم إزاحة كل العناصر، وهذا يستهلك عمليات ووقت أكثر.

أسئلة محلولة على الوحدة الأولى

1. اكتب برنامجاً يولد 150 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية البعد، ثم يطبع هذه الأعداد كل 5 أعداد في سطر، ثم يطبع في صف آخر مجموع تلك الأعداد .

الحل

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
int main ( )
 {
  int a[150],i;
long sum=0;
  for(i=0;i<150;i++)
  {
    a[i]=rand()%1000;
    sum+=a[i];
          cout<<a[i]<<”  “;
          if((i+1)%5==0)
         cout<<endl;
         }
         cout<<endl<<" the summation  is "<<sum;
                  return 0;
  }
```

2. اكتب برنامجاً يولد 150 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية البعد، ثم يطبع هذه الأعداد كل 8 أعداد في سطر، ثم يطبع في صف آخر أكبر وأصغر ومتوسط تلك الأعداد .

الحل

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
int main ( )
 {
  int a[150],i,max=0, min=1000;
```

```
 long sum=0;
  for(i=0;i<150;i++)
  {
    a[i]=rand()%1000;
    sum+=a[i];
          cout<<a[i]<<”  “;
   if(a[i]> max)
     max= a[i];
  else
    if(a[i]< min)
     min= a[i];

          if((i+1)%8==0)
         cout<<endl;
         }
         cout<<endl<<" the maximum  is "<<max<< “ the minimum is “<<min;
        cout<<" the averag  is “<<sum/i;
 return 0;
  }
//0000000000000000000000000000000000000
```

3. اكتب برنامجاً يولد 300 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية البعد، ثم يطبع هذه الأعداد كل 8 أعداد في سطر، ثم يطبع في صف آخر ثاني أكبر تلك الأعداد .

الحل

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
const maxs=300;
 int main ()
 {
  int a[maxs],i,m2,m1;
  m1=0;
  m2=0;
  for(i=0;i<maxs;i++)
  {
    a[i]=rand()%1000;
    cout<<setw(5)<<a[i];
   if(a[i]>m1)
   {
    m2=m1;
    m1=a[i];
    }
```

```
 else
 if(a[i]>m2)
 m2=a[i];
if((i+1)%8==0)
cout<<endl;
}
cout<<endl<<" the max no. is "<<m1<<endl;
cout<<" the second max is "<<m2;
 return 0;
 }
```

س4 اكتب بلغة C++ برنامجا يضع في مصفوفة ثنائية تحتوي على القيم التالية ثم يطبع محتويات المصفوفة بحيث تكون عناصر كل صف في سطر .

6	**11**	**16**	**21**	**26**	**31**	**36**	**41**	**46**
7	**12**	**17**	**22**	**27**	**32**	**37**	**42**	**47**
8	**13**	**18**	**23**	**28**	**33**	**38**	**43**	**48**
9	**14**	**19**	**24**	**29**	**34**	**39**	**44**	**49**
10	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
11	**16**	**21**	**26**	**31**	**36**	**41**	**46**	**51**

```
#include <iostream.h>
main( )
 {
int a[6][9],i,j,k=6;
for(i=0;i<6;i++)
{
 for(j=0;j<9;j++)
{
a[i][j]= k;
 cout<<k<<" ";
 k=k+5;
 }
   k=i+7;
 cout<<endl;
 }
retrun 0;
 }
```

س5 اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية كما هو موضح، ثم يطبع محتوياتها عناصر كل صف في سطر

6	**11**	**16**	**21**	**26**	**31**	**36**	**41**	**46**	**51**	**1**
7	**12**	**17**	**22**	**27**	**32**	**37**	**42**	**47**	**52**	**2**
8	**13**	**18**	**23**	**28**	**33**	**38**	**43**	**48**	**53**	**3**
9	**14**	**19**	**24**	**29**	**34**	**39**	**44**	**49**	**54**	**4**
10	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**	**55**	**5**
11	**16**	**21**	**26**	**31**	**36**	**41**	**46**	**51**	**56**	**6**

```
#include <iostream.h>
main( )
 {
int a[6][11],i,j,k=6;
for(i=0;i<6;i++)
{
 for(j=0;j<10;j++)
 {
  a[i][j]= k;
  cout<<k<<" ";
  k=k+5;
 }
k=i+7;
a[i][j]=i+1;
cout<< a[i][j]<<endl;
 }
retrun 0;
 }
```

الخلاصة

1. يمكن تشكيل ذاكرة الحاسوب على هيئة جداول / مصفوفات أحادية أو ثنائية تسمى Arrays
2. يمكن إستخدام الجداول / المصفوفات الأحادية والثنائية كهياكل للبيانات
3. من اهم عيوب المصفوفات، الأحادية البعد، أو الثنائية البعد، أنها تحتوي على مجموعات متشابهة من البيانات.
4. حجم هذه المصفوفات ثابتة ومحددة من قبل المبرمجين أثناء كتابة البرنامج ولا يمكن تجاوزة عند التنفيذ بأي حال من الأحوال.
5. عند التعامل مع المصفوفات الأحادية لا بد وأن يشار إلى كل عنصر من عناصر المصفوفة عن طريق ذكر اسم المصفوفة ثم ذكر رقم الموقع، و عند التعامل مع المصفوفات الثنائية لا بد وأن يشار إلى كل عنصر من عناصر المصفوفة عن طريق ذكر اسم المصفوفة ثم ذكر الصف و العمود، و يمكن أن تحتوي على (two loops) حلقتين كي تغطى كل خانات الجدول (المصفوفة)، ويكون مدى كل حلقة هو أما عدد الصفوف أو عدد الأعمدة للمصفوفة.
6. للتعامل مع المصفوفات الأحادية ، يكون الإحتياج إلى one loop، وذالك للمرور على كل خلايا المصفوفة الأحادية .
7. للتعامل مع المصفوفات الثنائية، يكون الإحتياج إلى two loops، وذالك للمرور على كل خلايا المصفوفة الثنائية.
8. معظم لغات البرمجة تدعم تكوين المصفوفات الأحادية و الثنائية، ويكون لتعامل مع خلايا المصفوفات مباشرة، Direct access .

اسئلة على الوحدة الأولى

1. اكتب برنامجاً يولد 50 عددا من نوع int ثم يضع هذه الأعداد في مصفوفة ثنائيه (مكونة من5 صفوف و11 أعمده) ثم يضع في العمود الأخير مجموع كل صف ثم يطبع هذه الأعداد كل صف في سطر، ثم يطبع قيمة وموقع أصغر هذه الأعداد في المجموعة .
2. اكتب برنامجاً يولد 50 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية ، ثم يطبع هذه الأعداد كل10 أعداد في صف، ثم يقوم بترتيب هذه العناصر تصاعدياً ثم يطبع هذه الأعداد كل5 أعداد في سطر.
3. اكتب برنامجاً يعرف مصفوفة مكونة من 7 صفوف و5 أعمدة، ثم يولد قيم عشوائية في كل صف ما عدى الصف الأخير والعمود الأخير من المصفوفة يضع فيهم أصفارا، ثم يقوم البرنامج بوضع المجموع لكل صف في العمود الأخير والمجموع لكل عمود في الصف الأخير ثم يطبع هذه الأعداد كل صف في سطر.
4. اكتب برنامجاً يولد 100 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية البعد، ثم يطبع هذه الأعداد كل 8 أعداد في سطر، ثم يطبع سطر جديد، ثم يطبع فقط محتويات المواقع الفردية من المصفوفة كل 8 أعداد في سطر، ثم يطبع سطر جديد، ثم يطبع فقط محتويات المواقع الزوجية من المصفوفة كل 8 أعداد في سطر.

2. الباب الثاني المؤشرات Pointers

الهدف من هذا الباب هو الأتي:

a) **التعرف على المؤشرات، و التعامل مع Pointers**

b) **القدرة على التعامل مع الذاكرة عن طريق المؤشرات Pointers**

c) **إستخدام مواقع الذاكرة للبيانات عن طريق المؤشرات Pointers**

2. المؤشرات Pointers

من العوامل التي جعلت لغة C ++ قوية ومشهورة بين لغات الحاسوب، و في كل أرجاء العالم، هو إستخدامها للمؤشرات Pointers ، حيث أن بإستخدام المؤشرات يتم تذليل الكثير من العقبات، و بشكل خاص بناء هياكل البيانات.

ولكن لوجود الصعوبة النسبية لفهم المؤشرات، يحاول كثير من المبرمجين تجاوز هذه الميزة الكبيرة.

ونحن في هذا ال Course سوف ندرس المؤشرات بعمق، بحيث نكون لدى الطالب/القارى مفهوما متكاملا لإستخدام المؤشرات، وسوف يكتشف الطالب/القارى بنفسه مميزات إستخدام المؤشرات بإذن الله تعالى.

2.0. تعريف المؤشر Pointer

يمكن أن تعريف المؤشر Pointer بأنه موقع في الذاكرة يمكن أن يحتوي على عنوان موقع أخرى

A pointer is a variable which contains address of another variable.

من التعريف نجد ان المؤشر Pointer عبارة عن موقع يمكن أن يحتوي فقط على عنوان، ولا يمكن أن يحتوي على قيمة عادية، ولكن الموقع الذي يشيرإليه المؤشر فإنه يمكن أن تكون فيه قيمة عادية بحسب التعريف.

1.2 الرموز الخاصة بالمؤشرات

عند التعامل مع المؤشرات، يتم إستخدام كثير من **الرموز مثل &، ***، <- ، ولكى نوضح إستخدام المؤشرات نتبع الجمل التالية:

أولا إعتبر التعريف التالي:

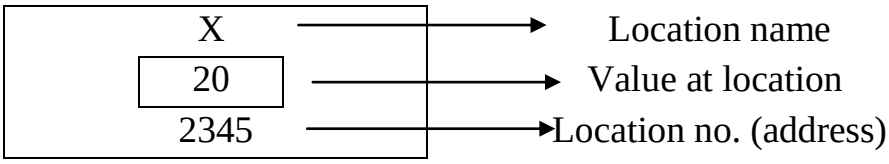
```
int x = 20;
```

هذا التعريف أو التصريح يخبر C++Compiler للقيام بالتالي:

1- حجز موقع في الذاكرة تحفظ فيه قيمة عدد صحيح.

2- اسم هذا الموقع x

3- عند التعريف توضع القيمة 20 في الموقع الذي إسمه x ويمكن أن يمثل موقع x في الذاكرة على شكل 2.1:



شكل 2.1 **تمثيل** موقع **في الذاكرة**

إن قيمة عنوان الموقع يعتمد على نوع وحالة الجهاز وسعت ذاكرته و.. الخ، وعليه، فإنه يمكن أن تكون هناك قيمة أخرى لعنوان الموقع، حتى في نفس الجهاز، إذا تم التعريف بأوقات مختلفة، كما أن العنوان يمثل دائما برقم صحيح لمإذا؟ ويمكن طباعة العنوان على الشاشة بإظهار الرمز الخاص به على النحو:

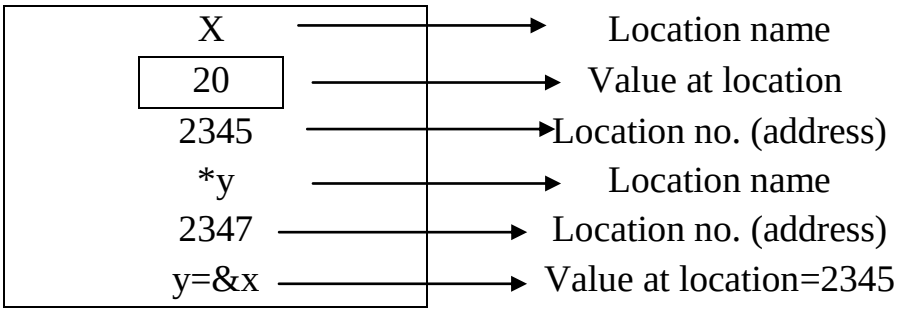
```
cout<<"address of x= "<<&x ;
```

```
int x = 20,*y;
y=&x;
```

ثانيا إعتبر الجمل التالية:

هذاه الجمل تخبر C++Compiler للقيام بالتالي:

1. حجز موقعين في الذاكرة ، الأول من نوع int تحفظ فيه قيمة عدد صحيح، إسم هذا الموقع x.
2. الثاني من نوع مؤشر إلى موقع من نوع int ، إسم هذا المؤشر y.
3. عند التعريف توضع القيمة 20 في الموقع الذي إسمه x .
4. الأمر الثاني y=&x اجعل في المؤشر y قيمة عنوان الموقع x
5. يمكن أن تمثل مواقع x , y في الذاكرة على شكل 2.2:



شكل 2.2 **تمثيل** مواقع **في الذاكرة**

و البرنامج التالي يوضح ذالك.

هذا البرنامج يوضح طباعة عنوان موقع في الذاكرة

```
int main (    )
{
int  x= 20;
cout<<“address of x= ”<<&x;
cout “Value of x=”<<x;
return 0;
}
```

هنا يتم طباعة العنوان

نلاحظ أن المخرجات سوف تختلف من جهاز لآخر بالنسبة للعنوان فقط، أما قمة x فإنها متماثلة في كل الأجهزة، وستكون المخرجات في بعض الأجهزة هي على النحو

```
Address of x = 2345
  Value of x = 20
```

فإذا نظرنا إلى الأمر الأول cout<<&x سنجد أنه يحتوي على &x أي أن المطلوب طباعته ليس x وأنما &x الذي هو عبارة عن عنوان الصندوق أو عنوان الموقع x.

أما الأمر الثاني cout<<x فإن المطلوب فيه طباعة قيمة x أو ما يحتويه الموقع الذي اسمه x.

إذا الرمز & يستخدم (إذا أضيف إلى المتغير) للدلالة على عنوان المتغير وليس على المتغير نفسه.

2 .2 . 1 **Pointer expressions**

إستخدام المؤشرات

لنفرض أننا كتبنا التعريف التالي int i=3,*j; فيكون معنى هذ التعريف كما يلى:

1) إحجز موقعاً من نوع int وسمى هذ الموقع i
2) ضع القيمة 3 في الموقع i
3) احجز موقعاً من نوع مؤشرا يشير إلى موقع من نوع int وسمى هذ الموقع j

عند إذ فيمكن أن نسند قيمة عنوان i إلى J ;j=&i لأن J من نوع pointer يشير إلى int ، ;j=&i وهذا الأمر معناه اجعل المؤشر J يشير إلى عنوان i ويمكن تخيله في الذاكرة على شكل 2.3:

شكل 2.3 **تمثيل** مواقع **i,j الذاكرة**

وعلى هذا يكون في المؤشر j قيمة العنوان للموقع i

أما هذا التصريح ; int *J فيقول للمترجم أن J سوف يستخدم لحفظ عنوان ما، لقيمة ما، من نوع int أي أن j يشير إلى int. إذا الرمز * مرفق مع اسم متغير يعني قيمة الموقع، أو القيمة الموجودة في هذا العنوان كما سنرى في البرنامج التالي:

2 .2 . 2 Example for using pointer

مثال عاى إستخدام المؤشر

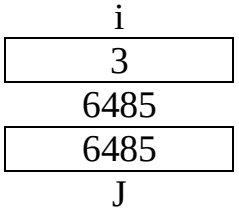
```
int main (    )
{
int  i= 3, *J;
J= &i;
cout<<endl<<" address of i= "<<& i;
cout<<endl <<"address of i= "<<J;
cout<<endl "address of J= "<<&J;
cout<<endl<<" Value of J= " <<J;
cout<<endl<<" Value of i= " <<i;
cout<<endl<<" Value of i= "<<*(&i);
cout<<endl <<"Value of i= "<<*J;
return 0;
}
```

مخرجات هذا البرنامج في بعض الأجهزة على النحو:

Address of i = 6485
Address of i = 6485
Address of J = 3276
Value of J = 6485
Value of i = 3
Value of i = 3
Value of i = 3

هنا سوف نحاول نتتبع البرنامج السابق خطوة خطوة.

في السطر الثالث يعرف المتغير i من نوع int ويضع القيمة 3 فيه، في السطر الرابع يعرف المتغير J من نوع مؤشر إلى int



في السطر الخامس المؤشر J يحتوي على عنوان المتغير i، إذا كانت قيمة عنوان i هو 6485 ، فإن J يحتوي على6485 ويجب أن نعرف أن J له أيضاً عنوان وليكن 3276.

J

6485

3276

في السطر السادس أمر بطباعة &i أي طباعة عنوان i فتظهر الرسالة مضافاً إليها 6485.

في السطر السابع أمر بطباعة J فتظهر قيمة J وهي 6485.

في السطر الثامن أمر بطباعة &J فتظهر قيمة J وهي 3276.

في السطر التاسع أمر بطباعة J فتظهر قيمة J وهي 6485.

في السطر العاشر أمر بطباعة i فتظهر قيمة i وهي 3.

في السطر الحادي عشر أمر بطباعة (&i)* وهي كما ذكرنا قيمة i فتكون 3 في هذا الأمر قلنا أن الأقواس لها أسبقية في العمليات فيتم إيجاد ما بداخل القوس أولاً وهو حساب قيمة &i الذي هو عبارة عن عنوان المواقع i ثم يعد ذلك يأتي حساب العنوان مع * الذي يعتبر قيمة كما ذكرنا، وعليه فيكون ناتج العمليات هو قيمة ما يحتويه الموقع i وهو 3.

في السطر الثاني عشر أمر بطباعة J* وهو عبارة عن قيمة الموقع الذي يشير إليهJ , وبالتالي فإن J مؤشر إلى عنوان، هذا العنوان هو موقع فيه قيمة، هذه القيمة هي 3.

تلخيص العمليات السابقة عند طباعتها على الشكل 2.4

ناتج الطباعة	الرمز
قيمة i	i
محتوي الموقع الذي عنوانه i	*i
عنوان i	&i
قيمة i	*(&i)

شكل 2.4 تلخيص العمليات على المؤشر

من هنا علينا أن نعرف أن المؤشر إنما يؤشر إلى عنوان وليس إلى قيمة وانما تكون القيمة موجودة في العنوان المشار إليه بواسطة المؤشر.

بعد هذا يمكن أن نتذكر تعريف ال Pointers

A pointer is a variable which contains address of another variable.

و ايضا نتذكر أن هناك ثلاثة أشياء، على الشكل 2.4 وهو تلخيص العمليات على المؤشر

1- قيمة الموقع الذي اسمه i

2- قيمة عنوان الموقع &i

3- قيمة الموقع الذي عنوانه i*

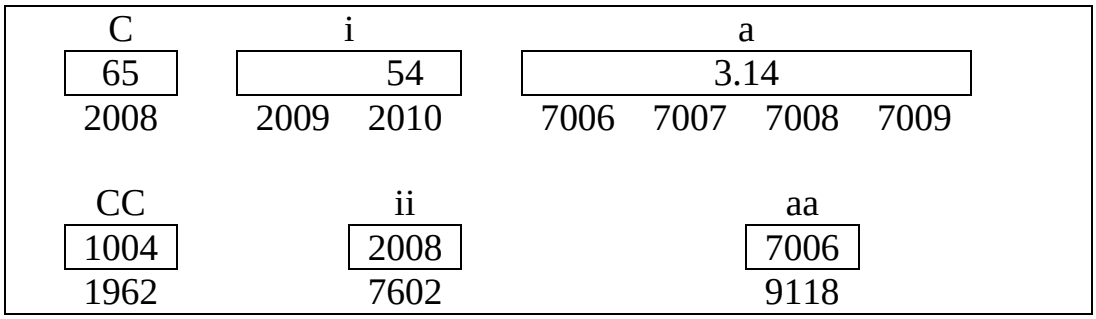
مزيد من الأمثلة على إستخدام المؤشرات

س1: ما هي مخرجات هذا البرنامج إذا كان عنوان كل من: a = 7006 i = 2008 C= 1004

```
void main ( )
{
char C, *CC;
int  i,*ii;
float a, *aa;
C=”A ”;  i=54; a = 3.14;
CC= &C; ii=&i; aa=&a;
cout<<endl<<” CC =”<<CC;
cout<< endl<<” ii = ”<<ii;
cout<< endl” aa = ”<<aa;
cout<<endl <<*CC;   cout<< endl <<*ii;
cout<<endl <<*aa;
}
```

يمكن تمثيل المساحة للمؤشرات و البيانات بمجموعة من المواقع داخل الذاكرة باشكل 2.5

لاحظ المساحة التي تحجز لكل متغير وخاصة المساحة للمؤشرات وهل تعتمد على النوع التي تشير إليه؟



شكل 2.5 تمثيل المساحة للمؤشرات و البيانات داخل الذاكرة

س2: ما هي مخرجات هذا البرنامج

```
int main (    )
{
char C=65;
int  i,*ii,a=21;
ii=&a;
for (i=0;i<10;i++)
 {
   C=C+i;
   a = a+i;
   cout<<* ii <<”  “<<C << endl;
}
return 0;
 }
```

مخرجات هذا البرنامج على النحو

21 A
22 B
23 C
24 D
25 E
26 F
27 G
28 H
29 J

Functions&Passing values or addresses to functions 2 .2 . 3

إن التواصل بين البرنامج الرئسي والدوال أو بين الدوال و الدوال يتم عن طريق المحاورات، حيث يمكن أن تمرر المحاورات إلى الدوال بإحدى طريقتين:

1- بإرسال قيم المحاورات.

2- بإرسال عناوين المحاورات.

المحاورات هي تلك المتغيرات التي تكون موجدة بين قوسي الدالة عند إستدعائها.

عند التعامل مع قيم المحاورات، فإن القيم الأصلية للمتغيرات المحاورة تضل كما هي عليه دون أن يحصل عليها أي تغير.

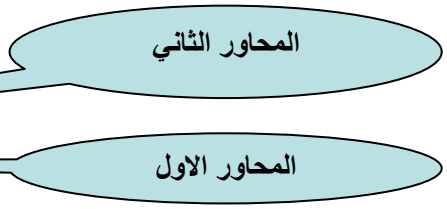
أما عند التعامل مع عناوين المحاورات، فإن القيم الأصلية للمتغيرات المحاورة تتغير.

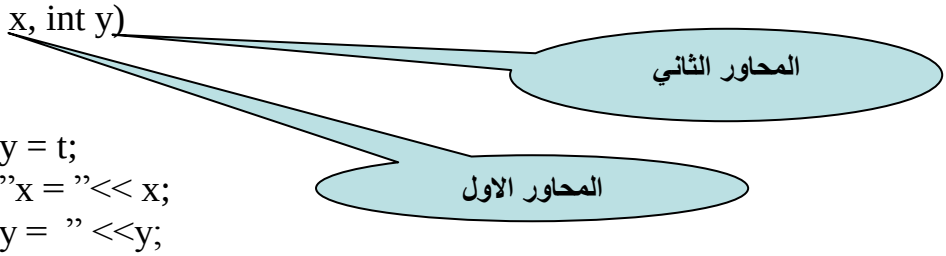
وبتتبع البرنامج التالي يمكن معرفة الفرق بين الطريقتين:

أولا الإستدعاء بإرسال قيم المحاورات

First calling function by values

```
void swapv(int x, int y);
void main (    )
{
int  a = 10;
int b= 20;
swapv (a, b);
cout<<endl<<" a = "<<a;
cout<<endl <<"b = "<<b;
}

void swapv(int x, int y)
{
int   t;
t = x;   x= y;    y = t;
cout<<endl <<"x = "<< x;
cout<<endl<< y =  " <<y;
}
```





عند تنفيذ البرنامج وبالرغم من استدعاء الدالة swapv لكن القيم الأصلية لكل من a,b تضل كما هي عليها، إذا إن المخرجات تكون على النحو التالي:

x = 20
y= 10
a = 10

b = 20

أي أن قيم المتغيرات التي طبعت داخل الدالة تم تغيرها، أما قيم المتغيرات التي طبعت خارج الدالة فإنهما لم تتغير والسبب في ذلك أننا استدعينا الدالة swapv ومررنا إليها قيم المحاورات، وهذا بدوره يجعل النظام يأخذ نسخة لكل متغير ويتعامل معها، أما لنسخه الأصلية فإنها تبقى كما هي دون أن يحصل عليها أي تغير.

2 . 2 . 4 Calling function by references (addresses)

ثانيا الإستدعاء بإرسال عناوين المحاورات

ولكن إذا أردنا تغير النسخة الأصلية فما علينا إلا أن نرسل عناوين المحاورات بدلاً من قيمها كما سنرى في المثال التالي.

```
void swapv (int * x, int  *y);
void main (    )
{
int  a = 10,  b= 20;
 swapv (&a,&b);
cout<<endl <<“a = ”<<a;
cout <<”b = ”<<b;
}
void swapv (int *x, int  *y)
{
int   t;
cout<<endl<<” x = “<< * x<<”y= “<< *y;
t = *x;
 *x= *y;
 *y = t;
 cout<<endl<<” x = “<< * x<<”y= “<< *y;
}
```

عند تنفيذ هذا البرنامج تكون المخرجات هي:

```
  x=10    y=  20

  x= 20    y=10
  a= 20    b= 10
```

أي أن النسخة الأصلية لكل من a,b قد حصل تغير عليها، والسبب أنه قد أرسل عناوين المحاورات عند الإستدعاء لدلالة swapv ، ولذلك فإن أي تغير يحصل على قيم المحاورات، فإن هذا التغير يعكس على النسخة الأصلية.

المؤشرات وأنواع البيانات

للتعامل بالمؤشرات مع أنواع البيانات دعنا نكتب البرنامج التالي ثم سوف يتم شرحه.

```
main (  )
{
int  i=3, * x;
float J = 1.5, *y;
char K= 'C', *Z;
cout<<endl <<"i ="<<i<<"  J= "<<"   K= "<< K;
x = &i;      y = &J;       Z= &K;
cout<<endl<<" x ="<< x<< y= "<<y<<  z= "<< z;
x ++;  y++; Z++;
cout <<endl <<"new values of x, y and z";
cout<<endl<<" x ="<< x<< y= "<<y<<  z= "<< z;
}
```

إفرض ان النظام حفظ المتغيرات

x ، J، i عند العناوين التالية:

&i= 1002
&J= 2004
&K= 5006

عند تنفيذ هذا البرنامج تكون مخرجاته على النحو التالي:

```
i = 3          J= 1.500000          K= C
x = 1002       y= 2004              Z= 5006
new values of x, y and z
x = 1004       y= 2008              Z= 5007
```

أنظر بعناية إلى هذه المخرجات ولاحظ مقدار الزيادة في كل من x ,y ,z.

القيمة x تم زيادتها بمقدار 2 حيث أن x مؤشرا إلى موقع من نوع int وأن البيانات من نوع int يتم حفظها بمكان في الذاكرة طوله 2bytes ، أما مقدار الزيادة فإنها تعني الإنتقال إلى عنوان القيمة التالية أي للعنوان التالي، هذا العنوان تكون بدايته بعد 2bytes من العنوان الحالي، ولهذا السبب تمت الزيادة بمقدار 2.

وكذلك تمت زيادة y بمقدار 4 لأن y مؤشرا إلى موقع من نوع float وهذا النوع من البيانات يتم حفظه في موقع طوله 4bytes، وعليه فإن العنوان الحالي إذا كانت قيمته إفتراضا 2004 فإن العنوان التالي له تكون قيمته 2008، وهي ما تحتويه y حالياً أي بعد الزيادة.

أما لمتغير Z فقد كانت الزيادة فيه بمقدار 1 أو واحدا، وذلك لأن z مؤشرا إلى بيان من نوع char الذي يحتاج لحفظه فقط one byte.

هذا بالنسبة للزيادة وبنفس الكيفية تكون عملية التنقيص- أي أن مقدار النقص يعتمد على النوع الذي يشير إليه المؤشر.

2 .2 . 5 Pointers and arrays

المؤشرات و arrays إسمان لمسمى واحد، فيمكن إستخدام المؤشرات مكان arrays والعكس صحيح.

ولكي نعرف مإذا يمكن أن تفعله المؤشرات مع arrays

```
#include<iostream.h>
void display (int *J, int n);
main (    )
{
int  a [ ]= {24, 34, 12, 44, 56, 17},I;
for(I=0;I<6;I++)
  cout<<a[I]<<”  “;
 cout<<endl;
display (a , 6);
return 0;
}
void display (int *J, int n)
{
  int I = 0;
 while(I<n)
  {
    cout <<*J<<”  “ ;
    I++; J++; /* increment pointer to point to next location */
   }
return ;
}
```

عند تنفيذ هذا البرنامج تكون مخرجاته على النحو التالي:

```
24 34 12 44 56 17
24 34 12 44 56 17
```

المخرجات الأولى تكون من البرنامج الرئيسي، و المخرجات في السطر الثاني تكون من الدالة display

مما سبق نستنتج الأتي:

1- تحفظ محتويات الجدول (المصفوفة) (array) في مواقع متصلة بالذاكرة.

2- عندما تزاد قيمة المؤشر بمقدار واحد فإن الزيادة هذه تعني أن المؤشر يشير بعد الزيادة إلى موقع جديد وبالتالي تكون الزيادة الفعلية هي عبارة عن طول سعة البيان المحفوظ في array.

مثال: int a[ ]= {10,20,30,40,50}

هذا يمثل بالذاكرة على النحو التالي إذا فرضنا إن العنوان يبدأ بالعدد 4001

a [0]	a [1]	a [2]	a [3]	a [4]
10	20	30	40	50
4001	4003	4005	4007	4009

شكل 2.6 تمثيل المساحة للمصفوفة و البيانات داخل الذاكرة

Passing an entire array to a function 2 .2 . 6

Function &Passing an entire array to a function تمرير كل array إلى دالة كمحاور

من الممكن أن نجعل محاول أي دالة array وذلك بإرسال عنوان array أي عنوان الخانة الأولى بالمصفوفة (array) ومن ثم تستطيع أن تزيد العنوان بمقدار واحد للحصول على عنوان الخانة التالية وهكذا.

ولكي نوضح هذه الفكرة فيتم كتابة البرنامج التالي ثم نتتبعه

```
#include<iostream.h>
void display (int [ ], int n);
main (    )
{
int  a [ ]= {24, 34, 12, 44, 56, 17};
display (a , 6);
return 0;
}
void display (int b[ ], int n)
{
  int I = 0;
 while(I<n)
  {
    cout <<endl<< "element= "<<b[I];
    I++; /* increment pointer to point to next location */
   }
return ;
}
```

a [0]	a [1]	a [2]	a [3]	a [4]	a [5]
24	34	12	44	56	17
4001	4003	4005	4007	4009	4011

شكل 2.7 تمثيل المساحة للمصفوفة a و البيانات داخل الذاكرة

عند إستدعاء الدالة display(a,6)، فإننا نستدعي هذه الدالة ويكون المحاور الأول هو عنوان a كاملاً، أي بداية عنوان a الذي هو عبارة عن عنوان الخانة [0] a وعند الزيادة داخل while فإن I تزداد بمقدار واحدا .

وعلى ذلك فتكون مخرجات هذا البرنامج هي كما في شكل 2.8:

element = 24
element = 34
element = 12
element = 44
element = 56
element = 17

شكل 2.8 تمثيل مخرجات البرنامج السابق

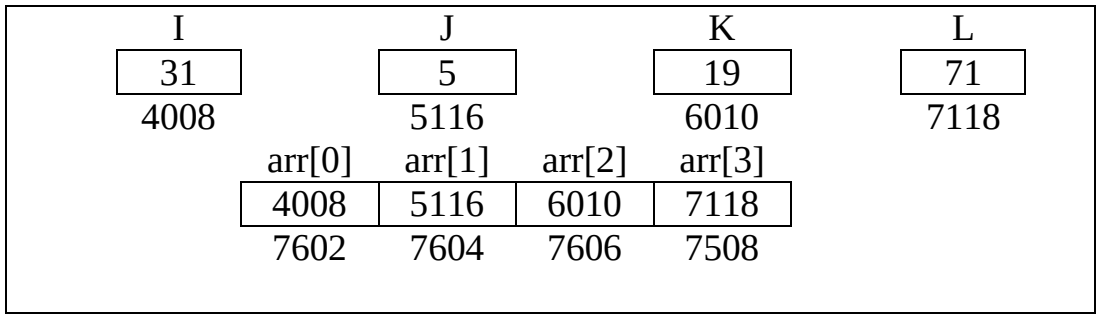
كذلك يمكن أن نستبدل a[0] & بالرمز a* حيث أن المؤشر a* يشير إلى عنوان أول خانة (array) كذلك a = *(a+0)* حيث أن (a+1)* هو عنوان الخانة الثانية a[1] وهذا ما يحدث بالفعل عندما يتم كتابة cout<< &a[1]; مثلاً، فإن النظام يوجد عنوان الخانة بهذه الطريقة: (a +i) * هذا يشير إلى عنوان القيمة a[i] ، ولمزيد من الفهم حاول أن تتبع البرنامج التالي وتعرف مخرجاته إذا كان النظام قد وضع الخانة الأولى a في العنوان 4001.

```
#include<iostream.h>
void display (int *J, int n);
main (    )
{
int  a [ ]= {24, 34, 12, 44, 56, 17, 20, 30, 40, 50};
display (a , 10);
return 0;
}
void display (int *J, int n)
{
  int I = 1;
 while(I<n)
  {
    cout <<endl<< "element= "<<*J ;
    I++; J++; /* increment pointer to point to next location */
   }
 return ;
}
```

7 . 2 .2 Array of pointers

بنفس الطريقة التي نكون فيها array من أي نوع، يمكن أن نكون array of pointers ، وذالك لأن pointers ماهي إلا متغيرات تحتوي على عناوين لمتغيرات أخرى. أي أن array of pointers عبارة عن مجموعة من العناوين. وممكن أن تكون هذه العناوين عبارة عن عناوين لمتغيرات أحادية single variable، أو عناوين array ، ويمكن توضيحها بتتبع البرنامج التالي إذاحاولنا أن نعرف ما هي مخرجاته.

```
#include<iostream.h>
main (  )
{
int  * arr [4 ];
int I=31, J= 5, K= 19, L= 71, m;
arr[o]= &I; arr[1]=&J; arr[2]= &K; arr[3] =&L;
for (m= 0; m <= 3; m++)
cout<< endl<<&arr[m];
return 0;
}
```



شكل 2.9 تمثيل المساحة للمصفوفة arr و البيانات داخل الذاكرة البرنامج السابق

وعلى ذلك تكون مخرجات البرنامج في بعض الأجهزة على النحو كما في شكل 2.10:

7602 7604 7606 7508

شكل 2.10 تمثيل مخرجات البرنامج السابق

والسبب أن عنوان أو خانة في array هو 4008 ، والمؤشر إليها (0 + arr) *

أما الخانة الثانية فيكون مؤشرها (arr+1)* 7604

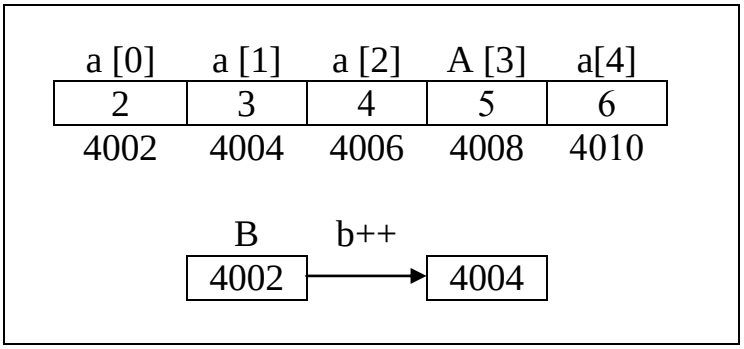
أما الخانة الثالثة فيكون مؤشرها (arr+2)* 7606

أما الخانة الرابعة فيكون مؤشرها (arr+3)* 7508

س: ما هي مخرجات هذا البرنامج؟

```
#include<iostream.h>
void change (int *b);
main ( )
```

```
{
int a[5] = {2,3,4,5,6},i;
change (a);
for (i=4; i>=0 ; i--)
cout<< &a[i]<<"    "<<a[i];
return 0;
}
void change (int *b)
{
int i;
for (i=0, i<=4; i++)
{
 *b= *b +1;
  b++;
 }
return ;
}
```



شكل 2.11 تمثيل مخرجات البرنامج السابق

البرنامج يستدعي الدالة change بالمحاور a الذي هو عبارة عن array وهو عبارة عن (a) * أي عنوان أول خانة في array ثم إن قيمة b* تزداد بمقدار واحد داخل for loop ويحصل زيادة بمقدار واحد b داخل for loop مع العلم بأن b مؤشرا، أي أن زيادته بمقدار واحدا، نعني إنتقال المؤشر إلى العنصر المجاور، وعلى ذلك تكون مخرجات البرنامج هي:

7 6 5 4 3

2.3 Character String السلاسل

عادة ما تكون أسماء الأشخاص وعناوينهم عبارة عن تتابع من الحروف، هذا التتابع من الحروف يسمى **String** أو سلسلة.

2.3.0 String عبارة عن سلسلة من الحروف أو سلسلة من الأشكال مترابطة مع بعضها. في كثر من لغات البرمجة، يتم التعامل مع الأسماء وكأنها عبارة عن مجموعة من الحروف "array of char" وعند قراءتها أو كتابتها، يتم تكوين loop تتكرر بعدد الحروف ثم يتم قرءاتها أو كتابتها حرفاً حرفاً.

ويمكن أن يمثل String على أنه عبارة عن سلسلة من الحروف (الأشكال)، تنتهي هذه السلسلة برمز يشير إلى نهاية السلسلة، هذ الرمز اسمه Null ورمزه " 0 back slash and 0\". هذا الرمز الذي يشير إلى نهاية السلسلة، يضاف تلقائياً من النظام، ولا يظهر عند طباعة السلسلة، بل يتخذه النظام للتعرف على نهاية السلسلة.

ومما ينبغي التنبيه إليه أن "0\ " رمزاً واحداً وليس رمزين، وله قيمة واحدة في ASCII تساوي 0 أى أن 0\=0 لأن قيمة 0 في ASCII= 48 بينما قيمة 0\ = 0.

2.3.1 تمثيل السلاسل string بلغة C++

من المكن أن تمثل ال string على هيئة مجموعة من الحروف على النحو

```
char name [ ] = {'S', 'A', 'L', 'E', 'H', '\0'};
```

في هذه الحالة لكي نتعامل مع name كنوع string لا بد من إضافة الرمز 0\ إلى آخر السلسلة.

ولكن هذه الطريقة أقل فعالية، إذ من الممكن تعريف نفس السلسلة على هذا النحو:

```
char name [ ] = "SALEH"
```

وفي الحالة الأخيرة لا داعي لوضع الرمز 0\ في نهاية السلسلة، لأن النظام يقوم بهذه العملية ذاتياً. ولكى يتم توضيح التعامل مع string نتبع هذا البرنامج ونلاحظ مخرجاته :

```
#include<iostream.h>
main ( )
{
 char name[ ] =  "SALEH NOUMAN";
 char *Ptr;
 Ptr= name; /*stores base address of string*/
 while (Ptr != '\0')
{
  cout<< *Ptr;
  Ptr++;
 }
return 0;
}
```

نلاحظ في هذ البرنامج أن المتغير الذى يحتوى على السلسلة "SALEH NOUMAN" هو name[ ] ، وأن حجمه غير مذكور صراحةً، وأنما يستنبطه النظام من عدد حروف السلسلة الموضوعة فيه.

أما مخرجات هذا البرنامج فهي كالتالي:SALEH NOUMAN

فإذا تتبعنا البرنامج السابق نجد أنه يتكون من 11 سطراً، في السطر الثالث تم تعريف name على شكل array فيه قيمة مبدئية من الحروف "SALEH NOUMAN"، وفي السطر الرابع يوجد تعريف Ptr* مؤشر إلى بيانات من نوع char ، في السطر الخامس Ptr=name أي أن Ptr يوضع فيه عنوان name لأن Ptr مؤشراً، و name عبارة عن array ، فهذا الأمر مقبول بلغة ++ C.

في السطر السادس تدخل while loop شرط دخولها هوعدم تساوي ما يشير إليه Ptr إلى القيمة "0\ "،

وفى داخل while loop في السطر الثامن أمر بالطباعة لقيمة موقع ما يشير إليه هذا العنوان Ptr* ، وفي السطر التاسع داخل while loop تحصل زيادة للمؤشر Ptr بمقدار واحد، أي أنه في كل مرة يشير المؤشر Ptr إلى البيانات التالية، وبما أن المؤشر Ptr يشير إلى بيانات من نوع char التي تحتاج إلى 1byte فقط لحفظ كل بيان، فإن مقدار الزيادة تكون فقط بمقدار واحد، وعلى هذا فيمكن تمثيل المعلومات في الذاكرة كما في شكل 3.28

S	A	L	E	H		N	O	U	M	A	N	\0
4001	4002	4003										4013
					Ptr							

شكل 2.12 يمثل السلسلة في الذاكرة

في السطر الثالث تم وضع قيمة مبدئية للعنوان name ، هذا العنوان هو نفسه عنوان أول خانة في name التي فيها الحرف S في السطر الرابع يتم وضع عنوان name في المؤشر Ptr.

فإذا فرضنا أن النظام أعطى قيمة لعنوان name = 4001 ، فإن هذه القيمة سوف تكون عنوان الخانة الأولى التي تحتوي على الحرف S، وكذلك يتم وضع 4001 في Ptr في السطر الرابع، ثم تتغير قيم Ptr في السطر التاسع داخل while loop حتى تصل إلى 4013، عندها تكون القيمة للموقع عند هذا العنوان هي 0\ عندها يختل الشرط وبالتالي ينتهي loop ويصل البرنامج إلى النهاية.

في البرنامج السابق كانت طباعة محتويات String حرفاً حرفاً، هذه الطريقة طويلة ويمكن إستبدالها بالأتي:

```
#include<iostream.h>
main ( )
{
char name[ ] = " SALEH NOUMAN";
cout<<  name;
return 0;
}
```

لاحظ الرمز [ ] الذي يدل على السلسلة، وبالتالي فإن أمر الطباعة سوف يظهر كل السلسلة، طبعاً النظام يتعرف على نهاية السلسلة عند وجود الرمز 0\ كما ذكرنا سابقاً.

وكذلك يمكن إدخال مجموعة من الحروف أي قراءة السلسلة جميعاً دون الحاجة إلى loop وهذه من خصائص لغة C++. والبرنامج التالي يوضح هذا المفهوم.

```
#include<iostream.h>
main ( )
{
 char name[25 ];
 cout<< "Enter your name:";
 cin>>name;
 cout<<"Welcome !"<< name;
return 0;
}
```

إذا كانت المدخلات لهذا البرنامج هي SALEH سوف تكون مخرجات البرنامج على النحو:

```
Enter your name: SALEH
Welcome ! SALEH
```

إذا كانت المدخلات لهذا البرنامج هي mohammed سوف تكون مخرجات البرنامج على النحو:

```
Enter your name: mohammed
Welcome ! mohammed
```

لاحظ هنا أننا أستغنينا عن عمل loop للقراءة أوالكتابة، ولاحظ كذلك التعريف name [25]، وأن عدد الحروف التي تم إدخالها فقط 5حروف، أي ليس بالضرورة أن يكون عدد الحروف المدخلة مساوياً لتلك التي ذكرة أثناء التعريف، بل إن النظام يكتفي بالحروف المدخلة، وعندما نضغط على المفتاح enter فإن إشارةً تصدر بأن محتويات الأ Array قد انتهت، عنده يكتفي النظام بهذا الحد من المدخلات.

Functions & Standard library of String 2.4

تمتاز لغة C++ باحتواء مكتبتها على كثير من الدوال التي تسهل للمستخدم إستخدامها دون الحاجة لكتابة مثل هذه الدوال. بل يكفي استدعاؤهن بالمحاورات الخاصة وسوف نوضح بعض هذه الدوال وخاصة التي تتعامل مع String.

هذه الدوال تحتاج إلى الموجه #include<string.h>

```
#include<iostream.h>
#include<string.h>
main ( )
{
 char str1[26 ]= " SALEH NOUMAN";
 char str2 [ ] = " IS YMEN";
 char str3 [30];
 int    i,k;
 i=strlen (str1);
 cout<<endl<<" length of string = "<< i;
 strcpy (str3, str1);
 cout <<endl <<"after copying, string str3= "<< str3;
 k= strcmp (str1, str2);
 cout<<endl <<"on comparing str1 and str2, k="<<k;
 k= strcmp (str3, str1);
 cout<<endl<<" on comparing str3 and str1, k="<<k;
 strcat (str1, str2);
 cout<<endl <<"on concatenation str1= "<< str1;
}
```

مخرجات هذا البرنامج هي

```
Length of string = 12
After copying, string str3 = Is YEMENI
On comparing str1 and str2, k = 1
On comparing str3 and str1, k = 0
On concatenation Str1 = SALEH NOUMAN IS YEMEN.
```

من خلال مخرجات البرنامج السابق نجد الأتي:

1- الدالة strlen ( ) لها محاور واحد وهي ترجع طول المحاور(أي طول string) الذي يرسل كمحاور لها.

2- الدالة strcpy لها محاوران وهي تقوم بعمل نسخة من الثاني في الأول.

3- الدالة strcmp لها محاوران وهي تقوم بمقارنة المحاور الأول مع المحاور الثاني ثم ترجع القيمة

الثاني < الأول	رقم موجب	هذا في بعض الأنظمة والبعض ترجع لفرق بين أول حرفين غير متشابهين.
الثاني > الأول	رقم سالب	
الثاني = الأول	0	

4- الدالة strcat لها محاوران وهي تقوم بلصق نسخة من الثاني إلى نهاية الأول.

اسئلة محلولة

س : ما هي مخرجات البرامج التالية:

1-

```
#include<iostream.h>
#include<string.h>
main ( )
{
char s[ ]= "SALLAH";
cout<<endl <<(strlen(s));
}
```

مخرجات البرامج 6 ، لأن عدد الحروف الموجودة في[ ]s هو 6 "SALLAH"

2-

```
main ( )
{
 char ch[20 ];
 int i;
for (i=0;i <19; i++)
   * (ch+i) = 66;
* (ch+i) = '\0';
 cout<<endl <<ch;
}
```

مخرجات البرامج BBBBBBBBBBBBBBBBBB

3-

```
main ( )
{
 char str[ ]= {48,48,48,48, 48, 48, 48, 48, 48, 48,};
 char *s;     int i ;
 s=str;
 for (i=0; i <9; i++)
{
 if (*s)
  cout<< *s;
 s++;
}
}
```

مخرجات البرامج 000000000

```
//----------------------------------------
#include<string.h>
#include<iomanip.h>
int main()
{
 int i,j;
   char s2[20]=" of sceince",s1[50]="university,"
                    ,s3[20]=" & technology" ,ch[30],*p,*q,fin[5],ch1;
            p=s1;
            q=s2;
           cout<<"ps ="<<p<<q<<endl;
           cout<<"s1 ="<<s1<<endl;
           strcat(s1,s2);
           cout<<"s1 ="<<s1<<endl;
```

```
            strcpy(s2,s1);
            cout<<"s2 ="<<s2<<endl;
            ch1=getchar();
            strcat(s1,s3);
            cin.getline(ch,2);
            i=strlen(s1);
            for(j=1;j<i;j++)
            {
             cout.write(s1,j);
             cout<<endl;
            }
             for(j=i;j>0;j--)
            {
             cout.write(s1,j);
             cout<<endl;
             }
            strcat(ch,p);
            cout<<"p ="<<p<<endl;
            strcat(p,s2);
            cout<<"p ="<<p<<endl;
            p=s1;
            cout<<"s3="<<s3<<endl;
            cout<<"the length of s1 is "<<strlen(s1)<<endl;
            cout<<"s2="<<s2<<endl;
            cout<<"the length of s2 is "<<strlen(s2)<<endl;
            cout<<"s1="<<s1<<endl;
            cout<<"the length of ch is "<<strlen(ch)<<endl;
            cout<<"p= "<<p<<endl;
            cout<<"the position is "<<strstr(p,"log")<<endl;
            // strcat(p,s3);
             cout<<"s1="<<s1<<endl;
             cout<<"the length of s1 is "<<strlen(s1)<<endl;
            // strcat(p,s1);
             cout<<"the length of p is "<<strlen(p)<<endl;
             cout<<"*p= "<<p<<endl;
             i=strcmp(s1,s2);
             cout<<"the compa. "<<i<<endl;
  return 0;
}
```

The output of executing this program as follows:

```
ps =university of science
s1 =university
s1 =university of science
s2 =university of science
u
un
uni
univ
unive
univer
univers
universi
universit
university
```

university
university o
University of
university of
university of s
university of sc
university of sce
university of scei
university of scein
university of sceinc
university of sceince
university of sceince
university of sceince &
university of sceince &
university of sceince & t
university of sceince & te
university of sceince & tec
university of sceince & tech
university of sceince & techn
university of sceince & techno
university of sceince & technol
university of sceince & technolo
university of sceince & technolog
university of sceince & technology
university of sceince & technolog
university of sceince & technolo
university of sceince & technol
university of sceince & techno
university of sceince & techn
university of sceince & tech
university of sceince & tec
university of sceince & te
university of sceince & t
university of sceince &
university of sceince &
university of sceince
university of sceince
university of sceinc
university of scein
university of scei
university of sce
university of sc
university of s
university of
university of
university o
university
university
universit
universi
univers
univer
unive
univ
uni
un
u

p =university of sceince & technology
p =university of sceince & technologyuniversity of sceince
s3=ogy
the length of s1 is 55
s2=nce
the length of s2 is 3
s1=university of sceince & technologyuniversity of sceince

```
the length of ch is 35
p= university of sceince & technologyuniversity of sceince
the position is logyuniversity of sceince
s1=university of sceince & technologyuniversity of sceince
the length of s1 is 55
the length of p is 55
*p= university of sceince & technologyuniversity of sceince
the compa. 7
```

الخلاصة

يعرف المؤشر Pointer بأنه موقع في الذاكرة يمكن أن يحتوي على عنوان موقع أخرى

A pointer is a variable which contains address of another variable.

إن التواصل بين البرنامج الرئسي والدوال أو بين الدوال و الدوال يتم عن طريق المحاورات، و يمكن أن تمرر المحاورات إلى الدوال بإحدى طريقتين:

1. بإرسال قيم المحاورات.
2. بإرسال عناوين المحاورات.

عند التعامل مع قيم المحاورات، فإن القيم الأصلية للمتغيرات المحاورة تضل كما هي عليه دون أن يحصل عليها أي تغير، أما عند التعامل مع عناوين المحاورات، فإن القيم الأصلية للمتغيرات المحاورة تتغير.

اسئلة على الوحدة

س1: ما هي مخرجات هذا البرنامج

```
#include<iostream.h>
int main (    )
{
char C=67;
int  i,*ii,a=23;
ii=&a;
for (i=0;i<10;i++)
 {
   C=C+i;
   a = a+i;
   cout<<* ii <<”  “<<C << endl;
}
return 0;
 }
```

مخرجات هذا البرنامج على النحو

23 C
24 D
25 E
26 F
27 G
28 H
29 J
30 K
31 L

س2: ما هي مخرجات هذا البرنامج

```
#include<iostream.h>
void swap (int * x, int  *y);
void main (    )
{
   int  a = 110,  b= 210;
   swap (&a,&b);
   cout<<endl <<“a = ”<<a;
   cout <<”b = ”<<b;
   return ;
}
void swap (int *x, int  *y)
{
```

```
  int   t;
  cout<<endl<<" x = "<< * x<<"y= "<< *y;
  t = *x;
  *x= *y;
  *y = t;
  cout<<endl<<" x = "<< * x<<"y= "<< *y;
 return ;
}
```

عند تنفيذ هذا البرنامج تكون المخرجات هي:

```
  x=110    y=  210

  x= 210    y=110
  a= 210    b= 110
```

الباب الثالث Sorting and Searching

الهدف من هذا الباب هو الأتي:

1. **التعرف على قضايا الترتيب والبحث**
2. **القدرة على البحث و ترتيب العناصر أو البيانات، ترتيبا تصاعدياً أو ترتيبا تنازلياً**

3 قضايا الترتيب والبحث Sorting and Searching issues

من القضايا المهمة التي يتم مواجهتها في الواقع العملي هما قضيتا **الترتيب للبيانات والبحث عن البيانات.**

3.0 الترتيب Sorting

كثيرا ما تكون الحاجة ماسة إلى ترتيب البيانات، ترتيباً تصاعديا أو ترتيبا تنازلياً، ويشار إلى هذه القضية sorting بترتيب العناصر أو البيانات، ترتيباً تصاعدياً أو ترتيباً تنازلياً.

3 .1 أولا الترتيب التصاعدي

الترتيب التصاعدي، يتم ترتيب العناصر، ترتيباً تصاعدياً، بوضع أصغر عناصر المجموعة في أول موقع في المصفوفة ، ثم الذى يليه مباشرة في الموقع الثاني في المصفوفة، ثم الذى يليه، ثم الذى يليه، وهكذا إلى أن يتم الوصول إلى أكبر عنصر في المجموعة، فيتم وضعه في الموقع الأخير في المصفوفة، وبهذه الطريقة يكون قد تم رتيب العناصر، ترتيباً تصاعدياً.

و الشكل 3.1 يمثل 13 عنصراً غير مرتبة في المصفوفة **a** كما في شكل 3.1

50	61	69	13	6	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.1 تمثيل المواقع فى الذاكرة

أما في الشكل 3.2 فيمثل 13 عنصراً، مرتبة تصاعدياً

10	15	18	20	26	30	35	38	40	50	55	77	80
0	1	2	3	4	5	6	7	8	9	10	11	12

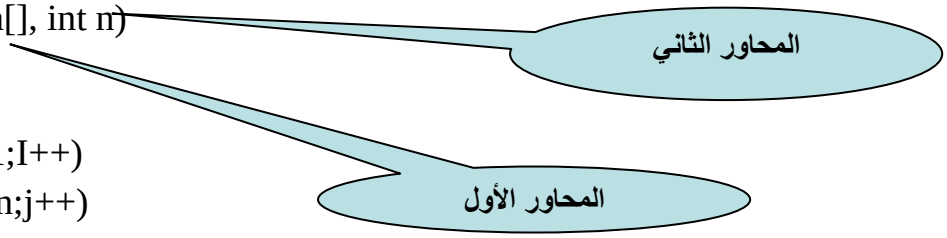
شكل 3.2 يمثل 13 موقعا فى الذاكرة

فإذ فرضنا أن لدينا المصفوفة **a** ، التي تتكون من n عنصرا، عناصرها غير مرتبة، كما هو موضح بشكل 3.3 و يراد ترتيب العناصر تصاعدياً، فيتم كتابة الدالة كما في code 3.1:

61	50	36	13	6	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.3 يمثل 13 موقعاً فى الذاكرة

```
void sort(int a[], int n)
{
 int I,j,t;
 for(I=0;I<n-1;I++)
  for(j=I+1;j<n;j++)
    if(a[I]>a[j])
    {
      t=a[I];
      a[I]=a[j];
     a[j]=t;
    }
return ;
 }
```



code 3.1 يمثل دالة لترتيب عناصر المصفوفة **a** تصاعدياً

عند النظر إلى الدالة sort نجد التالي: إن الدالة تحتوى على حلقتين، إحداهما داخلية، والأخرى خارجية، الحلقة الخارجية ممثلة بالعداد I، والحلقة الداخلية ممثلة بالعداد j، فيبدأ التنفيذ بالحلقة الخارجية، ثم بالحلقة الداخلية، وأن العنصر في الموقع الأول من المصفوفة **a** أكبر من العنصر في الموقع الثاني، وعلى ذلك يتم تنفيذ الأوامر داخل الجملة الشرطية if ، ومن خلال هذه الأوامر، يتم التبادل بين محتويات الموقعين، وتكون محتويات المصفوفة **a** على شكل 3.4:

50	61	36	13	6	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.4 يمثل 13 موقعا فى الذاكرة

ثم تنقل الحلقة الداخلية إلى الموقع الثالث من المصفوفة **a** ، وفى هذه الحالة قيمة الموقع الثالث من المصفوفة **a** ليس أكبر من قيمة العنصر في الموقع الأول، وعلى ذلك يتم تنفيذ الأوامر داخل الجملة الشرطية if ، ومن خلال هذه الأوامر، يتم التبادل بين محتويات الموقعين، وتكون محتويات المصفوفة **a** كما هو موضح بشكل 3.5:

36	61	50	13	6	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.5 يمثل 13 موقعا فى الذاكرة

ويتم نقل الحلقة الداخلية إلى الموقع الرابع، فيجد قيمة العنصر في الموقع الرابع أصغر من قيمة العنصر في الموقع الأول، وعلى ذلك يتم تنفيذ الأوامر داخل الجملة الشرطية if ، ومن خلال هذه الأوامر، يتم التبادل بين محتويات الموقعين، وتكون محتويات المصفوفة **a** كما هو موضح بشكل 3.6

13	61	50	36	6	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.6 يمثل 13 موقعا فى الذاكرة

ثم تستمر العملية على هذا النحو حتى تصل الحلقة الداخلية إلى نهايتها، ويكون شكل محتويات المصفوفة **a** كما هو موضح بشكل 3.7

6	61	50	13	36	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.7 يمثل 13 موقعاً في الذاكرة

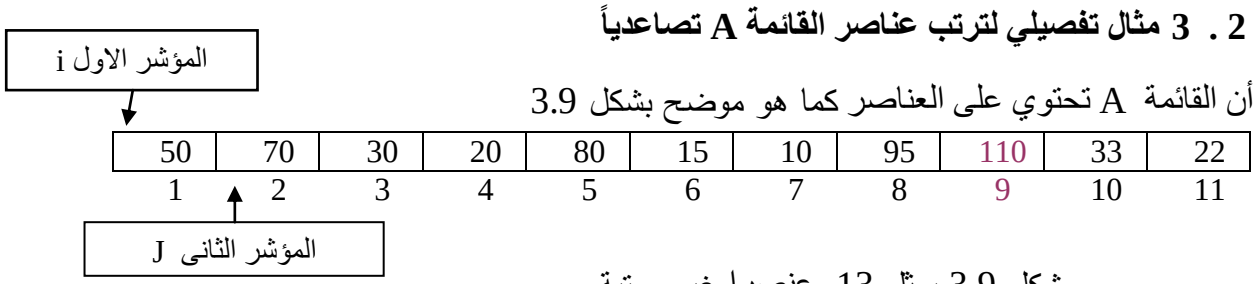
عندما تصل الحلقة الداخلية إلى نهايتها، يكون أصغر عنصرا في المصفوفة قد تم وضعه في الموقع الأول من المصفوفة، بينما تكون بقية عناصر المصفوفة غير مرتبة، وعلى ذالك تكون الحاجة إلى اعادة التكرار مرةً ومرة على هذا النحو، حتى تصل الحلقة الخارجية إلى نهايتها، عندها نجد أن كل عناصر المصفوفة مرتبة تصاعدياً كما هو موضح بشكل 3.8

6	8	10	11	13	22	35	36	50	55	61	77	100
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.8 يمثل 13 موقعاً فى الذاكرة

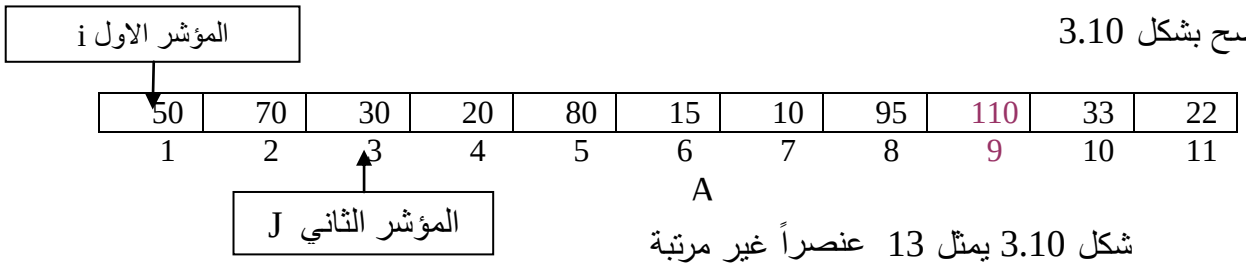
1 . 2 . 3 مثال تفصيلي لترتب عناصر القائمة A تصاعدياً

لنفرض أن القائمة A تحتوي على العناصر كما هو موضح بشكل 3.9



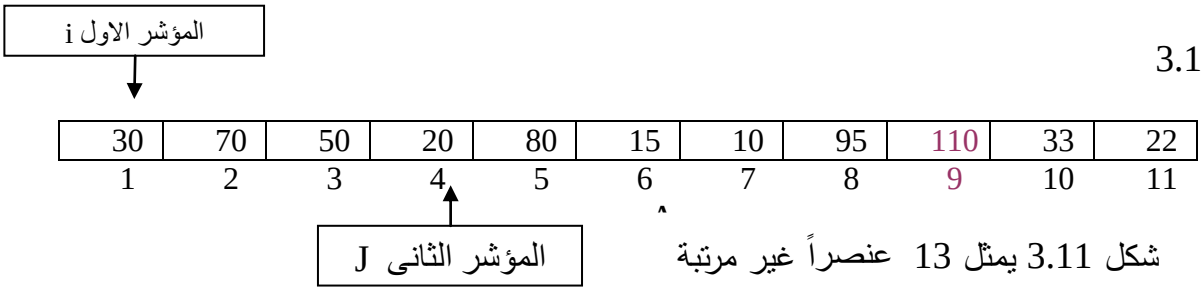
شكل 3.9 يمثل 13 عنصرا غير مرتبة

في الشكل السابق نلاحظ أنه في البداية أي في الحلقة الأولى من ال loop ، قد تم وضع المؤشر الأول i بحيث يشير إلى أول عناصر القائمة (قيمته هنا 50) و قد تم وضع المؤشر الثاني j بحيث يشير إلى العنصر الذي يلي المؤشر الأول في القائمة ، (قيمته هنا 70) ، فنجد أن العنصر الذي يشير إليه المؤشر الثاني أكبر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك لا يتم تبديل العنصرين و يكون وضع المصفوفة على ما كان عليه سابقا، و يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.10



شكل 3.10 يمثل 13 عنصراً غير مرتبة

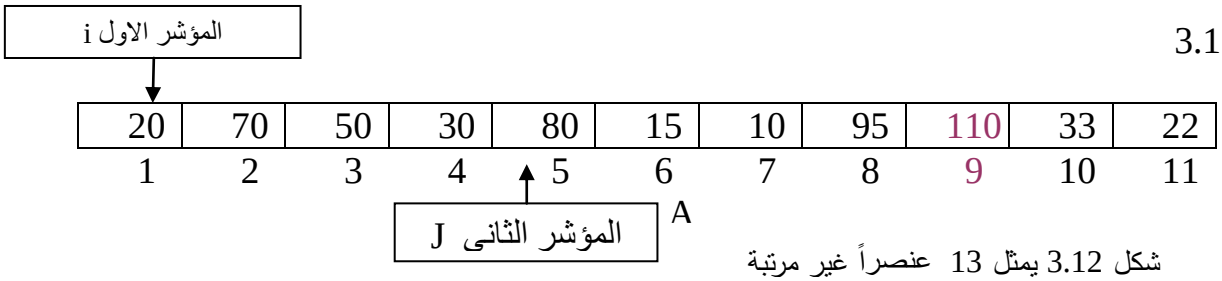
أما في الحلقة رقم 2 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أصغر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك يتم تبديل العنصرين و يكون وضع المصفوفة قد تغير على ما كان عليه سابقا، و يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.11



شكل 3.11 يمثل 13 عنصراً غير مرتبة

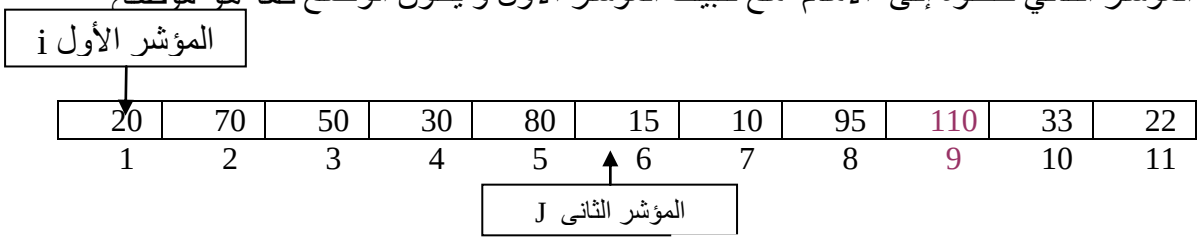
أما في الحلقة رقم 3 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أصغر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك يتم تبديل العنصرين و يكون وضع المصفوفة قد تغير على ما كان

عليه سابقا، و يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.12



شكل 3.12 يمثل 13 عنصراً غير مرتبة

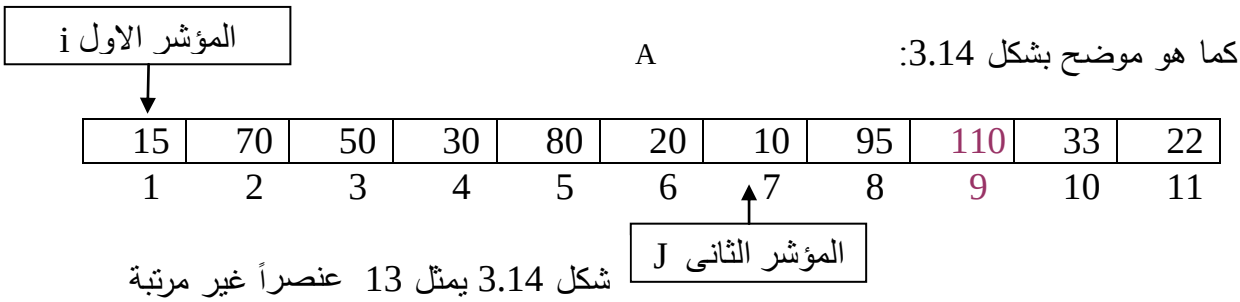
و أما في الحلقة رقم 4 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أكبر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك لا يتم تبديل العنصرين و يكون وضع المصفوفة على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.13



شكل 3.13 يمثل 13 عنصراً غير مرتبة

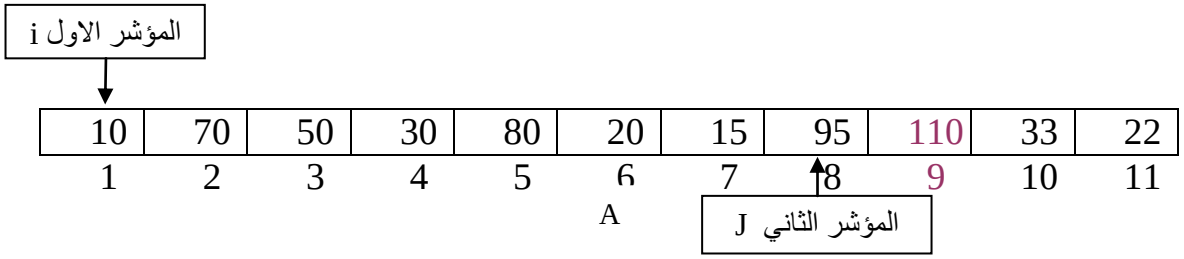
و أما في الحلقة رقم 5 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أصغر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك يتم تبديل العنصرين و يكون وضع المصفوفة قد تغير على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.14:

A



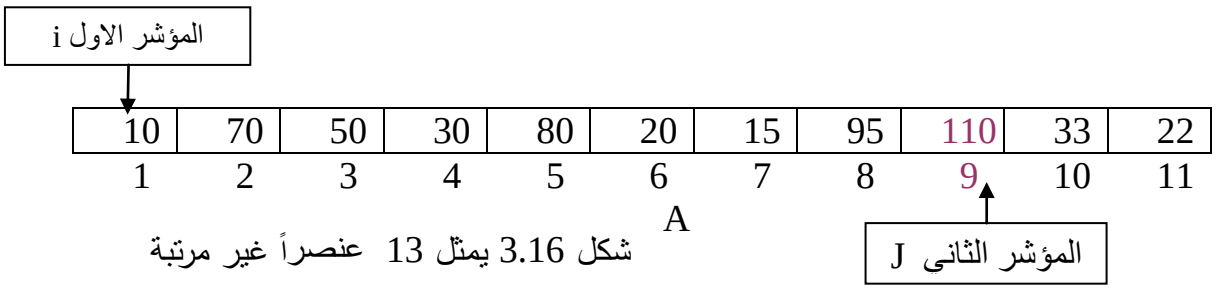
شكل 3.14 يمثل 13 عنصراً غير مرتبة

أما في الحلقة رقم 6 من ال loop الداخلية، فنجد أن العنصر الذي يشير إليه المؤشر الثاني أصغر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك يتم تبديل العنصرين و يكون وضع المصفوفة قد تغير على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.15:



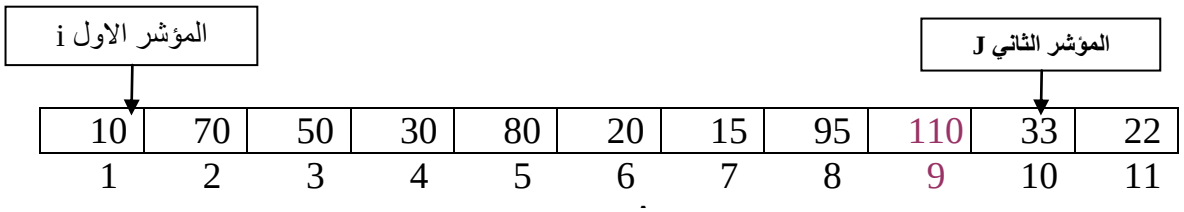
شكل 3.15 يمثل 13 عنصراً غير مرتبة

و أما في الحلقة رقم 7 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أكبر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك لا يتم تبديل العنصرين و يكون وضع المصفوفة على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.16:



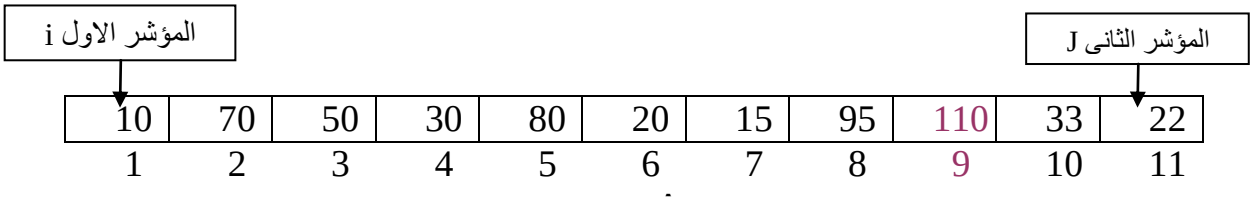
شكل 3.16 يمثل 13 عنصراً غير مرتبة

و أما في الحلقة رقم 8 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أكبر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك لا يتم تبديل العنصرين و يكون وضع المصفوفة على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.17



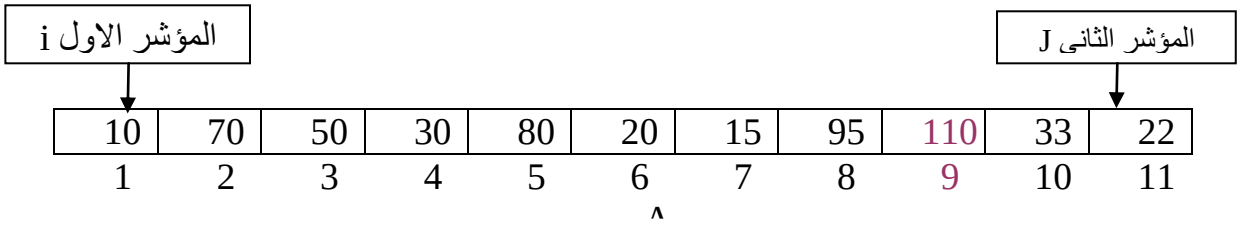
شكل 3.17 يمثل 13 عنصراً غير مرتبة

و أما في الحلقة رقم 9 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أكبر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك لا يتم تبديل العنصرين و يكون وضع المصفوفة على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.18



شكل 3.18 يمثل 13 عنصراً غير مرتبة

و أما في الحلقة رقم 10 من ال loop الداخلية، نجد أن العنصر الذي يشير إليه المؤشر الثاني أكبر من العنصر الذي يشير إليه المؤشر الأول، وعلى ذالك لا يتم تبديل العنصرين و يكون وضع المصفوفة على ما كان عليه سابقا، ثم يتم تقدم المؤشر الثاني خطوة إلى الأمام مع تثبيت المؤشر الأول و يكون الوضع كما هو موضح بشكل 3.19

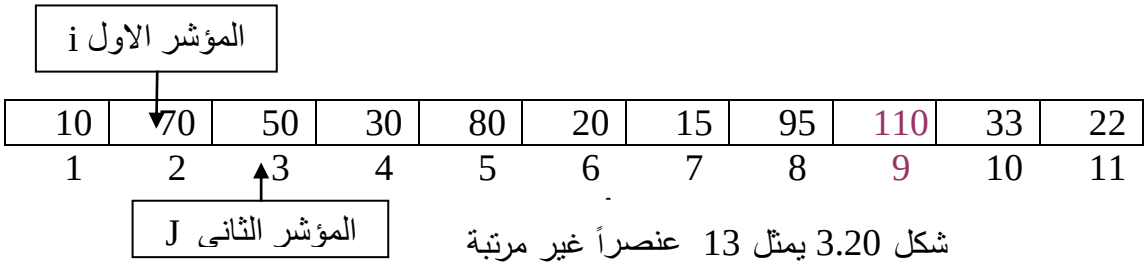


شكل 3.19 يمثل 13 عنصراً غير مرتبة

ثم يتم زيادة قيمة المتغيرj بمقدار 1 وتتم محاولة تنفيذ الدورة رقم 11 من ال for loop وذالك بفحص المتغير j (وقيمته في هذه الحالة 11) و مقارنته مع المتغيرn الذي قيمته 11، فتكون قيمة المتغيرj ليس اقل من المتغير n، و بالتالي لا يتم دخول الfor loop الداخلية، و يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

و من الشكل السابق نجد ان أصغر عنصرا في المصفوفة وقيمته 10قد تم نقلة إلى بداية المصفوفة، و هذا وضعه الطبيعي، لان المطلوب هو ترتب عناصر القائمة تصاعديا.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام ثم يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، و يكون الوضع كما هو موضح بشكل 3.20



شكل 3.20 يمثل 13 عنصراً غير مرتبة

ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 3 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في المجموعة وهو هنا 15 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 4 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في تلك المجموعة وهو هنا 20 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول ثم يتم تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 5 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في المجموعة وهو هنا 22 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية ال for loop الداخلية.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 6 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في تلك المجموعة وهو هنا 30 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 7 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في تلك المجموعة وهو هنا 33 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

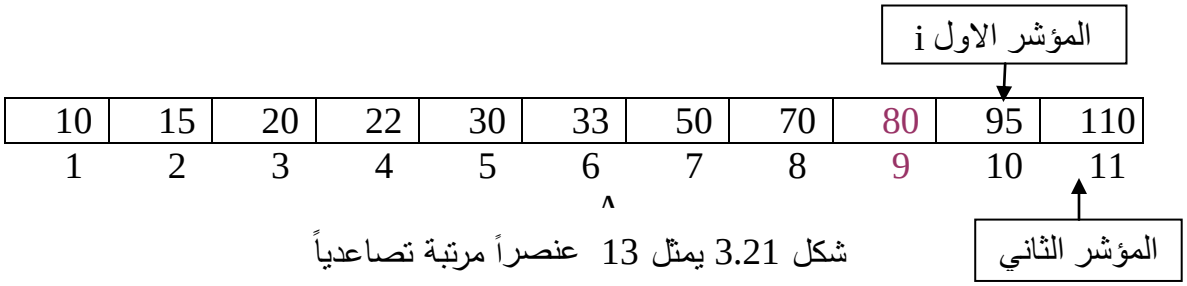
ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 8 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في تلك المجموعة وهو هنا 70 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 9 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل

هو نقل قيمة أصغر عنصر في تلك المجموعة وهو هنا 80 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية.

ثم ينتقل التنفيذ إلى الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، ثم يتكرر تنفيذ حلقات ال loop الداخلية مبتدئة من الموقع رقم 10 و منتهية بالموقع رقم 11 ، و ينتج عن ذالك تبديل في قيم مواقع عناصر المصفوفة، أهم تلك التباديل هو نقل قيمة أصغر عنصر في تلك المجموعة وهو هنا 95 إلى بداية المصفوفة، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الداخلية. ثم ينتقل التنفيذ إلى محاولة دخول الfor loop الخارجية ويتم تقديم المؤشر الأول خطوة إلى الأمام، و يتم وضع المؤشر الثاني بحيث يشير إلى العنصر التالي للمؤشر الأول، و بذالك يكون قد وصل التنفيذ إلى نهاية الfor loop الخارجية،

عندها يكون قد وصل التنفيذ إلى نهاية الدالة، و يكون الوضع كما هو موضح بشكل 3.21



شكل 3.21 يمثل 13 عنصراً مرتبة تصاعدياً

الخلاصة

يمكن ترتب عناصر قائمة (مصفوفة أحادية) تصاعديا أو تنازلياً، وهناك أكثر من طريقة لترتب عناصر المصفوفة.

اسئلة عن ترتب عناصر قائمة (مصفوفة أحادية) تصاعديا

س1 - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقأما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في سطر) ثم يطبع سطر جديد، ثم يطبع عناصر المصفوفة مرتبة تصاعدياً.

س2 - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً من نوع int ، ثم يستدعي دالة لتوليد أرقأما عشوائية في المصفوفة، ثم يستدعي دالة لطباعة محتويات المصفوفة (كل 8 عناصر في سطر) ، ثم يستدعي دالة لترتيب عناصر المصفوفة تصاعدياً ، ثم يستدعي دالة الطباعة بعد الترتيب.

3 . 3 ثانيا الترتيب التنازلي

الترتيب التنازلي، ويتم ترتيب العناصر، ترتيبا تنازلياً، بحيث يتم وضع العنصر الأكبر في أول موقع في المصفوفة، ثم الذى يليه مباشرة في الموقع الثاني، ثم الذى يليه، ثم الذى يليه، وهكذا إلى ان يتم الوصول إلى أصغر عنصر، فيتم وضعه في الموقع الاخير في المصفوفة. وبهذه الطريقة يتم ترتيب العناصر، ترتيباً تنازلياً.

الشكل 3.22 يمثل 13 عنصراً، هذه العناصر مرتبة ترتيباً تنازلياً .

100	77	61	55	50	36	35	22	13	11	10	8	6
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.22 يمثل 13 عنصراً مرتبة تنازلياً

أمثلة على الترتيب التنازلي

س$_1$ – اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً، من نوع int ، ثم يولد في هذه المصفوفة أرقأما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في سطر) ثم يطبع صف جديد ثم يطبع عناصر المصفوفة مرتبة تنازلياً.

س$_2$ – اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 موقعاً ، من نوع int ، ثم يستدعي دالة لتوليد أرقأما عشوائية في المصفوفة، ثم يستدعي دالة لطباعة محتويات المصفوفة (كل 8 عناصر في سطر) ، ثم يستدعي دالة لترتيب عناصر المصفوفة تنازلياً ، ثم يستدعي دالة الطباعة بعد الترتيب.

مقارنة بين ثلاثة أنواع من الترتيب لعناصر مصفوفة

هذا البرنامج التالي يقوم بمقارنة بين ثلاثة أنواع من الترتيب لمحتويات مصفوفة من حيث عدد العمليات و من حيث كمية الوقت اللازم لتنفيذ العمليات.

```
#include<iostream.h>
#include<stdlib.h>
#include<time.h>
#include<iomanip.h>
const n=2000;
 void sort1( int [] ,int );
 void sort2( int [] ,int );
 void sort3( int [] ,int );
 void pt( int [] ,int );
 void gt( int [] ,int );
 clock_t start,end;
 int b[n], c[n];
 int main( )
 {
```

```
        int a[n];
          gt(a,n);
         cout<<endl;
         start=clock();
         sort1(a,n);
         end=clock();
         cout<<"The required en. time in sort1 is "<<(end-start)/CLK_TCK<<endl;
         cout<<endl;
         start=clock();
         sort2(b,n);
         end=clock();
         cout<<"The required en. time in sort2 is "<<(end-start)/CLK_TCK<<endl;
         cout<<endl;
         start=clock();
         sort3(c,n);
         end=clock();
         cout<<"The required en. time in sort3 is "<<(end-start)/CLK_TCK<<endl;
         cout<<endl;
        return 0;
}
  //-------------------------------------------------------------
        void sort1( int a[] ,int n)
        {
         int i,c,n1,j;
         long m=1;
         n1=n-1;
         for(i=0;i<n1;i++)
         { m++;
                  for(j=i+1;j<n;j++)
                  { m++;
                           if(a[i]<a[j])
                           {
                            c=a[i];
                            a[i]=a[j];
                            a[j]=c;
                                    m=m+4;
                           }
                  }
         }
          cout<<" m in sort1 is "<<m<<endl;
          return ;
         }
         //*************************************************
        void sort2( int a[] ,int n)
        {
          int i,c,t,n1;
          long m=1;
          n1=n-1;
          do
          {   m++;
```

```
      t=0;
           for(i=0;i<n1;i++)
           { m++;
                    if(a[i]<a[i+1])
                    {
                      c=a[i];
                      a[i]=a[i+1];
                       a[i+1]=c;
                       t=1;
                      m=m+5;
                     }
             }
    }  while(t!=0);
 cout<<" m in sort2 is "<<m<<endl;
 return ;
}
//-------------------------------------------------------------
void sort3( int a[] ,int n)
{
  int i,n1,j,mi,mv;
  long m=1;
  n1=n-1;
  for(i=0;i<n1;i++)
  {
          mi=i; mv=a[i];
          m=m+3;
           for(j=i+1;j<n;j++)
           { m++;
                    if(a[j]>mv)
                    {
                     mv=a[j];
                     mi=j;
                     m=m+3;
                    }
           }
              a[mi]=a[i];
             a[i]=mv;
            m=m+2;
  }
 cout<<" m in sort3 is "<<m<<endl;
 return ;
}
//-----------------------
void pt( int a[] ,int n)
{
  int i;
  for(i=0;i<n;i++)
  {
                    cout<<setw(5)<<a[i];
                    if((i+1)%10==0)
```

```
                cout<<endl;
  }
 return ;
}
 void gt( int a[] ,int n)
{
  int i;
  srand(time(NULL));
  for(i=0;i<n;i++)
   {
        a[i]=rand()%10005;
        b[i]=a[i];
        c[i]=a[i];
 }
 return ;
}
```

4 . 3 linear search البحث الخطي

المقصود بالبحث الخطي هنا، هو إنجاز عملية البحث المتسلسل عن عنصر ما، وسط بيئة ما أو وسط مجموعة من العناصر، مثل وسط مصفوفة أحادية أو ثنائية.

وفي الحقيقة توجد طرق عديدة لإنجاز عملية البحث أهمهمن طريقتان هما البحث الخطي و البحث الثنائي، وكل طريقة لها مميزات وعيوب، ونحن لن نتطرق هنا لشرح طرق أنواع البحث، بل سيتم الإكتفاء بذكر الطريقة العادية، والتي تسمى بالبحث الخطي، و هناك طريقتان لإنجاز البحث الخطي هما **البحث الخطي بالإتجاه نحو الأمام، و البحث الخطي بالإتجاه نحو الخلف.**

4.3.1 أولاً البحث الخطي بالإتجاه نحو الأمام

يمكن تلخيص هذه الطريقة كالتالي:

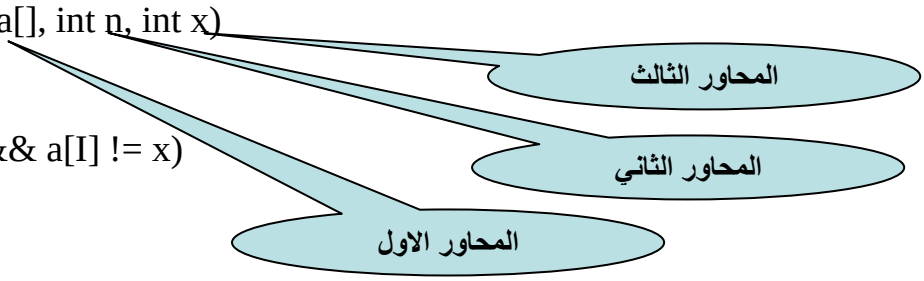
1. يتم جعل مؤشرا يشير إلى بداية قائمة العناصر أو مجموعة العناصر المراد البحث عن العنصر فيها.
2. يتم المقارنة بين العنصر الذى يشير إليه المؤشر في القائمة وبين العنصر المراد البحث عنه من حيث التساوي، فإذا كنت نتيجة المقارنة هي التساوي، يتم التوقف عن البحث، و يتم إعطاء إشارة بذالك، أما إذا كنت نتيجة المقارنة هي عدم التساوي فيتم انتقال المؤشر خطوةً نحو الأمام.
3. يتم التوقف عن البحث إذا جاوز المؤشر حدود الوسط أو حدود البيئة المراد البحث عن العنصر فيها، ثم يتم إعطاء إشارةً بذالك ، أو يتم الإنتقال إلى الخطوة رقم 2.

بهذه الطريقة يمكن أن تنجز عملية البحث عن عنصر ما وسط بيئة محدودة، ويتم إتخاذ القرار المناسب، هل العنصر المراد البحث عنه موجوداً أم غير موجود، أي أنه يجب الأخذ بعين الإعتبار وجود العنصر وعدم وجود العنصر. و للتوضيح أكثر يتم ذكر المثال التالي:

مثال 1- اكتب دالة تبحث عن العنصر x وسط مصفوفة أحادية من نوع int ، تتكون من n موقعاً، ثم تطبع رقم الموقع إذا كانت x موجودة، أو تطبع 1- إذا كانت x غير موجودة في المصفوفة.

الحل

```
void search(int a[], int n, int x)
{
 int I=0;
  while( I<=n && a[I] != x)
     I++;
 if(I==n)
  cout<<-1;
 else
 cout<<I;
}
```



code 3.2 يمثل دالة تبحث عن العنصر x وسط المصفوفة **a**

عند تتبع هذه الدالة search نجد الأتي:

- تحتوى الدالة في أول سطر على الكلمة void و التي تدل على أن الدالة لا ترجع شيء من البيانات، ثم نجد الكلمة search و التي تدل على اسم الدالة، ثم نجد العبارة (int a[ ], int n, int x)، حيث a[ ] هو اسم المصفوفة المراد البحث فيها، ثم نجد العبارة int n حيث n هي حجم المصفوفة a، ثم نجد العبارة int x حيث x هي قيمة العنصر المراد البحث عنه.
- تحتوى الدالة في السطر الثالث على ;I=0 وهو عبارة عن مؤشر يشير إلى بداية المصفوفة.
- تحتوى الدالة في السطر الرابع على (while(I<=n && a[I] != x وهو عبارة عن حلقة loop، يتم الخروج من هذه الحلقة إذا لم يتحقق أحد الشرطين، I<n، a[I] != x، و يتم الدخول في هذه الحلقة إذا تحقق كلا الشرطين، I<n، a[I] != x.
- تحتوى الدالة في السطر الخامس على الجملة ;I++ وهو عبارة عن نقل المؤشر خطوةًا نحو الأمام.
- أما بقية الاسطر فهي عبارة عن طريقة للتحقق من وجود العنصر في المصفوفة a من عدمه، فإذا ساوت قيمة المؤشر I قيمة n فإن x غير موجودة، أما ذا لم تساو قيمة المؤشر I قيمة n فإن x موجودة في المصفوفة a، شكل 3.23 وطبقاً لقيمة I تطبع الإشارة المناسبة.

6	61	50	13	36	8	10	11	55	100	35	77	22
0	1	2	3	4	5	6	7	8	9	10	11	12

شكل 3.23 يمثل المصفوفة **a**

فإذا فرضنا أن لدينا هذه المصفوفة، شكل 3.23 و يراد البحث عن العنصر 45 في هذه المصفوفة، فإن نتيجة البحث سوف تكون1- ، وسوف تكون قيمة المؤشر I هي 13 لأن العنصر 45 غير موجود ، أما إذا بحثنا عن العنصر 8 في نفس المصفوفة، فإن نتيجة البحث سوف تكون 5 وهي قيمة المؤشر I ، أي أن العنصر موجود في الموقع رقم 5 في المصفوفة a.

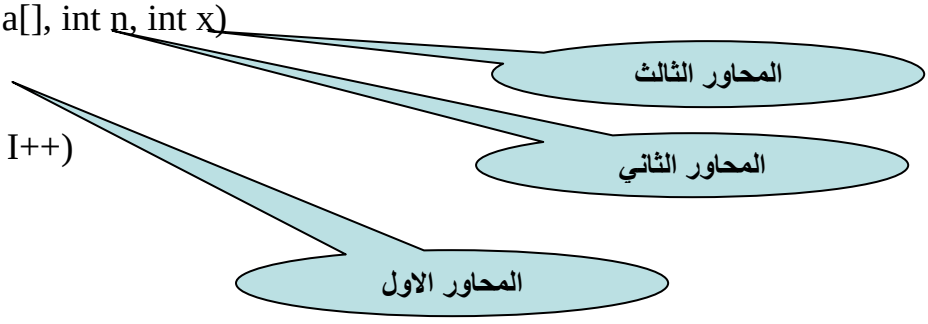
4.3.2 ثانيًا البحث الخطي بالإتجاه نحو الخلف.

ويمكن إنجاز البحث الخطي ايضا بالطريقة التالية:

1. إجعل موشراً يشير إلى نهاية قائمة العناصر أو مجموعة العناصر المراد البحث عن العنصر فيها.
2. قارن بين العنصر الذى يشير إليه المؤشر في القائمة وبين العنصر المراد البحث عنه من حيث التساوى، فإذا كنت نتيجة المقارنة هي التساوى، يتم التوقف عن البحث، و يتم إعطاء إشارة بذالك، أما إذا كنت نتيجة المقارنة هي عدم التساوى يتم نقل المؤشر خطوةً نحو الخلف.
3. يتم التوقف عن البحث إذا جاوز المؤشر الوسط أو البيئة المراد البحث عن العنصر فيها و يتم إعطاء إشارةً بذالك ، أو يتم الإنتقال إلى الخطوة رقم 2.

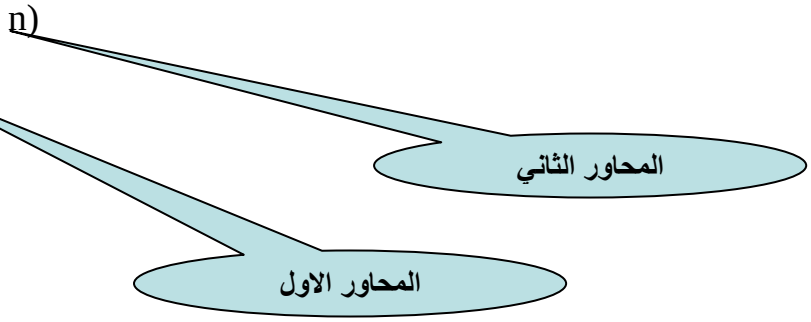
مثال 2- اكتب دالة تبحث عن عدد وجود العنصر x وسط مصفوفة أحادية من نوع int ، تتكون من n موقعاً، ثم تطبع عدد تكرار العنصر x إذا كانت x موجودة، أو تطبع 1- إذا كانت x غير موجودة في المصفوفة.

```
void search(int a[], int n, int x)
{
 int I,C=0;
  for (I=0; I<n; I++)
   if( a[I]==x)
     C++;
 if(C==0)
  cout<<-1;
 else
 cout<<C;
}
```



مثال3 – اكتب دالة تطبع عدد العناصر الزوجية في مصفوفة أحادية من نوع int ، تتكون من n موقعاً .

```
void search(int a[], int n)
{
  int I,C=0;
   for (I=0; I<n; I++)
    if( a[I]%2==0)
      C++;
    cout<<C;
}
```

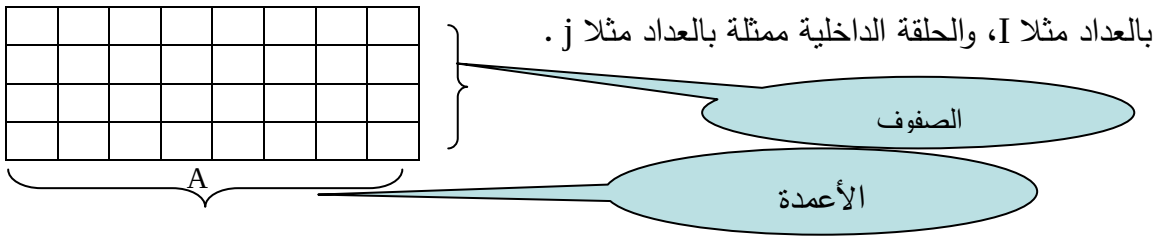


مثال4 – اكتب دالة تطبع عدد العناصر الزوجية وعدد العناصر الفردية في مصفوفة أحادية من نوع int ، تتكون من n موقعاً .

```
void search(int a[], int n)
{
  int I,C=0;
   for (I=0; I<n; I++)
     if( a[I]%2==0)
       C++;
    cout<<"even no is "<<C<<"  odd no "<<(n-C);
return ;
}
```

3 . 5 البحث الخطي عن عنصر ما في مصفوفة ثنائية

للبحث عن عنصر ما في مصفوفة ثنائية، فيتم البحث في كل صف أو في كل عمود من المصفوفة، فإذا فرضنا أن لدينا مصفوفة ثنائية A، شكل 3.24 و يراد البحث عن عنصر ما فيها، فإننا نحتاج إلى حلقتين، إحداهما داخلية، وتمثل الأعمدة في المصفوفة، والأخرى خارجية وتمثل الصفوف في المصفوفة، الحلقة الخارجية ممثلة بالعداد مثلا I، والحلقة الداخلية ممثلة بالعداد مثلا j .



شكل 3.24 يمثل مصفوفة ثنائية A

فيبدأ التنفيذ بوضع مؤشر الحلقة الخارجية على بداية الصف الأول في المصفوفة، ثم يتم وضع مؤشر الحلقة الداخلية على بداية العمود الأول بالمصفوفة، ثم تتم المقارنة بين العنصر الذى يشير إليه المؤشرين(مؤشر الصفوف ومؤشر الاعمدة)، في القائمة وبين العنصر المراد البحث عنه من حيث التساوى، فإذا كنت نتيجة المقارنة هي التساوى، يتم التوقف عن البحث، و يتم إعطاء إشارة بذالك، أي قيمة المؤشرين(مؤشر الصفوف ومؤشر الاعمدة)، أما إذا كنت نتيجة المقارنة هي عدم التساوى فيتم نقل مؤشر الحلقة الداخلية خطوةً نحو الأمام، ثم تتم المقارنة، إلى أن يصير مؤشر الحلقة الداخلية في أعلى قيمة له، أو تكون نتيجة المقارنة هي التساوى فيتم التوقف عن البحث و يتم إعطاء إشارة بذالك، ثم يتم نقل مؤشر الحلقة الخارجية خطوةًا نحو الأمام، ثم يوضع مؤشر الحلقة الداخلية على بداية العمود الأول من الصف الحالي بالمصفوفة، ثم تتم المقارنة إلى أن يصير مؤشر الحلقة الداخلية في أعلى قيمة له.

وتستمر العملية في المقارنة إلى أن يصير مؤشر الحلقة الخارجية في أعلى قيمة له أو تكون نتيجة المقارنة هي التساوى فيتم التوقف عن البحث، و يتم إعطاء إشارة بذالك، أي قيمة المؤشرين.

3.6 الإستدعاء الذاتي للدوال Recursion

يعتبر الإستدعاء الذاتي من الأساليب المهمة في برمجة الحاسوب وخاصة في الذكاء الصناعي، حيث يمكن إنجاز مهام كبيرة بكتابة كمية صغيرة من الأوامر.

3.6.1 تعريف الإستدعاء الذاتي

الإستدعاء الذاتي لدالة عبارة عن إستدعاء الدالة لنفسها، ويتم ذالك عند ذكر اسم الدالة داخل الدالة نفسها.

لا بد لكل دالة فيها إستدعاء ذاتي أن تحتوى على شرط توقف، هذ الشرط يعمل على توقف الإستدعاء للدالة.

عند إستدعاء الدالة لنفسها يتم إيجاد بيئة جديدة يتم التعامل معها، كما يتم حفظ قيم محاورات المتغيرات الحالية في Stack وذالك عند الإستدعاء الذاتي.

وإيجاد البيئة الجديدة ممكن أن نتخيلها بأن الدالة تنفذ مرةً أخرى من البداية وبقيم جديدة للمتغيرات.

ولتوضيح ذالك نذكر المثال التالي:

لنفرض أن لدينا مصفوفة إسمها A وتحتوى على 10 عناصر كما هو موضح في شكل 3.25

10	15	20	25	30	35	40	45	50	55
0	1	2	3	4	5	6	7	8	9

شكل 3.25 يمثل مصفوفة احادية

فإذا أردنا طباعة عناصر المصفوفة بإستخدام الإستدعاء الذاتي فيتم كتابة هذه الدالة التالية pt:

```
void pt(int A[], int n)
{
 if(n<0)
 return ;
 pt(A,n-1);
 cout<<setw(4)<< A [n];
 }
```

شرط توقف الدالة

الإستدعاء الذاتى للدالة

أمر الطباعة

عند تنفيذ هذه الدالة نجد أن المخرجات كما في شكل 3.26 :

10 15 20 25 30 35 40 45 50 55

شكل 3.26 يمثل مخرجات الاستدعاء الذاتي

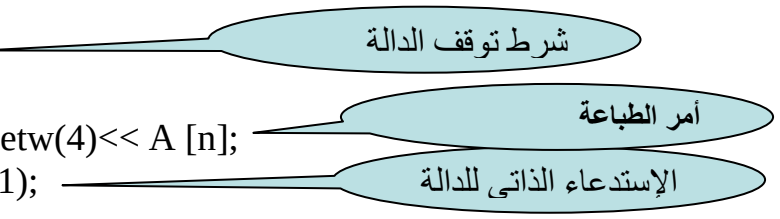
الذي حدث أن أمر الطباعة موجود بعد أمر الإستدعاء الذاتي، وعليه فإنه لن يتم الوصول إليه عند تنفيذ أمر الإستدعاء الذاتي، وبالتالي يتم تخزين أمر الطباعة في Stack ثم يتم تنفيذ إستدعاء الدالة مرةً أخرى من البداية وبقيمة جديدة للمتغير n أي بالقيمة 7، وفي كل مرة يتم تخزين أمر الطباعة في Stack وبقيمة جديدة للمتغير n ، وهكذا إلى أن تصبح قيمة المتغير n صفر، عندها يتحقق الشرط

ومن ثم يتوقف الإستدعاء الذاتي للدالة، ثم يتم عمل pop لأمر الطباعة من Stack ثم يتم تنفيذه وبالتالي تظهر المخرجات على هذا النحو:

10 15 20 25 30 35 40 45 50 55

ولكن ماذا يحدث لو أن أمر الطباعة سبق أمر الإستدعاء الذاتي للدالة ؟

```
void pt(int A[], int n)
{
 if(n<0)
 return ;
 cout<<setw(4)<< A [n];
  pt(A,n-1);
}
```



عند تنفيذ هذه الدالة نحد أن المخرجات كما في شكل 3.27 :

55 50 45 40 35 30 25 20 15 10

شكل 3.27 يمثل مخرجات الاستدعاء الذاتي

والسبب في ذالك أن عملية الطباعة هنا تتم قبل عملية الإستدعاء الذاتي، أما في المثال السابق فإن عملية الطباعة هناك تتم بعد عملية الإستدعاء الذاتي.

مثال آخر

مستخدما الإستدعاء الذاتي، اكتب دالة يرسل إليها عددين صحيحين ثم تقوم هذه الدالة بحساب القاسم المشترك الأكبر لهاذين العددين.

الحل:

تعريف القاسم المشترك الأكبر لعددين

القاسم المشترك الأكبر لعددين هو أكبر عدد، العددان يقبلا القسمة عليه بدون باقي.

```
// This function to compute the Greatest common multiply&y!=0
int gcd(int x,int y)
{
 if(x%y==0 )
 return y ;
 return gcd(y,x%y );
}
```

مثال آخر

مستخدما الإستدعاء الذاتي، اكتب دالة يرسل إليها عددين صحيحين ثم تقوم هذه الدالة بحساب المضاعف المشترك الأصغر لهاذين العددين.

الحل:

تعريف المضاعف المشترك الأصغر لعددين

المضاعف المشترك الأصغر لعددين هو أصغر عدد يقبل القسمة على العددين بدون باقي.

```
// This function to compute the smallest common multiply of x & y
    // x and y!=0
    int scm(int i, int x, int y)  //  i=
   {
    if(x%y==0)
    return x;
      return scm(i,i+x,y);
    }
```

مثال آخر

مستخدما الإستدعاء الذاتي، اكتب دالة يرسل إليها عدد صحيح ثم تقوم هذه الدالة بحساب مضروب العدد.

الحل:

```
    // This function to compute the factorial of x
    int fact(int x)
    {
      if(x <=1 )
         return 1 ;
       return x*fact(x-1);
    }
```

الخلاصة

من القضايا المهمة التي يتم مواجهتها في الواقع العملي هما قضيتا **الترتيب للبيانات والبحث عن البيانات.**

كثيرا ما تكون الحاجة ماسة إلى ترتيب البيانات، ترتيباً تصاعديا أو ترتيبا تنازلياً،

الترتيب التصاعدي، يتم ترتيب العناصر، ترتيباً تصاعدياً، بوضع أصغر عناصر المجموعة في أول موقع في المصفوفة ، ثم الذى يليه مباشرة في الموقع الثاني في المصفوفة، ثم الذى يليه، ثم الذى يليه، وهكذا إلى أن يتم الوصول إلى أكبر عنصر في المجموعة، فيتم وضعه في الموقع الأخير في المصفوفة، و العكس بالنسبة التنازلي.

للبحث عن البيانات توجد طرق أهمهمن طريقتان هما البحث الخطي و البحث الثنائي، وكل طريقة لها مميزات وعيوب، و هناك طريقتان لإنجاز البحث الخطي هما **البحث الخطي بالإتجاه نحو الأمام، و البحث الخطي بالإتجاه نحو الخلف.**

إن التواصل بين البرنامج الرئسي والدوال أو بين الدوال و الدوال يتم عن طريق المحاورات، حيث يمكن أن تمرر المحاورات إلى الدوال بإحدى طريقتين:

1. بإرسال قيم المحاورات. عند التعامل مع قيم المحاورات، فإن القيم الأصلية للمتغيرات المحاورة تضل كما هي عليه دون أن يحصل عليها أي تغير.
2. بإرسال عناوين المحاورات. أما عند التعامل مع عناوين المحاورات، فإن القيم الأصلية للمتغيرات المحاورة تتغير.

الإستدعاء الذاتي لدالة عبارة عن إستدعاء الدالة لنفسها، ويتم ذالك عند ذكر اسم الدالة داخل الدالة نفسها. يعتبر الإستدعاء الذاتي من الأساليب المهمة في برمجة الحاسوب وخاصة في الذكاء الصناعي، حيث يمكن إنجاز مهام كبيرة بكتابة كمية صغيرة من الأوامر.

لا بد لكل دالة فيها إستدعاء ذاتي أن تحتوى على شرط توقف، هذ الشرط يعمل على توقف الإستدعاء للدالة.

عند إستدعاء الدالة لنفسها يتم إيجاد بيئة جديدة يتم التعامل معها، كما يتم حفظ قيم محاورات المتغيرات الحالية في Stack وذالك عند الإستدعاء الذاتي.

الباب الرابع الهياكل الخاصة ذوات الأحجام الثابتة

الهدف من هذا الباب هو الأتي:

1. **القدرة على تكوين هياكل خاصة تستخدم كهياكل تحتوي على مجموعات مختلفة من البيانات مثل السجلات والقوائم المتصلة و الطوابير والمكادس والاشجار.**
2. **التعامل مع الهياكل الخاصة.**

4.0 السجلات Structures

عند دراستنا للجداول (arrays) ، ذكرنا أن محتويات المصفوفات من البيانات، يجب أن تكون **من نفس النوع**، ولكن في الواقع العملي، نجد أن البيانات الخاصة بكينونة ما تختلف من حيث النوع، عن كينونة أخرى، فمثلا سجل طالب يكون فيه رقم الطالب من نوع int ، جنس الطالب من نوع int ، المواد التي أخذها (عددهن n مادة) من نوع int ، المعدل التراكمي من نوع float ، اسم الطالب من نوع char وفي هذه الحالة تكون الجداول أو المصفوفات عديمة الفائدة، وعليه كان لا مفر من التفكير في البديل، ويمكن أن يكون البديل للجداول هو السجل Structure.

4.1 Definition of structur

السجل Structure عبارة عن مجموعة من المواقع المتجاورة في الذاكرة، يمكن أن تحتوي على أنواع (مختلفة / أو متشابهة) من البيانات تنتمي إلى كيان واحد، هذه البيانات تنتمي إلى كيان واحد، أو تشكل كياناً واحداً وتحت مسمى واحد.

ملحوظة عناصرStructure دائما ترتب في مواقع متجاورة في الذاكرة، ويكون عنوان أول موقع فيStructure هو العنوان لل Structure.

تعريف ال Structure بلغة C++

يتم تعريف ال Structure عن طريق ذكر كلمة struct ثم ذكر اسم السجل على هذا النحو:

```
struct student
{
  Data types;
 …
…..
  Data types;
    };
```

4.2 التعامل مع السجلات struct

للتعلمل مع السجلات struct يتم ذكر إسم السجل ثم يتم كتابة العلامة . ثم يتم كتابة الحقل المطلوب التعلمل معه، والمثال التالي يوضح إستخدام struct .

مثال$_1$: اكتب التعريف اللازم بلغة C++ وذالك لتوصيف سجل لطالب بياناته كالتالي:

اسم ، ورقم ، وجنس الطالب، المقررات الدراسية وتمثل بدرجات المواد (5 مقررات)، المعدل العام، حالة الطالب.

```
struct student
{
 int sno, state, subj[5],six;
 char sname [15];
 float avrage;
};
```

ففي هذ المثال يوجد سجل Structure إسمه student ويتكون من الأتي:

1- sno; اسم الطالب
2- state, حالة الطالب
3- subj[5], 5 مقررات دراسية وتمثل بدرجات المواد
4- six, جنس الطالب
5- sname [15], اسم الطالب
6- average, المعدل العام،

كما نشاهد في هذ المثال يوجد سجل Structure مكون من ثلاثة أنواع مختلفة من البيانات ، وكل هذه البيانات تشكل كياناً واحداً، وهذه أهم ميزة لل struc . والبرنامج التالي يوضح إستخدام struct .

```
#include<iostream.h>
#include<string.h>
main ( )
 {
    struct account
    {
     int no;
     char acc-name [15];
     float bal;
    };
    account a1, a2, a3;
    cout<<endl<<" Enter account nos., names, and balances" <<endl;
```

```
cin>> a1.no>> al.acc-name>>a.bal; // 20  SALEH 10  250.55
cin>> a2. no>>a2. acc-name>> a2.bal;// 50  mohammed  200.44
cin>> a3. no>> a3. acc-name>>a3.bal;// 1  allah   1
cout<<endl  >>a1. no>>a1.acc-name>> a1.bal;
cout<<endl  >>a2. no>> a2. acc-name>> a2.bal;
cout<<endl >>a3. no >>a3. acc-name>> a3.bal;
return 0;
}
```

عند فحص هذا البرنامج نجد الأتي:

1- التعريف في بداية البرنامج يضم ثلاث فقرات من البيانات مختلفة النوع ويضمهم في كيان واحد اسمه account، وهم على النحو:

- int sno, state, subj[5],six;
- char sname [15];
- float avrage;

2- a3,a2,a1 عبارة عن أسماء لمتغيرات من نوع structure.

3- عناصر structure تعالج بواسطة (.) يسبقها اسم المتغير ويلحقها اسم الفقرة مثل: a1.no = 20;

4- الفقرات التي تكون بذاتها عبارة عن array فإنها تحمل بنفسها عنوان القاعدة (أول خانة في array)، وبالتالي عند قرأتها لا داعي لكتابة ما يدل على العنوان مثل &، بل يكتفي بذكر إسم المتغير لها.

5- عناصرStructure دائما ترتب في مواقع متجاورة في الذاكرة على النحو الموضح بشكل 0 .4 الذي يمثل عناصر السجل account .

sno	state	subj[5]	six	sname[15]	avrage
الرقم	الحالة	الموضوع	الجنس	الاسم	المعدل

شكل 0 .4 يمثل عناصر السجل account .

مثال2 اكتب بلغة C++ التعريف اللازم لتوصيف السجل التالي:

Name	No	Six	Salary	Taxes	Net-salary
الاسم	الرقم	الجنس	المرتب	الضريبة	صافي المرتب

الحل

```
struct account
{
 int No, Six,;
 char name [15];
 float Salary,Taxes, Net-salary;
};
```

representation of strucur in memory 4. 1. 2

يمكن تمثيل عناصر السجلات في البرنامج السابق . a1,a2, a3 من نوع stract account في الذاكرة عند ادخال نفس البيانات المرافقة على النحو الأتي:

أولا السجل a1

يمكن تمثيل عناصر السجلa1 كما في الشكل1 .4 :

item_name	a1.no	a1.acc-name	a1. bal
item_value	20	SALEH10	250.55
item_address	4001	4003 - 4018	4022

. الشكل4.1 يمثل عناصر السجلa1

في الشكل 4.1 يكون عنوان .a1 هو مثلا 4001 وهو عنوان أول عنصر في السجل.

ثانيا السجل a2

يمكن تمثيل عناصر السجلa2 كما في الشكل 2 .4 :

تمثيل عناصر السجل.a2 من نوع stract account في الذاكرة

item_name	a2.no	a2.acc-name	a2. bal
item_value	50	mohammed	200.44
item_address	4023	4025 - 4040	4044

. الشكل2 .4 يمثل عناصر السجلa2

في الشكل السابق يكون عنوان .a2 هو مثلا 4023 وهو عنوان أول عنصر في السجل.

ثالثا السجل a3

يمكن تمثيل عناصر السجلa3 كما في الشكل 3 .4 :

item_name	a3.no	a3.acc-name	a3. bal
item_value	1	allah	1
item_address	4045	4047 - 4058	4062

. الشكل 4.3 يمثل عناصر السجلa3

في الشكل السابق يكون عنوان .a3 هو مثلا 4045 وهو عنوان أول عنصر في السجل.

An array of structures 4. 1. 3

في المثال السابق كأن لدينا ثلاث كينونات a1, a2 ,a3 وكان كل متغير يمثل سجل نوع stract account. ولكن إذا كأن لدينا 100 سجل أو أكثر من نوع stract account، فهل يعقل أن نذكر 100 متغيراً أو أكثر كتابياً؟ أو بأسلوب أفضل يمكن تمثيل array of structures و يكون ترتيب array of structures بالذاكرة على الشكل 4.4 :

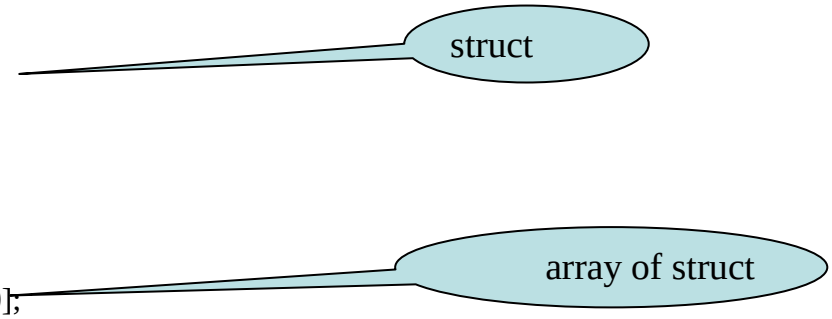
Indacc[0]	Indacc[0].no	Indacc[0].name	Indacc[0].bal
Indacc[1]	Indacc[1].no	Indacc[1].name	Indacc[1].bal
Indacc[2]	Indacc[2].no	Indacc[2].name	Indacc[2].bal
.	.	.	.
.	.	.	.
Indacc[99]	Indacc[99].no	Indacc[99].name	Indacc[99].bal

الشكل 4.4 يمثل array of structures .

إن الحل يكمن في التعامل مع array of structures ، حيث يتم ربط مجموعة من السجلات معا في حزمة، والبرنامج التالي يوضح ذالك.

والآن دعنا نكتب برنامجا نتعلم من خلاله array of structure

```
main ( )
{
    struct account
    {
    int no;
    char name[2];
    float bal;
    };
     account acc[100];
    int i;
    cout<<endl<<" Enter account no., names, and balance:";
    for (i=0; i<100; i++)
    {
    cin>>  acc [i].no >> acc[i].name>> acc[i] .bal;
    cout << acc [i].no << acc[i].name<< acc[i] .bal<<endl;
    }
  }
```



سوف يكون ترتيب array of structures بالذاكرة على الشكل 4 .5 :

acc[0]	acc[0].no	acc[0].name	acc[0].bal
acc[1]	acc[1].no	acc[1].name	acc[1].bal
acc[2]	acc[2].no	acc[2].name	acc[2].bal
.	.	.	.
.	.	.	.
acc[99]	acc[99].no	acc[99].name	acc[99].bal

. الشكل 4.5 يمثل محتويات array of structures المذكورة في البرنامج السابق

مثال$_1$: اكتب برنامجا يقوم بالتالي:

1. يعرف سجل بلغة C++ وذالك لتوصيف طالب بياناته تتكون من

اسم ، ورقم ، وجنس الطالب، المقررات الدراسية وتمثل بدرجات المواد (5 مقررات)، المعدل العام، حالة الطالب.

2. يكون 100 من سجلات الطلاب على هيئة هيكل
3. يولد ويحسب البيانات اللازمة لسجلات الطلاب
4. يطبع سجلات الطلاب كل سجل في سطر.

الحل

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
 struct student
 {
  int no ,subj[5], sex,state;
  char name[10];
  float av;
  };
 student list[100];
 int main ( )
 {
  int i,j,sum=0;
  for(i=0;i<100;i++)
  {
      list[i].no=i+1;
     list[i].state=1;
   for(j=0;j<10;j++)
     list[i].name[j]=rand()%26+65;
   list[i].sex=rand( )%2;
```

```
  for(j=0;j<5;j++)
 {
    list[i].subj[j]= rand( )%60+40;
     if(list[i].subj[j]<50)
       list[i].state=0;
      sum+= list[i].subj[j];
  }
list[i].av=sum/5.0;
sum=0;
cout<<setw(4)<< list[i].no<<" ";
   for( j=0;j<10;j++)
    cout<< list[i].name[j];
    cout<<" "<< list[i].sex<<" ";
  for( j=0;j<5;j++)
     cout<<setw(5)<< list[i].subj[j];
  cout<<setw(9)<< list[i].av<<" "<< list[i].state<<endl;
}
   return 0;
  }
```

و يمكن كتابة نفس البرنامج بإ ستخدام الدوال على النحوالتالي:

الحل

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
 struct student
 {
  int no ,subj[5], sex,state;
  char name[10];
  float av;
  }  list[100];

void Pt(student[ ], int );
void Gt(student[ ] , int );
int main (  )
 {
  int i;
  for(i=0;i<100;i++)
  {
    Gt(list,i);
    Pt(list,i);
  }
return 0;
}
```

```
//-------------------------------------------
void Gt(student list[ ], int n )
{
   int j,sum=0;

   list[n].no=n+1;
    list[n].state=1;
  for(j=0;j<10;j++)
    list[n].name[j]=rand()%26+65;
  list[n].sex=rand( )%2;
  for(j=0;j<5;j++)
  {
    list[n].subj[j]= rand( )%60+40;
     if(list[n].subj[j]<50)
       list[n].state=0;
      sum+= list[n].subj[j];
  }
list[n].av=sum/5.0;
sum=0;
return ;
}
//-------------------------------------------
void Pt(student list[ ], int n )
{
    int j;
cout<<setw(5)<< list[n].no<<" ";
   for( j=0;j<10;j++)
    cout<< list[n].name[j];
    cout<<"   "<< list[n].sex<<"   ";
  for( j=0;j<5;j++)
     cout<<setw(5)<< list[n].subj[j];
  cout<<setw(9)<< list[n].av<<" "<< list[n].state<<endl;
   return;
  }
```

4.1.4 السجلات و المؤشرات Structures & Pointers

ذكرنا سابقاً أن المؤشر عبارة عن موقع يمكن أن يحمل عنوان موقع آخر، فيمكن أن يكون عنوان الموقع الآخر هو عنوان Structure . وطالما أن المؤشر عنوان موقع فيمكن أن يوضع هذ العنوان في مؤشر .

والبرنامج التالي يوضح إستخدام المؤشر **Pointers**

```
# include<iostream.h>
main ( )
{
 struct book
 {
   char name [25], author [25];
   int cal;
 };
book b1 = {“let us c++”, “Pc”, 262},*ptr;
 ptr= &b1;
cout   << b1. name<< b1. author<<b1. cal<<endl;
cout   <<ptr-> name <<ptr->author<< ptr-> cal<<endl;
}
```

struct

حجز سجل في الذاكرة من نوع book

التعامل مع الحقل داخل السجل

operator => يجب أن يكون على يساره متغير من نوع مؤشر

عند تنفيذ هذا البرنامج نجد الأتي:

```
let us c++ Pc 262
let us c++ Pc 262
```

و يمكن تمثيل محتويات structures بالذاكرة على الشكل 6 .4 :

b1.name	b1.author	b1.cal
let us C++	PC	262
4001	4026	4051

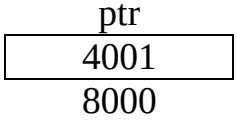
. الشكل 4.6 تمثيل محتويات structures بالذاكرة في البرنامج السابق

وعند فحص البرنامج السابق نجد الأتي:

في الأمر ptr= &b1; نجد أن ptr متغير من نوع مؤشر، وهو يحمل عنوان struct الذي هو من نوع book والذى يساوى فرضا 4001 .

book *ptr; ptr= &b1;

فإذا وجد هذا الأمر ptr++ فماهي قيمة ptr ؟



والبرنامج التالي يوضح إستخدام المؤشرات مع السجلات ايضا **Pointers and struct**

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
 const maxs=6,l=5 , maxn=100;
 struct student
 {
  int no;
  char name[l];
  float av, subj[maxs];
  short s,sex;
  };
 student list[maxn];
 void pt( student *p);
 void ipt(student *p);
 void state( student *p);
 int main ( )
 {
  int i;
  student *p;    p=list;
  for(i=0;i<maxn;i++)
  {
      p->no=i+1;
      ipt(p);
      state(p);
     pt(p++);
   }      cin>>i;
   return 0;
  }
//------------------------------
  void pt( student *p)
  {
    int i;
   cout<<setw(4)<<p->no<<" ";
```

```
    for( i=0;i<l;i++)
     cout<< p->name[i];
     cout<<" "<<p->sex<<" ";
   for( i=0;i<maxs;i++)
      cout<<setw(5)<<p->subj[i];
   cout<<setw(9)<<p->av<<" "<<p->s<<endl;
   return ;
   }
//-------------------------------
void ipt(student *p)
   {
   float sum=0;
   int i,j;
   for(j=0;j<l;j++)
     p->name[j]=rand()%27+64;
     p->sex=rand( )%2;
   for( i=0;i<maxs;i++)
     {
       p->subj[i]=float (rand( )%60)+40;
       sum+= p->subj[i];   }
     p->av=sum/maxs;
    return ;
   }
//-------------------------------
  void state( student *p)
   {
   int i=0;
   while(i<maxs&&p->subj[i]>49)
         i++;
    if(i<maxs)   p->s=0;
     else
       p->s=1;
     return ;
     }
```

وهنا برنامج آخر يوضح إستخدام المؤشر مع السجلات ايضا **Pointers and structs**

```
#include<iomanip.h>
#include<stdlib.h>
#include<fstream.h>
 const maxs=6,l=5 , maxn=100;
 fstream Dataf1;
 struct student
 {
  int no;
  char name[l];
  float av;
  float subj[maxs];
  short s,sex;
  };
 student list[maxn];
 void pt( student *p);
 void ipt(student *p);
 void state( student *p);
int main ( )
 {
  int i;
  student *p;
  p=list;
  Dataf1.open("da2.cpp",ios::out);
  for(i=0;i<maxn;i++)
  {
  p->no=i+1;
  ipt(p);
  state(p);
  pt(p++);
  }
   Dataf1.close();
   cin>>i;
   return 0;
  }
//-------------------------------
  void pt( student *p)
  {
  int i;
   cout<<setw(4)<<p->no<<" ";
     Dataf1<<setw(4)<<p->no<<" ";
   for( i=0;i<l;i++)
    {
    cout<< p->name[i];
     Dataf1<< p->name[i];
    cout<<" "<<p->sex<<" ";
     Dataf1<<" "<<p->sex<<" ";
     }
```

```
  for( i=0;i<maxs;i++)
   {
  cout<<setw(5)<<p->subj[i];
   Dataf1<<setw(5)<<p->subj[i];
   }
  cout<<setw(9)<<p->av<<" "<<p->s;
   Dataf1<<setw(9)<<p->av<<" "<<p->s;
  cout<<endl;
   Dataf1<<endl;
  return ;
  }
//-------------------------------
  void ipt(student *p)
  {
  float sum=0;
  int i,j;
  for(j=0;j<l;j++)
    p->name[j]=rand()%27+64;
    p->sex=rand()%2;
  for( i=0;i<maxs;i++)
  {
   p->subj[i]=float (rand()%60)+40;
   sum+= p->subj[i];
   }
   p->av=sum/maxs;
  return ;
  }
//-------------------------------
  void state( student *p)
  {
  int i=0;
  while(i<maxs&&p->subj[i]>49)
   i++;
   if(i<maxs)
    p->s=0;
    else    p->s=1;
    return ;
    }
  //-------------------------------
```

The output of executing this program as follows:

```
 1 V 1 V 1 T 1 J 1 S 1       55   95   68   86   44   78       71 0
 2 G 1 J 1 X 1 J 1 L 1       63   67   99   41   90   45     67.5 0
 3 B 0 W 0 L 0 C 0 J 0       61   71   95   53   54   90  70.6667 1
 4 J 0 Q 0 I 0 @ 0 U 0       66   64   52   43   66   64  59.1667 0
 5 H 0 J 0 O 0 S 0 J 0       88   46   87   62   74   89  74.3333 0
 6 @ 0 D 0 @ 0 F 0 Y 0       45   79   90   53   93   89  74.8333 0
 7 Q 0 S 0 Q 0 W 0 Z 0       80   78   76   40   42   82  66.3333 0
 8 U 1 D 1 W 1 C 1 V 1       48   59   41   57   75   41     53.5 0
 9 Z 1 O 1 P 1 X 1 T 1       43   47   72   71   93   53  63.1667 0
10 E 0 F 0 K 0 S 0 I 0       64   74   71   99   66   50  70.6667 1
11 W 0 J 0 G 0 O 0 A 0       64   60   88   64   59   48  63.8333 0
12 W 0 N 0 P 0 P 0 Q 0       98   56   67   64   48   80  68.8333 0
```

13 H 0 G 0 @ 0 W 0 O 0 85 91 99 42 84 51 75.3333 0
14 U 1 I 1 Y 1 @ 1 B 1 42 72 73 58 52 41 56.3333 0
15 C 0 Y 0 M 0 X 0 F 0 98 65 69 43 52 76 67.1667 0
16 P 0 R 0 R 0 J 0 F 0 69 53 79 47 90 46 64 0
17 M 0 T 0 F 0 R 0 U 0 95 47 94 85 78 63 77 0
18 F 1 F 1 T 1 G 1 B 1 99 51 63 78 54 64 68.1667 1
19 J 0 Z 0 E 0 S 0 G 0 46 62 45 53 65 48 53.1667 0
20 P 0 W 0 Y 0 A 0 W 0 46 96 54 41 89 83 68.1667 0
21 R 0 Z 0 M 0 V 0 @ 0 95 52 97 91 86 44 77.5 0
22 R 1 J 1 N 1 U 1 Q 1 56 71 82 89 75 78 75.1667 1
23 L 0 N 0 C 0 F 0 Q 0 66 74 55 59 76 96 71 1
24 K 1 W 1 N 1 P 1 Z 1 41 40 85 89 70 41 61 0
25 Q 1 G 1 S 1 T 1 F 1 40 84 90 78 87 41 70 0
26 F 1 H 1 B 1 I 1 Q 1 71 45 61 82 86 94 73.1667 0
27 D 1 A 1 T 1 D 1 Z 1 42 72 65 48 80 73 63.3333 0
28 I 1 U 1 E 1 A 1 E 1 76 63 54 76 99 89 76.1667 1
29 L 0 V 0 P 0 E 0 W 0 61 94 64 43 54 93 68.1667 0
30 L 0 A 0 Q 0 Q 0 A 0 64 91 99 72 55 67 74.6667 1
31 I 1 F 1 I 1 H 1 R 1 56 89 44 42 65 40 56 0
32 G 0 Z 0 V 0 U 0 P 0 88 50 65 45 93 87 71.3333 0
33 @ 0 Z 0 P 0 U 0 Z 0 49 45 55 48 42 71 51.6667 0
34 R 0 M 0 U 0 U 0 N 0 44 54 98 72 95 64 71.1667 0
35 O 1 A 1 Y 1 P 1 L 1 84 99 73 96 92 88 88.6667 1
36 M 1 W 1 T 1 L 1 L 1 90 58 67 95 72 45 71.1667 0
37 F 1 V 1 Q 1 X 1 Q 1 59 85 86 84 42 41 66.1667 0
38 D 0 I 0 @ 0 @ 0 Y 0 65 81 86 93 75 41 73.5 0
39 K 0 I 0 G 0 X 0 E 0 84 55 89 51 63 43 64.1667 0
40 O 1 F 1 J 1 W 1 P 1 51 71 45 96 82 47 65.3333 0
41 V 0 X 0 G 0 A 0 X 0 50 87 50 89 57 80 68.8333 1
42 O 0 E 0 C 0 V 0 W 0 83 67 62 98 92 71 78.8333 1
43 K 0 P 0 M 0 I 0 C 0 53 90 77 55 74 80 71.5 1
44 V 1 J 1 K 1 Q 1 V 1 47 84 41 45 80 83 63.3333 0
45 E 1 W 1 U 1 F 1 Q 1 57 48 61 41 58 97 60.3333 0
46 Z 1 V 1 E 1 @ 1 D 1 91 53 53 71 75 76 69.8333 1
47 D 1 M 1 C 1 U 1 K 1 67 64 66 54 79 59 64.8333 1
48 Y 1 J 1 B 1 U 1 J 1 77 64 58 46 86 62 65.5 0
49 L 0 C 0 Y 0 Q 0 A 0 60 66 63 71 43 53 59.3333 0
50 T 0 H 0 J 0 V 0 N 0 87 42 88 69 41 95 70.3333 0
51 F 1 E 1 T 1 C 1 T 1 88 59 97 92 53 70 76.5 1
52 X 0 N 0 D 0 F 0 D 0 52 66 79 90 93 96 79.3333 1
53 H 0 U 0 M 0 S 0 D 0 60 60 58 62 81 64 64.1667 1
54 F 1 N 1 U 1 R 1 L 1 99 42 73 62 51 87 69 0
55 I 0 N 0 W 0 S 0 I 0 68 79 62 59 67 77 68.6667 1
56 N 0 F 0 V 0 Y 0 V 0 62 89 48 92 44 51 64.3333 0
57 S 0 Q 0 Y 0 W 0 F 0 46 73 54 45 92 41 58.5 0
58 C 1 L 1 A 1 Z 1 W 1 56 95 50 98 47 99 74.1667 0
59 Z 1 J 1 A 1 G 1 U 1 57 66 72 79 50 57 63.5 1
60 Q 0 T 0 I 0 D 0 T 0 60 79 47 65 53 64 61.3333 0
61 I 0 Q 0 X 0 K 0 N 0 67 97 81 42 83 80 75 0
62 @ 1 S 1 O 1 E 1 N 1 48 48 65 98 88 86 72.1667 0
63 W 1 M 1 Y 1 N 1 P 1 52 77 43 42 62 92 61.3333 0
64 H 0 J 0 L 0 E 0 D 0 66 57 99 52 68 99 73.5 1
65 P 0 G 0 Z 0 W 0 J 0 64 89 63 51 97 93 76.1667 1
66 V 1 R 1 U 1 P 1 D 1 43 66 58 63 53 40 53.8333 0
67 N 1 D 1 Y 1 O 1 B 1 57 97 44 81 64 74 69.5 0
68 Z 1 I 1 G 1 W 1 I 1 95 52 91 92 76 46 75.3333 0
69 Y 1 C 1 C 1 G 1 O 1 62 50 45 71 92 86 67.6667 0
70 J 1 Q 1 V 1 G 1 D 1 54 68 72 63 65 70 65.3333 1
71 N 0 V 0 Q 0 Q 0 F 0 97 86 87 71 73 52 77.6667 1
72 F 1 U 1 @ 1 I 1 H 1 70 41 55 48 70 86 61.6667 0

73 C 1 M 1 S 1 F 1 E 1 71 45 77 97 61 92 73.8333 0
74 X 1 C 1 E 1 Y 1 V 1 87 82 89 69 91 57 79.1667 1
75 J 0 O 0 B 0 Q 0 F 0 63 93 55 72 87 64 72.3333 1
76 Z 1 A 1 X 1 P 1 X 1 77 65 44 50 80 87 67.1667 0
77 X 1 W 1 Q 1 O 1 S 1 86 87 44 60 61 44 63.6667 0
78 O 1 Z 1 G 1 C 1 L 1 94 54 61 73 91 47 70 0
79 M 0 C 0 @ 0 N 0 T 0 82 40 59 71 54 95 66.8333 0
80 R 1 E 1 H 1 H 1 Q 1 61 76 51 94 59 47 64.6667 0
81 Z 0 B 0 E 0 D 0 U 0 78 98 45 67 88 91 77.8333 0
82 @ 1 R 1 Q 1 A 1 F 1 92 55 87 57 70 51 68.6667 1
83 L 0 X 0 Q 0 I 0 P 0 99 88 62 73 99 64 80.8333 1
84 Z 0 Q 0 T 0 K 0 Z 0 63 57 98 92 43 86 73.1667 0
85 T 1 B 1 R 1 N 1 V 1 94 45 44 89 41 82 65.8333 0
86 A 1 W 1 D 1 W 1 I 1 46 49 62 56 69 84 61 0
87 P 0 H 0 O 0 T 0 F 0 96 78 83 72 40 87 76 0
88 B 1 E 1 R 1 V 1 M 1 60 59 72 96 70 98 75.8333 1
89 Y 1 A 1 O 1 G 1 T 1 89 71 76 81 51 46 69 0
90 O 1 S 1 Q 1 E 1 N 1 70 42 87 86 69 62 69.3333 0
91 Q 1 H 1 M 1 Q 1 L 1 89 54 85 92 73 98 81.8333 1
92 L 1 @ 1 T 1 P 1 O 1 88 61 60 44 45 40 56.3333 0
93 I 1 L 1 A 1 A 1 T 1 46 96 54 94 40 51 63.5 0
94 B 1 L 1 A 1 Q 1 I 1 93 86 61 92 43 90 77.5 0
95 R 0 M 0 @ 0 T 0 F 0 93 42 75 43 75 72 66.6667 0
96 H 1 Q 1 F 1 F 1 @ 1 94 90 66 60 83 92 80.8333 1
97 O 1 Y 1 O 1 K 1 R 1 52 65 75 83 77 72 70.6667 1
98 @ 1 P 1 M 1 A 1 H 1 42 43 71 95 88 97 72.6667 0
99 T 0 W 0 J 0 R 0 E 0 62 87 71 58 80 85 73.8333 1
100 Q 1 O 1 T 1 I 1 S 1 61 80 92 51 92 84 76.6667 1

4.2 المكدس The Stack

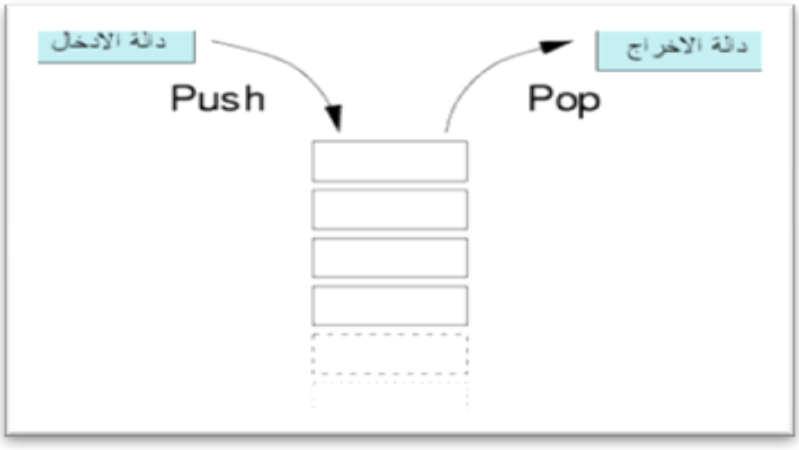
المكدس Stack عبارة عن abstract data type ، وهوغير معرف في لغة C++ أو لغة Pascal، وإنما يتم تعريفه من قبل المبرمجين، ويعتبر Stack من التطبيقات المهمة في الحياة، ويمكن أن يعرف علميا على النحوالتالي:

A stack is an ordered collection of items into which new items may be inserted and from which items may be deleted at one end, called the top of the stack.

كما يمكن أن يتم تشبيه ال stack بأنبوبة طويلة مفتوحة من جهة ومغلقة من الجهة الأخرى، كما في شكل 4.7 الذي يمثل **المكدس** stack .

4.2.1 عمليات إدخال المعلومات أو إخراج المعلومات الى و من المكدس

عملية إدخال المعلومات أو إخراج المعلومات تتم فقط من الجهة المفتوحة للstack ، وبناءً على طبيعة الأنبوبة، فإن أول داخل يكون آخر خارج. First in last out

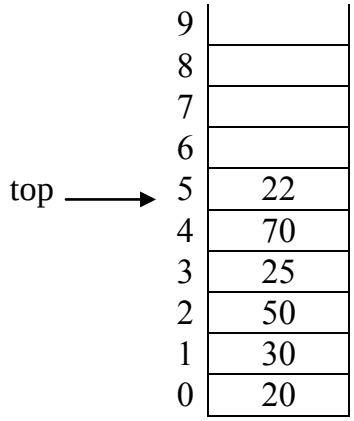


شكل 4.7 تمثيل **سالمكد** stack

ويسمى أحياناً FILO اختصاراً للكلمات السابقة First in last out

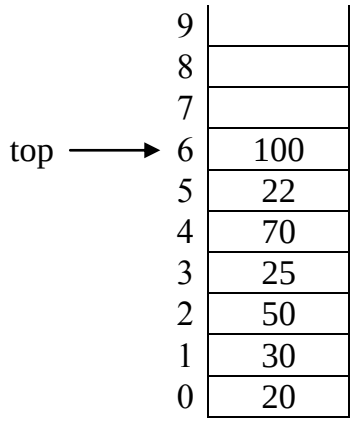
وكما ذكرنا سابقا أن **المكدس** stack عبارة عن هيكل يعرف من قبل المستخدم /المبرمج أي أنه لا يوجد جاهزا في كثير من اللغات، وبالتالي على المبرمج بناؤه.

في الشكل 4.8 يوجد مكدس stack اسمه S يحتوي على 6 عناصر، سعته الكلية 10عناصر، المؤشر top يشير إلى العنصر السادس في stack، وقد تم إدخال العناصر إلى ال stack بهذا الترتيب 20 ,30 ,50 ,25 ,70 ,22.



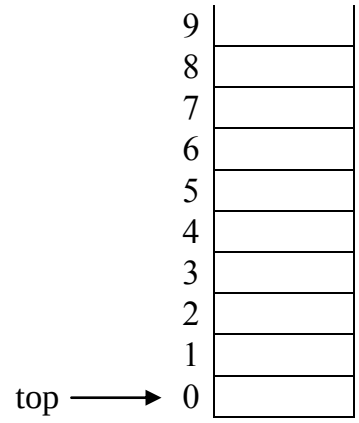
شكل 4.8 يمثل **مكدس** يحتوي على 6 عناصر

فإذا أردنا اضافة العنصر 100 إلى **المكدس**، فيكون شكل **المكدس** على هذا النحو شكل 4.9 :



شكل 4.9 يمثل **مكدس** يحتوي على 9 عناصر

وعند إخراج محتويات **المكدس** فسيكون ترتيبهن على هذا النحو: 100 22, 70, 25, 50, 30,20 أي على عكس ترتيب إدخالهن، ويكون شكل **المكدس** على هذا النحو شكل 4.10 .



شكل 4.10 يمثل **مكدس** فارغ

4.2.2 إستخدامات ال Stack

يستخدم ال Stack في كثير من التطبيقات، وخاصة في أنظمة التشغيل لإدارة المهام، و من أهم إستخدام ال Stack تحديد أولويات العمليات (حسابية أو غير ذالك)، والكشف عن صحة التعابير الجبرية وخاصة في المترجمات، و الأمثلة التالية توضح ذالك.

- Y= ((a+b)*c+d-e)/(f+g)-(h+j)*(k-l))/(m-n)
- X= (a+b))*((c+d)

في المثال الأول نجد ان عدد الاقواس المفتوحة لا يساوي عدد الاقواس المغلقة، وبالتالي هناك خطا من حيث عدد الاقواس، أما في المثال الثاني فنجد ان عدد الاقواس المفتوحة يساوي عدد الاقواس المغلقة، ولكن هناك خطا من حيث مواقع الاقواس.

4.2.3 العمليات المشهورة على Stack

توجد عمليتين رئيسيتين على Stack أحدهما Push، يتم من خلالها إدخال عنصر إلى Stack.

والعملية الثانية اسمها Pop، يتم من خلالها إخراج عنصر من داخل Stack.

إذا فالتعامل مع Stack يتم عن طريق الدالتين Push، Pop فقط ولا يمكن إدخال أو إخراج عنصر بدونها على التوالي.

كذلك توجد عمليتين فرعيتين على Stack هي Full, empty.

الأولى للكشف عن stack هل فيه معلومات أم هو فارغ.

والثانية للكشف عن stack هل هو مملوء أم لا، وسوف نتعرف على الكل بإذنه تعالى إنه على ما يشاء قدير.

4.2.4 بناء هيكل على هيئة Stack ثابت الحجم

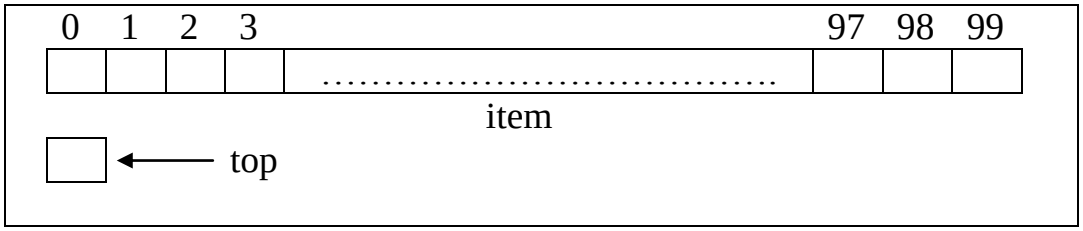
ممكن أن يكون بناء الهيكل على حجم ثابت، ولكن العيب الأساسي هو عدم القدرة على تجاوز هذا الحجم الثابت، فمثلا لو تم حجز مجموعة معينة من المواقع في الذاكرة بحيث تتم عملية إدخال المعلومات أو إخراج المعلومات فقط من جهة واحدة (الجهة المفتوحة للstack) :مثل

```
# define size 100
struct Stack
{
int item [size];
int top;
} s;
```

مصفوفة سوف تستخدم على هيئة هيكل Stack

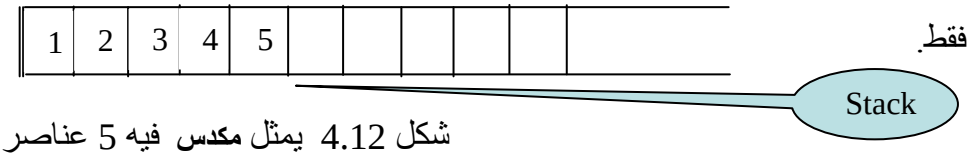
مؤشر ال Stack

ويمكن تمثيل المكدس السابق بشكل 4.11



شكل11 .4 هيكل على هيئة Stack يستوعب 100 من العناصر

الشكل أعلى يمكن التعامل معه على هيئة Stack من حيث الإلتزام بعمليتي إدخال و إخراج المعلومات من جهة واحدة فقط.



شكل 4.12 يمثل **مكدس** فيه 5 عناصر

وهنا يجب أن لا يزيد حجم المكدس عن قيمة size التي تساوي 100

4.2.5 كيفية إدخال عنصر إلى Stack

ذكرنا سابقا أن Stack ممكن تخيله على هيئة أنبوبة طويلة ولذلك لا بد من عملية ما، تتواكب مع هذا التعريف و لكي نضع فيه عنصراً ما، نحتاج إلى دالة من خلالها يتم وضع العنصر داخل Stack. أي أن العملية push هي الجديرة بوضع العنصر في الموقع الملائم داخل stack.

من شكل الهيكل السابق يعرف أن محتوياته هما:

1) مصفوفة أو array حجمه = Size n ، ونوعه int هو الذي سوف يتم حفظ البيانات فيه.

2) أما العنصر الثاني فهو مؤشر نوعه int سوف يستخدم لكي يشير إلى أعلى عنصر في stack، وهذا يسمى top وتكون قيمته -1 عندما يكون ال stack فارغاً، أو size -1 عندما يكون stack مملوء.

4.2.5.1 الدالة Push، عملية إدخال عنصر ما داخل Stack

الدالة Push هي الأداة التي من خلالها يتم وضع عنصر ما داخل Stack، فإذا أردنا أن ندخل عنصراً إلى Stack فما علينا أولاً إلا أن نتأكد من وجود فراغ للعنصر المطلوب أم لا؟

فإذا وجد فراغ، فيتم زيادة قيمة المؤشر (top) بمقدار واحد، ثم يتم وضع العنصر item المراد حفظه في المصفوفة عند المؤشر top.

أما إذا لم يوجد فراغ فيتم إظاهر رسالة توضح ذلك، مثل .Stuck is full or stack overflow

```
void Push ( Stack * ps, int x)
{
 if (ps => top = =size-1)
   {
  cout stack overflow;  exit (1);
   }
else
ps=>item [++(ps=>top)] = x;
return;
}
```

مؤشر الى ال Stack

العنصر المراد ادخاله الى Stack

التأكد من وجود فراغ للعنصر في Stack

وضع العنصر في Stack

1	2	3	4	5						

عملية إخراج عنصر من ال Stack

4.2.5.2 الدالة Pop

طالما وأن Stack على هيئة أنبوبة فعلينا أن نبحث عن وسيلة ما لإخراج عنصر ما من Stuck، هذه الوسيلة تتمثل في شفط العنصر إلى أعلى من الأنبوبة إن وجد هناك عنصراً.

إذاً علينا في البداية أن نفحص هل ال stack فارغاً أم لا ؟ فإذا كان ال stack فارغاً، يتم توجيه رسالة توحي بذلك مثل Empty stack . وإذا لم يكن فارغاً، فيتم سحب أول عنصر يواجه في stack، ثم يتم تنقيص المؤشر top بمقدار واحد.

```
int pop ( Stack  *Ps)
{
if (Ps→ top= = -1)
{
cout<<"Empty stack";  exit (1); }
else
return (Ps → item [ps →top --]);
}
```

مؤشر Stack

التأكد من وجود عنصر في Stack

سحب العنصر من Stack

Empty and full functions

وقبل أن نأخذ أمثلة على الدالتين push, pop، دعنا نكتب الدالتين الفرعيتين empty and full .

4.2.5.3 الدالة empty

```
int empty (struct stack *ps)
    {
    if (Ps → top = = -1)
     return (-1);
    return (1);
    }
```

4.2.5.4 الدالة full

```
    int full (struct stack *ps)
    {
    if (Ps → top = = size-1)
    return (-1);
   return (1);
    }
```

4.2.5.5 أمثلة على إستخدام Stack

مثال (1): اكتب برنامجا يولد عشرة أعداد، ثم يطبعهم بحيث يكون آخر عدد تم توليده هو أول عدد يطبع.

الحل

```
#include<iostream.h>
#include<stdlib.h>
 # define size 100
struct stack
{
int item [size];
int top;
} s ;
int pop (stack *ps);
void push ( stack *ps, int x);
main (  )
{
 struct stack *ps, s;
 int value,i;
 ps = &s; s.top = - 1;
 for (i= 0; i < 10; i ++)
  {
         value =rand()%1000;
         push (ps, value);
         cout<<  value<< " ";
         }
  ps = &s;
  cout<<endl;
  for (i = 0; i < 10; i ++)
         cout <<endl<< pop (ps);
return 0;
}
//-------------------------
int pop ( stack *ps)
 {
   int i;
  if (ps->top == -1)
         {
           cout<<"empty stack" <<endl;
                 return -1;
          }
         i=ps->top;
          ps->top--;
```

```
 return (ps->item[i]);
}
//------------------------
void push (struct stack *ps, int x)
          {
           int i;
           if (ps->top == size - 1 )
             {
              cout<<" stock over flow" <<endl;
              return ;
             }
            ps->top++;
            i=ps->top;
            ps->item[i] = x;
          return;
         }
//-----------------------------------
```

مثال (2):

```
#include<iostream.h>
#include<conio.h>
# define size 50
struct stack
{
 int item [size], top;
 }s;
 void push (stack *ps, int);
  int pop (stack *);
         main (  )
         {
          int x,o;
          stack *Ps;
          Ps = &s; Ps->top = -1;
         do
         {
          cout<<"1-  push    2-pop    3 exit "<<endl;
          cout<<"Enter your choice "; cin>>o;
          switch (o)
           {
                 case 1 : cout<<"Enter the value to be pushed ";
                                 cin>>x;
                                 push (Ps, x);
                                 break;
                 case 2: cout<<"The value is ";
```

```
                    pop (Ps);
                    break;
            case 3: exit(0);
            default: cout<<"out of range ";
                     break;
             }
        } while (o !=3);

 return 0;
 }
//-------------------------
int pop (stack *Ps)
{
if (Ps ->top == -1)
  cout<<"empty stack" <<endl;
else
  return (Ps->item[Ps->top --]);
}
//-------------------------
void push (stack *Ps, int x)
{
if (Ps ->top == size - 1 )
  cout<<" stock over flow" <<endl;
else
  Ps->item[++Ps->top] = x;
return;
}
```

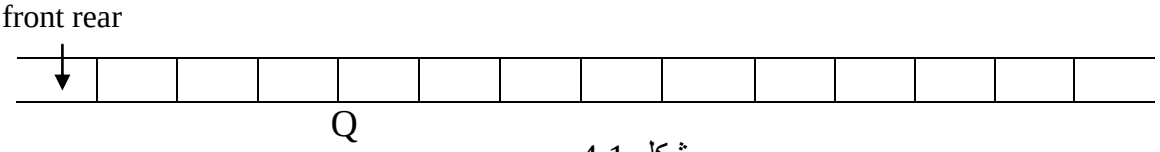
4.3 Queue الطابور

يمكن تعريف الطابور على النحو التالى:

A queue is an ordered collection of items from which items may by deleted at one end (called front) and into which items may by inserted at the other end (called rear)

ال queue عبارة عن abstract data type نوع من أنواع هياكل البيانات المغلفة، ممكن أن نتخيل أن Queue الطابور يمكن التعبير عنه بعبارة عن أنبوبة مفتوحة الطرفين، أي مجموعة محدودة من المواقع المتجاورة في الذاكرة نوعها حسب الطلب، و تتم عملية إدخال المعلومات إلى الطابور من الخلف، و تتم عملية إخراج المعلومات من الطابور من الأمام. أي أننا نحتاج إلى مؤشرين أحدهما يؤشر للأمام ، ومؤشر اخر يؤشر للخلف. ، كما في الشكل 4 .1 .

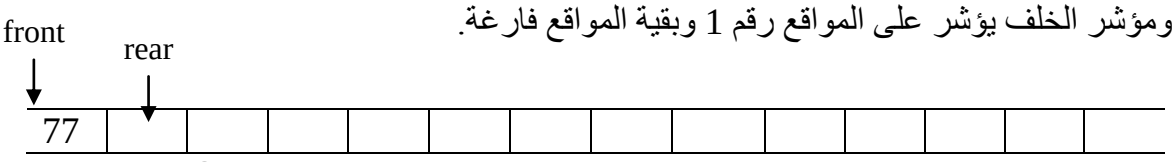
تسمي عملية الإدخال و الإخراج من queue أحياناً FIFO وهي إختصار للكلمات First in First out



شكل 4.1

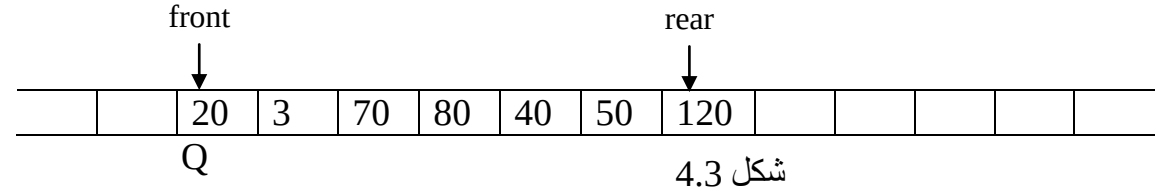
في الشكل 4.1 يوجد queue سعته 14 عنصراً لا يوجد فيه أي عنصر، مؤشر الأمام وكذالك مؤشر الخلف كلاهما يؤشرا على المواقع رقم 0 وكل المواقع فارغة.

أما في الشكل 4.2 فيوجد queue سعته 14 عنصراً ، يوجد فيه عنصرا واحدا، ومؤشر الأمام يؤشر على المواقع رقم 0 ومؤشر الخلف يؤشر على المواقع رقم 1 وبقية المواقع فارغة.



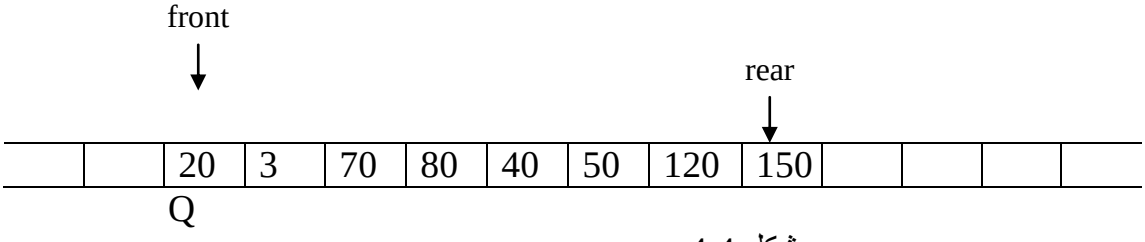
شكل 4.2

و في الشكل 4.3 يوجد queue سعته 14 عنصراً يوجد فيه 7 عناصر، مؤشر الأمام يؤشر على العنصر الثالث لأن الأول والثاني قد تم إخراجهما، ومؤشر الخلف يؤشر على العنصر رقم 8 وبقية المواقع فارغة.



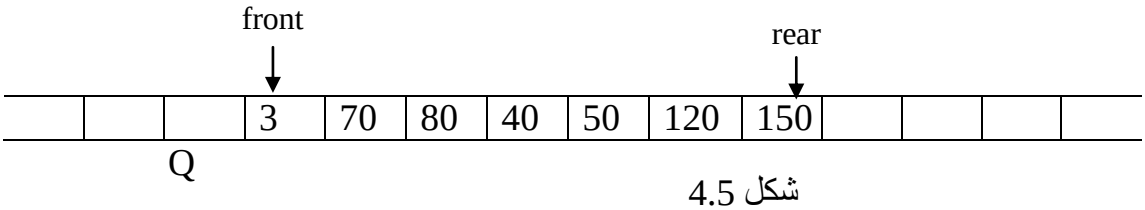
شكل 4.3

فإذا أردنا إضافة العنصر150 إلى الqueue فيكون الطابور على النحو التالي شكل 4.4



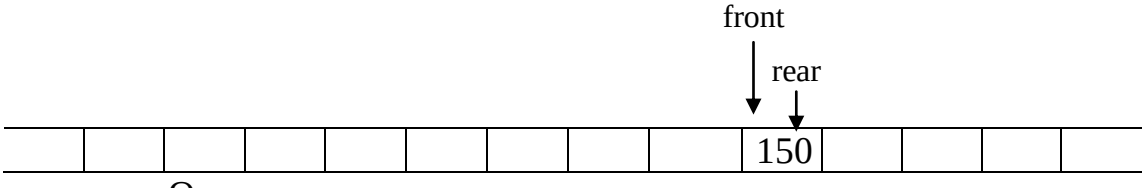
شكل 4.4

أما إذا أردنا حذف عنصر من الqueue فيكون الطابور على النحو التالي شكل 4.5



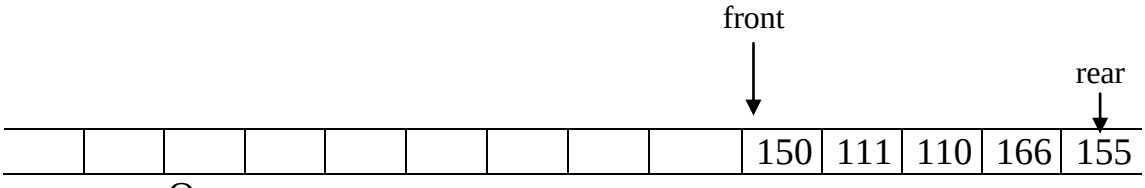
شكل 4.5

و إذا أردنا حذف 6 عناصر من الqueue فيكون الطابور على النحو التالي شكل 4.6



شكل 4.6

و إذا أردنا إضافة 4 عناصر إلى الqueue فيكون الطابور على النحو التالي شكل 4.7



شكل 4.7

وهنا نجد أن الطابور قد وصل إلى نهايته بالرغم من أن هناك متسع لتسعة عناصر، ولا يمكن استغلال الفراغ.

الذي يجب أن نؤكد عليه في queue ، هو أن عملية إدخال المعلومات إلى الطابور ال queue لا بد وأن تتم من الخلف، وعملية الحذف للمعلومات لا بد وأن تكون من الأمام، ولا يجوز غير ذالك في العمليات التي تسري على الطابور ال queue.

4.3.1 العمليات الروتينة على الطابور ال queue

توجد أربع عمليات يمكن تطبيقهن على الطابورqueue كماهو موضح بالجدول 4.1.

الجدول 4.1

عمل الدالة	اسم الدالة
إدخال عنصر إلى ال queue	Insert (q, x)
إزالة عنصر من ال queue	Remove (q)
استعلام هل ال queue فارغة	Empty (q)
استعلام هل ال queue مملوء	full (q)

4.3 .2 كيف نبني طابور queue في لغة C++

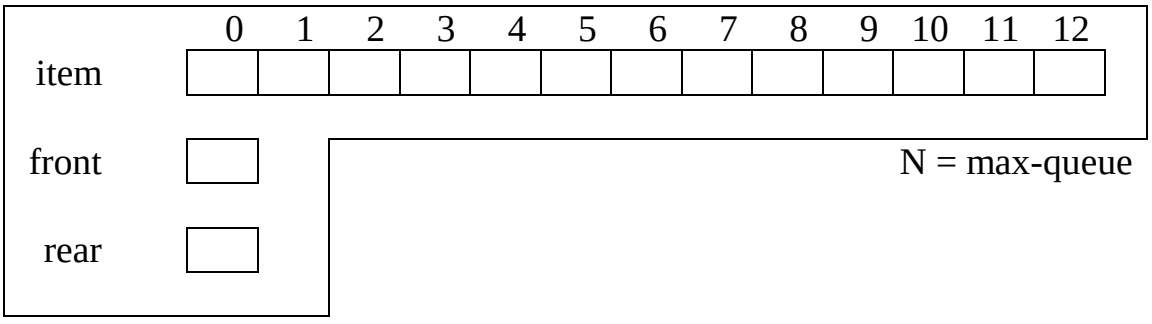
لكي نجيب على هذا السؤال، علينا أن نعرف مإذا نريد من التعريف السابق (queue). ذكرنا انه ممكن أن نتخيل أن الqueue عبارة عن أنبوبة مفتوحة الطرفين، يمكن تمثيل الأنبوب با array (مجموعة محدودة من المواقع في الذاكرة نوعها حسب الطلب)، وبهذا يكون الجزء الأول قد تم تمثيله، الجزء الثاني أن الإضافة من الخلف ويكون الحذف من الأمام.

إذا علينا تعريف مؤشرين أحدهما أمامي front والآخر خلفي rear نوع هذين المؤشرين لا بد وأن يكون int .

إذا ترجم هذا الكلام إلى لغة C++ فيكون على هذا النحو:

```
# define max_queue 12
stract queue
 {
    int item [max queue];
    int front, rear;
  }q;
```

من الممكن أن يكون نوع item أي نوع آخر غير int ، وكذلك سعة سجل ال queue ممكن أن تكبر أو تصغر حسب الطلب، أما إضافة الحرف q بعد إغلاق القوس { وقبل الفاصلة المنقوطة، فإنه يعني أن q متغيرا نوعه هو queue والطابور queue هو بناء سجل يمكن تمثيله في الذاكرة على النحو شكل 4.8 :



شكل 4.8.

إذاً الطابور ال queue عبارة عن abstract data type نوع من أنواع هياكل البيانات المغلفة كما رأيناها من الشكل السابق، أي لا نستطيع التعامل مع عناصر ال front مباشرةً، ولا نستطيع أن نقول ;front=20 لأن front عنصراً مغلف، ولكن بواسطة q يمكن وضع أي قيمة في ال front فمثلاً ;q. front = 50 أمراً مقبول لدى C++.

4.3.3 كيفية التعامل مع الطابور queue

في البداية و عند تكوين ال queue، لا بد وأن تكون ال queue فارغة من أي بيانات، وبالتالي فإن front=rear ، وممكن أن نضع فيهما قيمة سالبة للدلالة على أن queue فارغة من البيانات، ولكن يفضل أن نضع فيهما أعلى قيمة لمؤشرال (array)، وعليه تكون queue فارغة عند تساوري قيمة q. rear مع q.front، ولهذا نضع القيمة المبدئية ممكن ان تكون 0 وممكن أن تكون q.front = q.rear = maxqueue

والآن دعنا نكتب الدالة empty

```
empty ( queue*pq)
{
        return ((pq -> front = = pq -> rear)? TRUE: FALSE);
}
```

False or True عند فحص هذه الدالة يتم إرجاع إحدى القيمتين

if (empty (q)) ويكون فحصها

بعد هذا دعنا نكتب الدالة insert ثم الدالة remove كي نتابع سيرهما عن قرب، وبالله التوفيق.

4.3.4 الدالة insert

وقبل أن نبدأ بكتابة العمليات على queue دعنا نتخيل ماذا سوف يحدث :

إفرض أن لدينا queue سعته القصوى 8 عناصر شكل 4.9

0	1	2	3	4	5	6	7

شكل 4.9.

فإذا أعطينا قيمة مبدئية ;front = 0; q.rear = 1 وإذا أردنا إدخال أي عنصر إلى ال queue نفحصها أولاً هل هناك فراغا أم لا؟ فإذا كان يوجد فراغا، فإننا ندخل العنصر المراد إدخاله، وذلك بأن نزيد q.rear++ ثم ندخل العنصر q.item[q. rear]=x حيث x هي العنصر المراد إدخاله.

لنفرض أن لدينا الطابور التالي شكل 4.10:

70	20	50	13	40	45	15	
0	1	2	3	4	5	6	
front							rear

شكل 4.10.

إفـرض أننـا حـذفنا 5عناصـر فيكون شكل queue

70	20	50	13	40	45	15	
0	1	2	3	4	5	6	
					front	rear	

إفـرض أننـا أضـفنا أيضـاً عنصرا إلى ال queue فيكون شكل ال queue. شكل 4.11

					45	15	
1	2	3	4	5	6	1	

شكل 4.11.

إفرض أننا حذفنا عنصرا فيكون في شكل 4.12 queue

					45	15	18
0	1	2	3	4	5	6	
					Front	rear	

						15	18
1	2	3	4	5	6	1	

شكل 4.12.

في هذا الوضع، إذا حاولنا أن نضيف عنصرا أخرا لا نستطيع، لأن rear يشير إلى نهاية سعة queue ولكن في الحقيقة لا يوجد فيها إلا عنصرين .

طبعاً كل الفروض السابقة مبنية على أن القيمة المبدئية لكل من front, rear تساوي 1- ، صفرا على التوالي، وعند الإدخال يتم التأكد من أن قيمة q.rear لا يساوي القيمة العظمى، عندها يزاد q.rear بمقدار واحدا ثم تدخل القيمة.

0	1	2	3	4	5	6	7

وعملية الإخراج أو الحذف تتم بأن تفحص قيمة q.front فإذا كانت أقل من القيمة العظمى تزاد قيمة q.front بمقدار واحدا وتكون الحالة خاصة عند ما تتساوى q.front، q.near مع القيمة العظمى، ولكن كما رأينا أنه يوجد عنصرين في الطابور ال queue عند النهاية وبقية الطابور ال queue فارغة، ولا نستطيع أن نضع فيها أى عنصر، إلا أن نحرك كل العناصر الموجودة في ال queue أثناء عملية الحذف إلى الأمام.

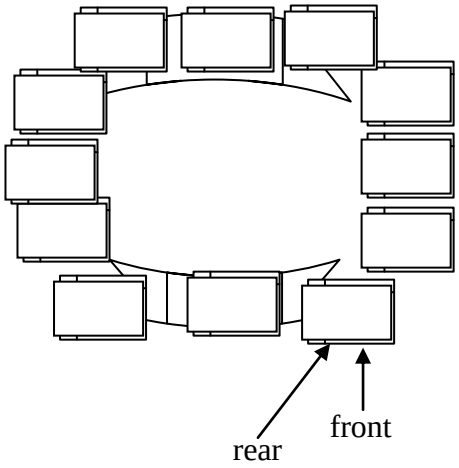
ولكن طبعاً هذه الطريقة غير فعالة حينما يكون الطابور ال queue طويلة، ولهذا قبل أن يتم كتابة دوال العمليات على queue ، لا بد وأن نوجد صورة أكثر فعالية من الصورة السابقة ، دعنا نتصور أن لدينا طابور queue ثم جعلناها بشكل داري.

4.3.5 Queue الطابور ذو الشكل الداري

يمكن أن نوجد صورة للطابور أكثر فعالية من الصورة السابقة وذالك على النحو التالي:

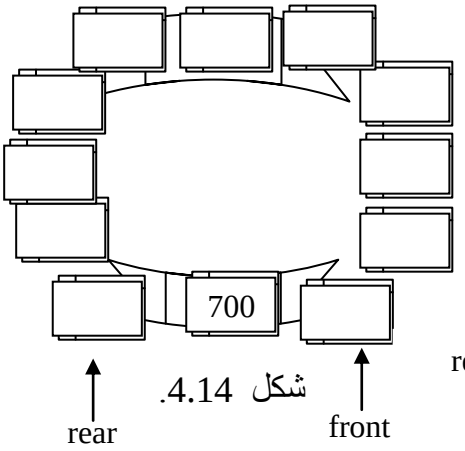
- لنعطي قيمة مبدئية لكل من q.rear, q.front ولتكن القيمة العظمى maxqueue
- لتكن حالة تساوي ال front مع ال rear هي **عدم وجود أي عنصر في** ال queue.
- لتكن القيمة العظمى في ال queue هي (n-1) حيث n هي عدد المواقع في queue.

لو فرضنا أنه في البداية لدينا طابور شكل 4.13،



شكل 4.13.

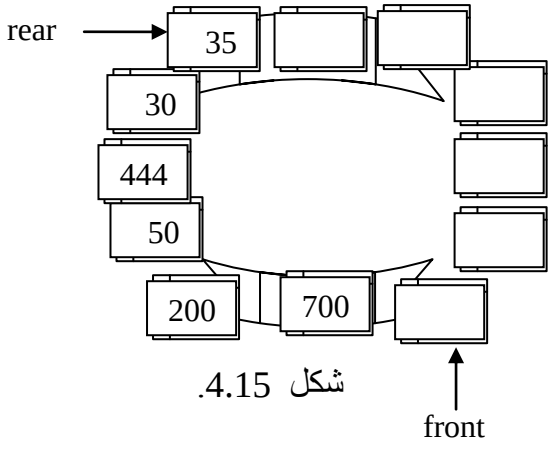
فإذا أدخلنا القيمة الأولى 700 يكون الطابور ال queue على شكل 4.14



شكل 4.14.

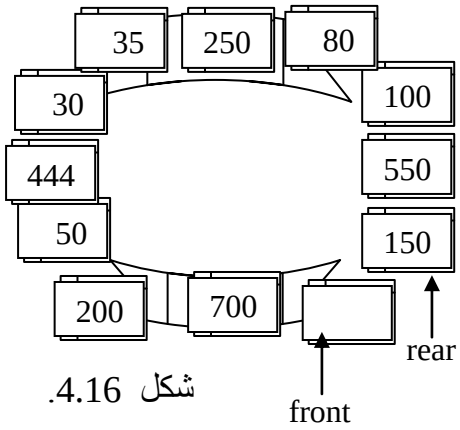
فإذا أدخلنا 5 قيم أخرى سوف

تكون ال queue على شكل 4.15



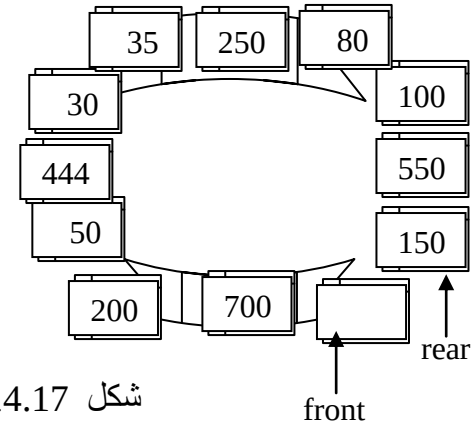
شكل 4.15.

فإذا أدخلنا 5 قيم
أخرى سوف تكون ال
queue على شكل 4.16



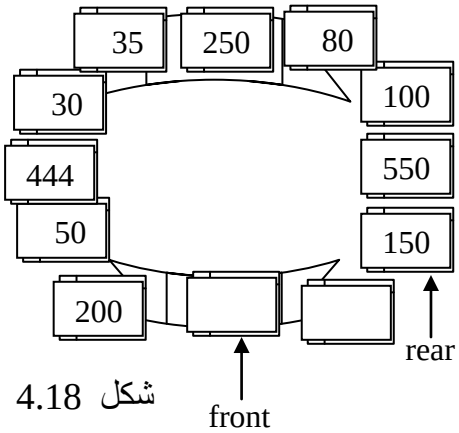
شكل 4.16.

في الشكل 4.17 عدد المواقع = 12 ، n-1=11 ، أي أننا سوف نترك دائماً موقعاً فارغاً من أي معلومات.



شكل 4.17.

فإذا حذفنا القيمة الأولى يكون الطابور ال queue على شكل 4.18

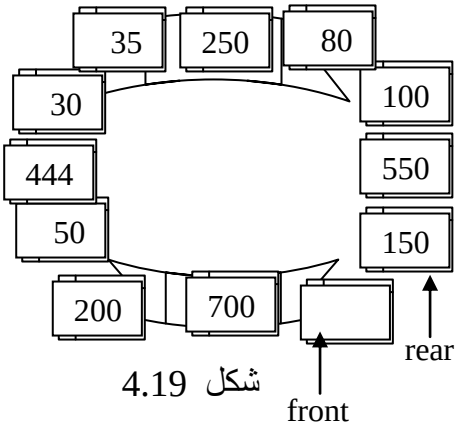


شكل 4.18

بناءً على الفرضيات السابقة، فإننا في عملية الإدخال أولا نزيد قيمة rear بمقدار واحد ثم نفحص قيمتي rear مع front إذا كانتا متساوية فإن الqueue مملوءة، أما إذا كان front =!rear فإن ال queue تعتبر فيها فراغ. وتكون عمليتي الحذف والإضافة بزيادة كل من ال front وال rear بمقدار واحد، فإذا وصلت قيمة أحداهما إلى القيمة العظمى تكون الزيادة بأن تعطي القيمة صفر (أي بداية الدائرة).

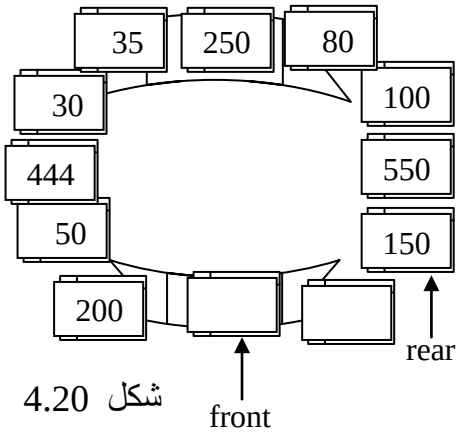
وسوف يتضح المفهوم بكتابة الدوال، والتطبيق عليهن بإذنه تعالى.

في هذه الحالة يكون قد وصل الطابور الqueue إلى القيمة العظمى شكل 4.19



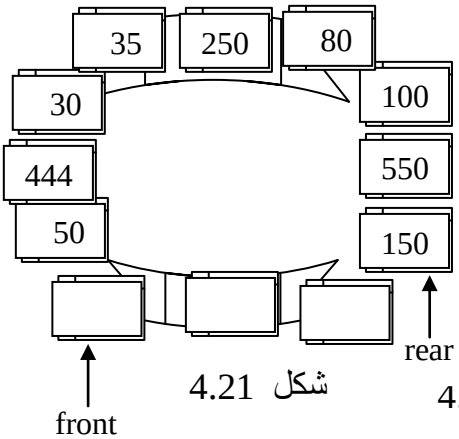
شكل 4.19

لو فرضنا أنه لدينا طابور شكل 4.20



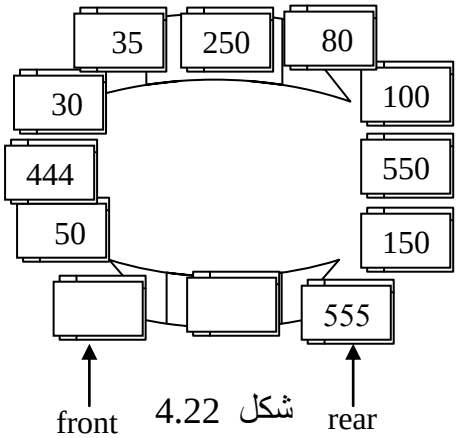
شكل 4.20

فإذا حذفنا قيمة منه يكون ال queue على شكل 4.21



شكل 4.21

فإذا أضفنا القيمة 555 إليه يكون الqueue على شكل 4.22



شكل 4.22

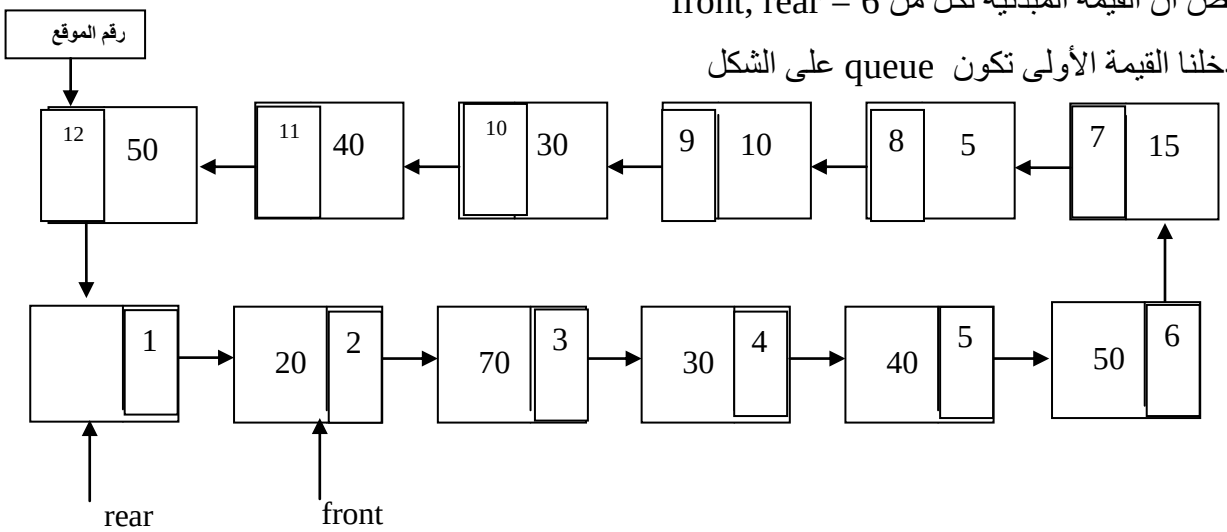
4.3.6 عملية الإدخال The insert operation

```
void insert ( queue*pq, int x)
{
 /*make place for new element */
 if (pq->rear == max_queue-1)
 pq -> rear = 0;
 else
 (pq ->rear ) ++;
 if (pq -> rear == pq ->front ) /* check for over flow */
{
  cout<<"queue is over flow";
  exit (1);
 }
 pq ->item [pq ->rear] = x;
 return;
}
```

لكي نتعرف هل عمل هذه الدالة صحيح أم لا نفرض أن القيمة العظمى = 12 ∴ Maxqueue -1 =11

ونفرض أن القيمة المبدئية لكل من front, rear = 6

إذا أدخلنا القيمة الأولى تكون queue على الشكل



هنا إذا حاولنا أن نضيف عنصرا آخر فلن نستطيع، لأنه سوف يحصل overflow للطابور.

4.3.7 الدالة remove

تستخدم الدالة remove لإزالة عنصر من الطابور queue

والآن نحاول أن تتم كتابة code الدالة remove لإزالة عنصر من queue.

```
int remove (struct queue* pq)
{
  if (empty (pq))
  {
    cout<<"queue under flow";    return -1;
    }
 if (pq->front == maxqueue-1)
  pq ->front = 0;
 else
    (pq-> front )++;
 return (pq->item [pq-> front]);
}
```

أما rear فإن قيمته تشير إلى آخر موقع حصل فيه الإضافة وعندما نريد إضافة عنصر إلى الطابور، أولاً نزيد قيمة rear بمقدار واحد، ثم نفحص هل هناك مكان فارغ في الطابور أم لا ؟ وعملية الفحص هنا هل queue ممتلئة أم لا، فإذا كانت ممتلئة فإن رسالة ما يجب أن تظهر لتوضح ذلك، ما لم فتتم عملية لإضافة.

نلاحظ أيضاً أن الدورة تأخذ مجراها، فإذا وصل أحد منهما front أو rear إلى القيمة العظمى لا يتم الحذف.

إدخال بعض التحسينات على الدالة Insert و الدالة remove

```
/* remove an element from queue*/
int remove (queue *pq)
{
if (empty (pq))
{
cout <<"queue underflow";   exit (1);
}
pq->front = (++pq ->front)% (maxqueue-1));
return (pq ->item [pq -> front]);
}
//--------------------------------------
void insert (queue *pq, int x)
{
pq -> rear = (++ (p->rear) % (max queue –1));
if (pq -> rear = = pq ->front )
{
cout<<" queue overflow";
exit (1);
}
pq ->item [pq ->rear] = x;
return;
```

```
}
//this program to implement  queue using some function
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
# define maxqueue 12
 # define TRUE 1
 # define FALSE 0
struct queue
 {
    int item [maxqueue];
    int front, rear;
  }q;
//-------------------------------------
 void pt(queue *p)
 {
  int i;
  i=p->front;
  if(i==maxqueue-1)
    i=0;
  else
     i++;
  while(i!= p->rear)
  {
   if(i==maxqueue-1)
    i=0;
   cout<< p->item[i]<<" ";
   i++;
    }
    cout<< p->item[i]<<" ";
   return ;
  }
//-------------------------------------
  int empty ( queue *pq)
  {
    return ((pq->front == pq->rear)? TRUE: FALSE);
  }
//-------------------------------------
  void insert ( queue *pq)
  {
  /*make place for new element */
          int x;
   x=rand()%1000;
  if (pq->rear == maxqueue-1)
    pq->rear = 0;
  else
  (pq->rear ) ++;
  if (pq->rear == pq->front ) // check for over flow
   {
   cout<<"queue is over flow";
```

```
            return; //exit(1);
 }
 cout<<pq->rear<<" ";
 pq->item[pq->rear] = x;
 return;
}
//-------------------------------------
 int remove (queue *pq)
  {
   if (empty (pq))
    {
      cout<<"queue under flow"<<endl;
       exit (1);
    }
   if (pq->front == maxqueue-1)
    pq->front = 0;
  else
    (pq->front )++;
 return (pq->item [pq->front]);
}
//-------------------------------------
 int main (  )
{
 queue a,*p;
 int i,n,s;
 p=&a;
 p->front=maxqueue-1;
 p->rear=maxqueue-1;
 cout<<" enter no element to be added to the queue ";
 cin>>n;
 cout<<endl;
  for(i=0;i<n;i++)
     insert(p);
    do
 {
          cout<<endl<<"----------------------------"<<endl;
       cout<<" 1 : add value to the queue "<<endl;
          cout<<" 2 : delete value from the queue "<<endl;
          cout<<" 3 : print the value from the queue "<<endl;
          cout<<" 4 : exit "<<endl;
          cout<<" enter your selection "<<endl;
          cout<<endl<<"----------------------------"<<endl;
           cin>>s;
           switch(s)
           {
            case 1 :  insert(p);  break;
            case 2 :  cout<<remove(p);  break;
            case 3 :  pt(p);   break;
            case 4 : cout<<endl<<" by by ";  exit(0);
            default  : cout<<" out of range "<<endl;  break;
```

```
        }
      } while(s!=4);
      cout<<endl<<" by by ";
   cin>>n;
 return 0;
}
```

الخلاصة

1. **يمكن تصميم مجموعات مختلفة من هياكل البيانات حسب الطلب، مثل السجلات والقوائم المتصلة و الطوابير والمكادس والاشجار.**
2. **يمكن جعل الهياكل تحتوي على مجموعات مختلفة من البيانات**
3. **حجم هذه الهياكل ثابتة ومحددة من قبل المبرمجين أثناء كتابة البرنامج**

الباب الخامس

5 الهياكل الخاصة ذوات الأحجام المتغيرة

الهدف من هذا الباب هو الأتي:

1. **القدرة على تكوين هياكل خاصة ذوات أحجام متغيرة تستخدم كهياكل تحتوي على مجموعات مختلفة من البيانات مثل السجلات والقوائم المتصلة و الطوابير والمكادس والاشجار.**
2. **التعامل مع الهياكل الخاصة ذوات أحجام متغيرة.**

5.1 حجز مواقع في الذاكرة أثناء التنفيذ Dynamic memory

تكلمنا فيما سبق على أن هياكل البينات لها أشكال مختلفة منها tables, stacks, queues, structure وبينا مميزات كل على حدة، ولكن العيب الأساس لمثل هذه الأنواع من هياكل البيانات structur data types هو أن حجمها يجب أن يجدد مسبقاً – أي أثناء كتابة البرنامج – وهناك الكثير من الحالات قد لا يتم فيه معرف كمية البينات المطلوبة أو المعلومات الناتجة، ولهذا يتم حجز كمية كبيرة من الذاكرة وذلك لنكون في مأمن من النقص في الذكرة المحجوزة.

طبعاً هذا السلوك قد لا يكون محبذاً في معظم الحالات، إذ أن حجم الذاكرة المخصص غالبا ما يكون محدودا، وبالتالي لجأ العلماء في علوم الكمبيوتر للبحث عن مخرج آخر يحقق المطلوب ويكون فعال وإقتصادي في إستخدام الذاكرة وهو حجز المواقع المحتاجة أثناء التنفيذ وليس مسبقاً وهو بما يسمى **Dynamic memory allocation. وعلى ذالك يمكن تقسيم الحجز** للذاكرة على النحو:

- الحجز للمواقع مسبقاً يسمى Static memory allocation. مثل المصفوفات.
- الحجز للمواقع أثناء التنفيذ يسمى Dynamic memory allocation ،وهو ما سوف نراه لاحقا.

5.2 كيفية حجز مواقع في الذاكرة أثناء التنفيذ

من الممكن حجز مواقع في الذاكرة أثناء تنفيذ البرنامج في معظم لغات البرمجة، و من ضمن تلك اللغات لغة C++ ، إن مكتبة C++ تحتوي على دوال قادرة على حجز مواقع في الذاكرة أثناء تنفيذ البرنامج، من هذه الدوال new, malloc, calloc ويمكن إستدعاؤهن لحجز مواقع في الذاكرة، ثم الوصول إلى تلك المواقع وذالك بوضع مؤشر إلى بداية الموقع المحجوز، ومن ثم وضع بيانات فيه. والمثال التالي يوضح إستخدام الدالة new لحجز موقع في الذاكرة على هذا النحو:

```
p=new data-type ;
```

حيث p يجب أن يكون من نوع مؤشر pointer، لأنه سوف يحتوى على عنوان

و data-type هو نوع البيان الذي يراد حجز المواقع لأجله.

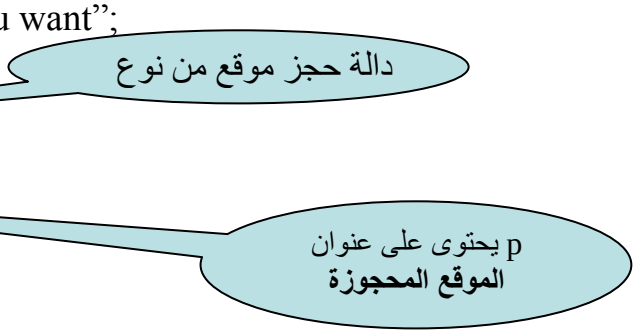
هذه الدالة new سوف ترجع مؤشرا(يحتوى على عنوان) يشير إلى بداية الموقع المحجوز، هذا إذا نجحت عملية الحجز، أو ترجع مؤشر يشير إلى Null ، إذا لم تنجح عملية الحجز، ولهذا السبب يفضل دائماً التأكد بعد طلب الحجز لموقع ما، من أن المؤشر لا يشير إلى Null.

وسوف نوضح إستخدام الدالة new بكتابة البرنامج الحالي:

اكتب برنامجا يحجز 10 موقع من نوع int عن طريق إستخدام الدالة new ثم يضع قيم في المواقع ويطبهن.

الحل

```
#include<iostream.h>
main ( )
{
int    i, *p,n =11;
//cout <<"in Enter the no. times you want";
 for (i=0;i<n;i++)
   {
  p =new int ;
  p=&i;
  cout<<*p<<"  ";
  }
return 0;
}
```

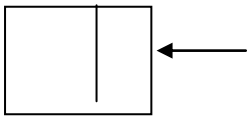


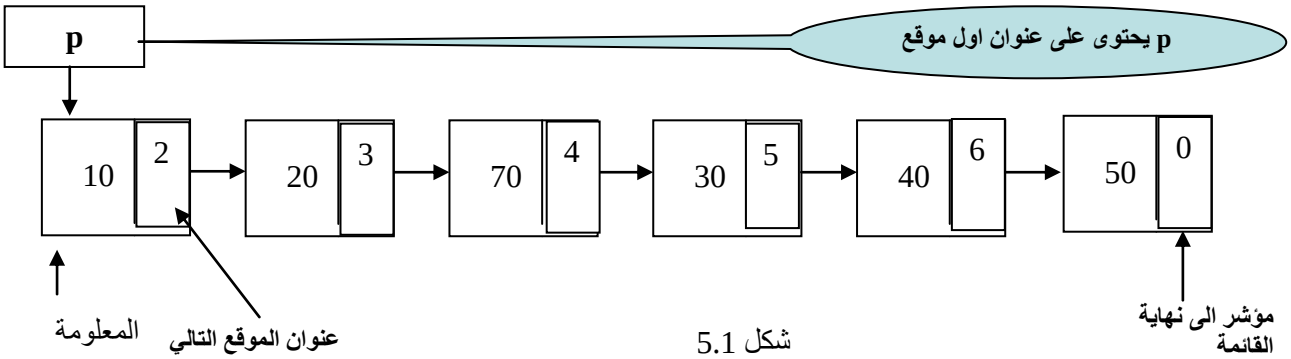
مخرجات هذا البرنامج عباره عن قيم عناوين المواقع التي تم حجزهن عن طريق الدالة .new وهن على هذا النحو: 10 9 8 7 6 5 4 3 2 1 0 هذا إذا تم إدخال قيمة n=11

5.3 القوائم المتصلة Linked lists

Linked list is a very common data structure often used to store similar data called (nodes) in memory.

شكل 5. 0

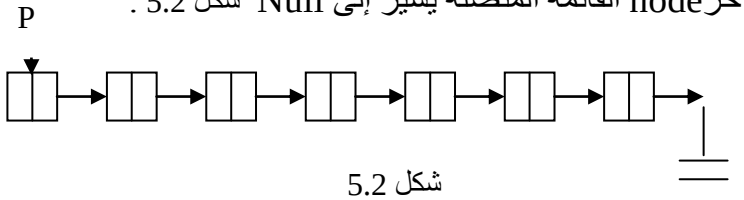




شكل 5.1

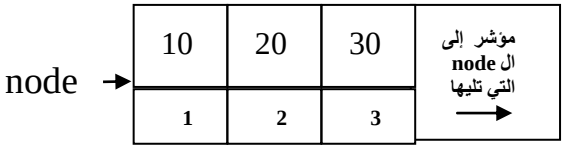
القوائم المتصلة عبارة عن هياكل من هياكل البينات تسمى linked lists ، يمكن تكوين القوائم المتصلة من مجموعة من nodes كما في شكل 5.0 ، كل node تحتوي على مؤشر يشير إلى ال node التي تليها، شكل 5.1 وكذالك كل node ممكن أن تحتوي على حقل واحد أو عدة حقول من البيانات المختلفة، و أقل node يمكن أن تحتوي على مؤشر شكل 5.0 يشير إلى node التي تليها وحقلا واحدا آخر يحتوي على البيانات، أي أن اقل node يجب أن تحتوي حقلين أحدهما مؤشر يشير إلى node التي تليها وحقلا آخر يحتوي على البيانات اللازمة.

كل قائمة متصلة تكون مكونة من مجموعة من ال nodes، المتشابهة، هذه المجموعة تحتاج إلى مؤشر يشير إلى بداية القائمة، ويكون المؤشر عبارة عن عنوان لموقع ما في الذاكرة. في الشكل 5.1، p عبارة عن مؤشر يشير إلى بداية القائمة المتصلة ، آخرnode القائمة المتصلة يشير إلى Null شكل 5.2 .



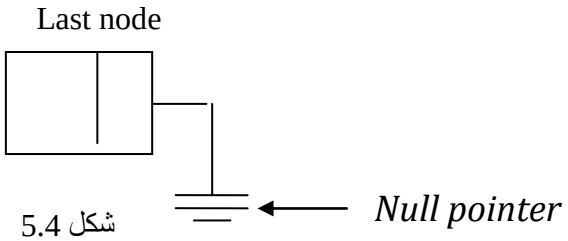
شكل 5.2

ملحوظة محتويات كل one node توجد في مواقع متجاورة في الذاكرة ، شكل 5.3



شكل 5.3

ولكن ليس بالضرورة أن تكون جميع nodes متجاورة في الذاكرة، بل إن linked lists صممت على أن تكون كل node في موقع ما في الذاكرة، والذي يربط بين nodes هو المؤشرات، حيث أن كل node تحتوي على مؤشر هذا المؤشر يحتوي على عنوان ال node التالية وهكذا إلى أن يحتوي المؤشر الأخير على Null ، شكل 5.4 وكلمة 'Null' عبارة عن رمز يدل على نهاية السلسلة (القائمة).

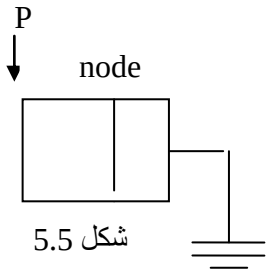


شكل 5.4

بهذه الطريقة يتم تجاوز مشكلة array من حيث ان الحجم تابت، وطالما قد استطعنا إيجاد أو حجز مواقع في الذاكرة أثناء التنفيذ، فيمكن تكوين قائمة من هياكل البيانات ذات حجم متغير وهي ما تسمى بالقوائم المتصلة linked list.

5 . 2 .1 Linked lists implementation كيفية بناء القوائم المتصلة المتغيرة

لكي نقوم ببناء القوائم المتصلة متغيرة الأحجام، ببساطة نتبع التالي:



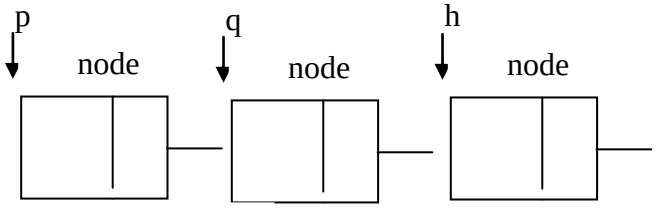
شكل 5.5

1. نعرف سجل node تحتوي على الحقول المطلوبة

نعرف أيضاً حقلا إضافيا من نوع مؤشر إلى node شكل 5.5

2. نحجز الذاكرة اللازمة لكل node
3. نحتفظ بمؤشر كعنوان ل node ثابتة
4. نضع البيانات في الحقول المخصص لها
5. نضع في الحقل الخاص بالمؤشر عنوان node التالية وهكذا.

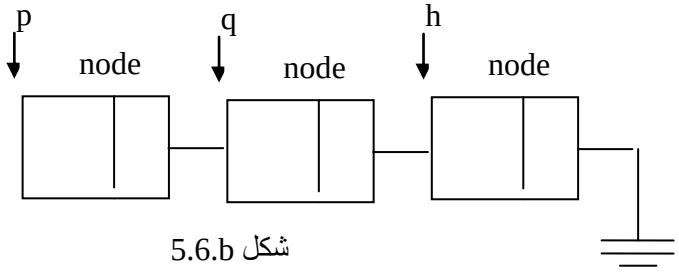
وهنا مثال لتكوين قائمة مكونة من ثلاث nodes شكل 5.6, a. 5.6, b .



شكل 5.6.a

```
p= new node;
q= new node;
h= new node;
```

من خلال هذه الأوامر يتم تكوين ثلاث nodes شكل 5.6.a



شكل 5.6.b

ومن خلال هذه الأوامر يتم ربط nodes الثلاث شكل 5.6.b

```
p->link=q;
q->link=h;
h->link=Null;
```

ولكي نوضح الفكرة أكثر فيتم كتابة البرنامج التالي لتكوين قائمة مكونة من ثلاث nodes

```
struct node
{
int data;
node *link;
};
main ( )
{
node *p, *q, *h;
p= new node;
p -> data = 20;  p ->  link = null; H= p;
q= new node;
q ->  data = 30;  q ->  link = p;   p= q;
q= new node;
q ->  data = 40;
 q ->  link = Null;
p-> link= q;
p= H;
while (p! Null)
 {
  cout<< p->  data;
  p = p -> link;
 }
}
```

تعريف سجل node تحتوي على الحقول المطلوبة

تعريف حقلا إضافيا من نوع مؤشر إلى node

تعريف مؤشرات من نوع مؤشر إلى node

حجز موقع الذاكرة node و جعل المؤشر p يحتوى على عنوان node **الموقع**

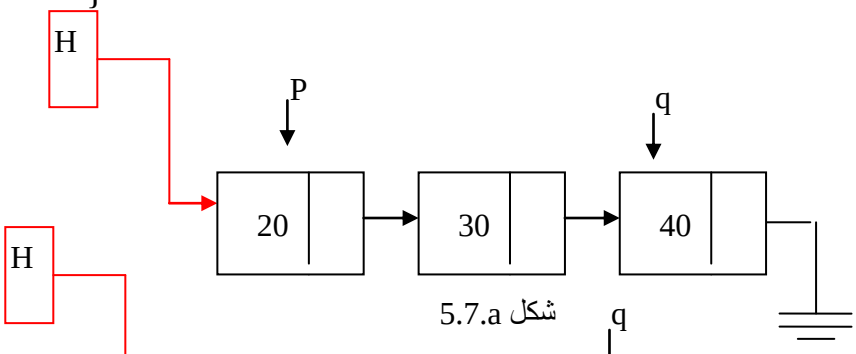
وضع بيانات في الحقول المخصص لها من الموقع node

حجز موقع آخر في الذاكرة من نوع node و جعل المؤشر q يحتوى على عنوانه

هذا loop لطباعة محتوى ال nodes

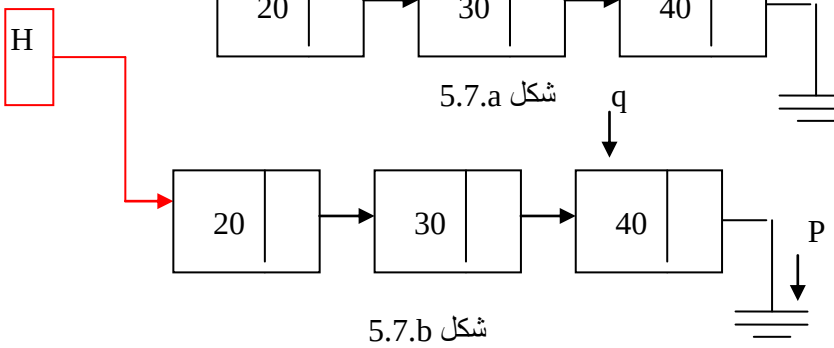
المؤشرH يشير إلى بداية القائمة

الشكل5.7.a يوضح تمثيل المواقع والبيانات في الذاكرة قبل الطباعة.



شكل 5.7.a

والشكل 5.7.b يوضح تمثيل المواقع والبيانات في الذاكرة بعد الطباعة.

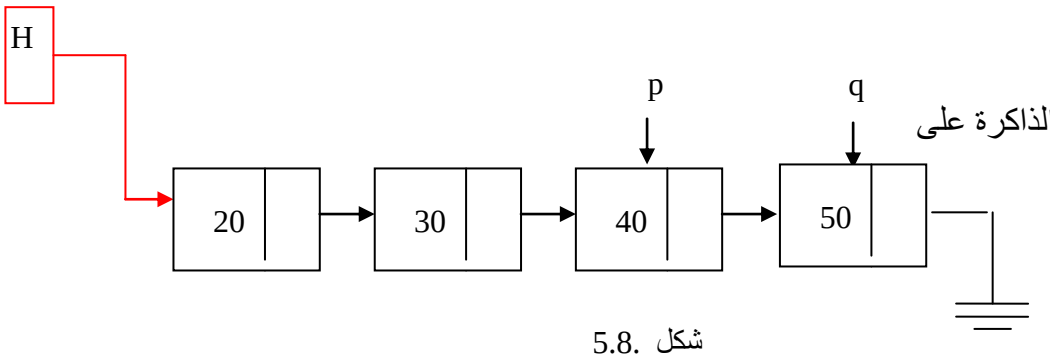


شكل 5.7.b

وإذا أردنا اضافة node إلى نهاية القائمة، فيتم إضافة الأوامر التالية قبل ال while: وذالك لحجز موقع آخر في الذاكرة من نوع node و جعل المؤشر q يحتوى على عنوانه، ثم لوضع بيانات في الحقول المخصص لها من الموقع node أما ال while فهي للوصول إلى نهاية القائمة.

```
p=q;
q= new node;
q ->  data = 50;
p ->  link = q;
p ->  link = Null
```

فيمكن تخيل شكل البيانات في الذاكرة على النحو التالي شكل .5.8:



شكل .5.8

طباعة محتوى القائمة

فلو أردنا طباعة محتوى القائمة، فيتم إضافة الأوامر التالية

أولا جعل المؤشر p يشير الى عنوان اول node ، وهي المشار إليها بالمؤشر H عن طريق الأمر p=H; .

ثانيا نحتاج إلى while loop للمرور على كل محتوى القائمة، ويكون الخروج من while loop عندما يصل المؤشر p إلى Null .

```
p=H;
while (p! Null)
{
cout<< p->  data;
p = p -> link;
}
```

جعل المؤشر p يشير الى عنوان اول node

طباعة محتوى node

جعل المؤشر p يشير الى عنوان ال next node

اكتب برنامجا يكوين قائمة متصلة مكونة من 5 nodes ، شكل 5.9. ثم يضع في كل node رقم تكوينها، ثم يطبع المخرجات .

الحل

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  };
main( )
{
 int i;
node *p,*H;
H= new node;
H->data=1;
H->next=Null;
for(i=2 ;i<6;i++)
 {
  p= new node;
  p->data=i;
  p->next=H;
  H= p;
 }
 while (p! Null)
  {
    cout<< p->data;
    p = p-> next;
  }
return 0;
}
```

توصيف محتويات السجل nodes المستخدمة في القائمة المتصلة

حجز أول node

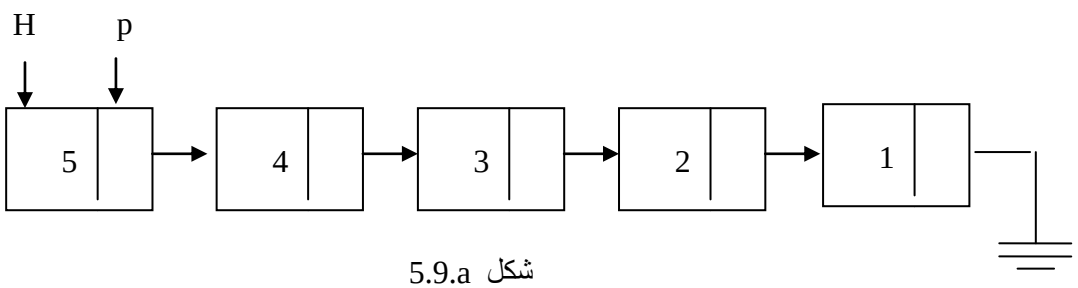
وضع البيانات فى أول node

حجز node جديدة

وضع البيانات في node الجيدة

ربط ال node الجيدة مع node القديمة

طباعة محتوى node



شكل 5.9.a

مخرجات هذا البرنامج سوف تكون على هذا النحو:

5 4 3 2 1

و يمكن كتابة البرنامج ايضا على هذا النحو

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  };
main( )
{
 int i;
node *p,*H, *q;
H= new node;
H->data=1;
q=H;
for(i=2 ;i<6;i++)
 {
  p= new node;
  p->data=i;
  q->next= p;
  q= p;
  }
 p->next= Null;
p=H;
  while (p! Null)
    {
      cout<< p->data;
      p = p-> next;
    }
return 0;
}
```

حجز أول node

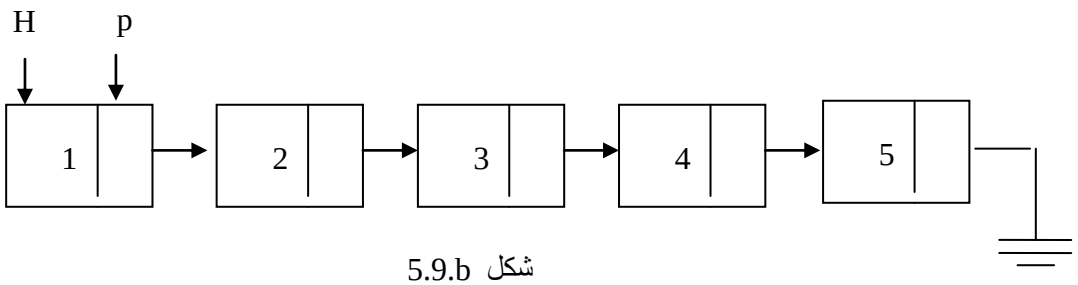
وضع البيانات فى أول node

حجز node جديدة

وضع البيانات في node الجيدة

ربط ال node الجيدة مع node القديمة

طباعة محتوى node



شكل 5.9.b

مخرجات هذا البرنامج سوف تكون على هذا النحو:

1 2 3 4 5

والبرنامج التالي لتكوين قائمة متصلة مكونة من n nodes و يطبع المخرجات في ملف اسمه "da2.cpp"

```
#include<fstream.h>
#include<iomanip.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  }*S;
main( )
{
 int i,n;
      fstream Dataf1;
     Dataf1.open("da2.cpp",ios::out);
  S=NULL;
   node *p,*q;
    Dataf1<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
       cout<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
       cin>>n;     S=new node;
   S->next= NULL;
   S->data=rand()%1000;
   cout<<endl<<" during generation"<<endl;
   Dataf1<<endl<<" during generation"<<endl;
   cout<<" ["<<setw(3)<<S->data<<"]";
   Dataf1<<" ["<<setw(3)<<S->data<<"]";
   q=S;
  for(i=1;i<n;i++)
  {
        p=new node;
    q->next= p;
    p->data=rand()%1000;
    p->next= NULL;
    Dataf1<<" ["<<setw(3)<<S->data<<"]"
    cout<<" ["<<setw(3)<<p->data<<"]";
       if((i+1)%10==0)
     {
              cout<<endl;
     Dataf1<<endl;
     }
   q=p;
   p=S;
   cout<<endl<<"_____"<<endl;
```

توصيف محتويات السجل nodes المستخدمة في القائمة المتصلة

هنا يتم القيام بإيجاد موقع جديد ل node ثم توليد بيانات ووضعها في محتويات ال node والربط بين ال nodes لتكوين القائمة المتصلة

هنا يتم القيام بطباعة محتويات ال nodes عند تكوين القائمة المتصلة

```
Dataf1<<endl<<"_____"<<endl;
cout<<endl<<" after generation"<<endl;
Dataf1<<endl<<" after generation"<<endl;
   if(p==NULL)
    {
          Dataf1<<" empty list "<<endl;
  cout<<" empty list "<<endl;
          return 0;
    }    i=0;
    while (p!=NULL)
    {
     cout<<" ["<<setw(3)<<p->data<<"]
  Dataf1<<" ["<<setw(3)<<S->data<<"]
    if((i+1)%10==0)
  {
          Dataf1<<endl;
      cout<<endl;
  }   i++;
      p=p->next;
     }
  cout<<endl<<" enter any no";     Dataf1 <<endl<<" enter any no";      cin>>i;
  Dataf1.close();
     return 0;
    }
```

هنا يتم القيام بطباعة محتويات ال nodes الموجودة بالقائمة المتصلة بعد التكوين

عند تنفيذ هذا البرنامج سوف تظهر النتائج التالية في الملف لذي اسمه "da2.cpp"

enter int the number of nodes to be constructed as stack

during generation

[346] [346] [346] [346] [346] [346] [346] [346] [346] [346]

[346] [346] [346] [346] [346] [346] [346] [346] [346] [346]

[346] [346] [346] [346] [346] [346] [346] [346] [346] [346]

---

after generation

[346] [346] [346] [346] [346] [346] [346] [346] [346] [346]

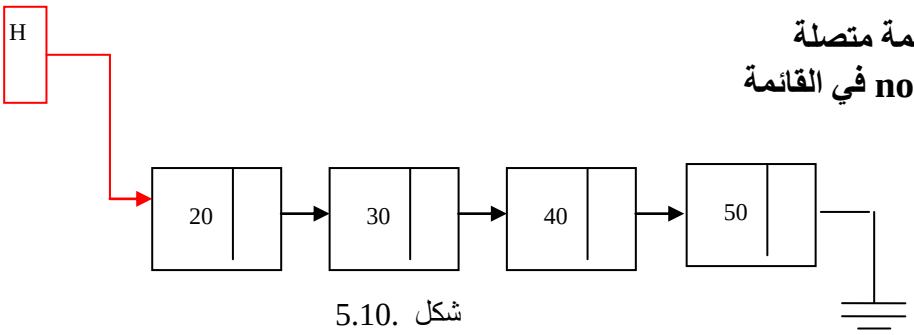
[346] [346] [346] [346] [346] [346] [346] [346] [346] [346]

[346] [346] [346] [346] [346] [346] [346] [346] [346] [346]

enter any no

5.2.2 إضافة node إلى قائمة متصلة

أ- إضافة node إلى بعد node في القائمة

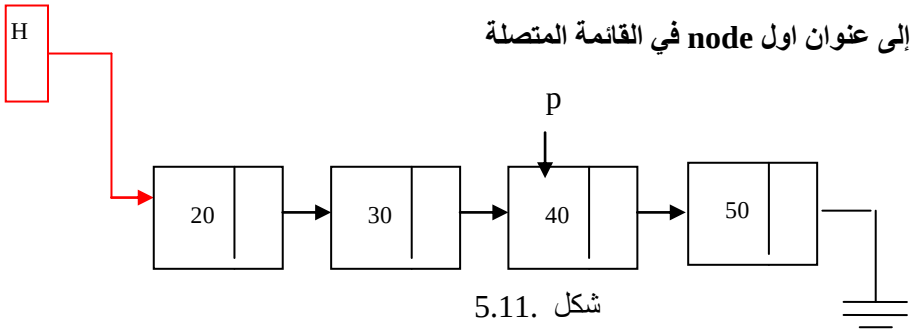


شكل 5.10.

فلو كان لدينا قائمة متصلة في الذاكرة على شكل 5.10.

، ونريد إضافة، node تحتوى على البيانات 45 بعد ال node التي تحتوى على البيانات 40 في القائمة المتصلة، فيتم عمل الخطوات التالية

1. **جعل المؤشر p يشير إلى عنوان اول node في القائمة المتصلة**



شكل 5.11.

2. **البحث عن ال node التي تحتوى على البيانات 40 في القائمة المتصلة (إذا وجدت)**

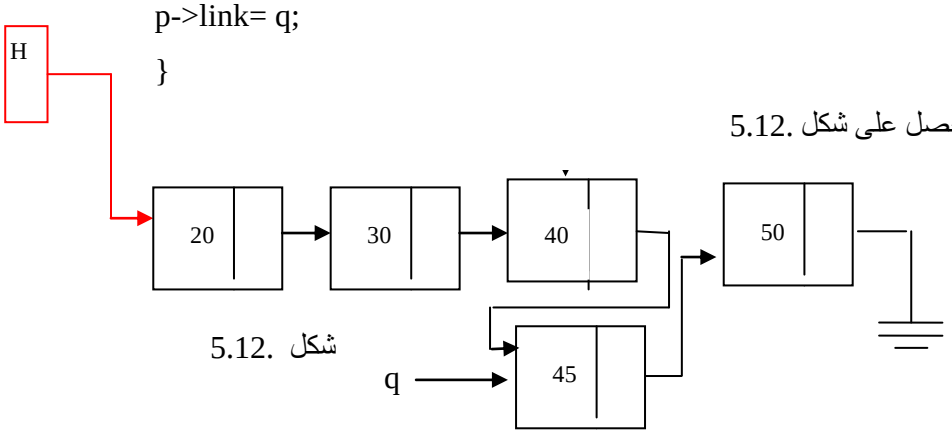
و يتم ذالك عن طريق الأوامر التالية شكل 5.11.

```
p=H;
while (p! Null && p->data!=40)
  p = p ->link;
```

3. ثم يتم ربط ال **node التي تحتوى على 45 (بعد تكوينها عن طريق المؤشر q) في القائمة المتصلة و يتم ذالك عن طريق الأوامر التالية**

```
if (p!= NULL)
{
q->link= p->link;
p->link= q;
}
```

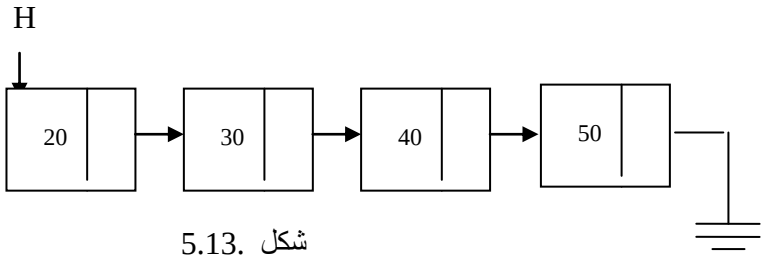
وبعد تنفيذ الأوامر السابقة نحصل على شكل 5.12.



شكل 5.12.

أ- إضافة node إلى قبل node في القائمة

فلو كان لدينا قائمة متصلة في الذاكرة على شكل 5.13.، **ونريد إضافة، node تحتوى على البيانات 35 إلى قبل ال node التي تحتوى على البيانات 40 في القائمة المتصلة، فيتم عمل الخطوات التالية**

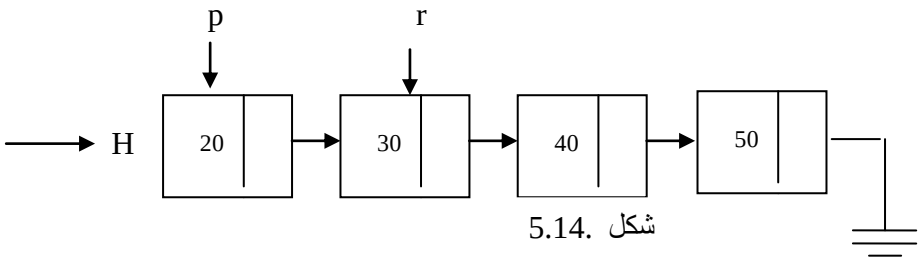


شكل 5.13.

1. **جعل المؤشر p يشير إلى عنوان اول node في القائمة المتصلة**
2. **البحث عن ال node التي تكون قبل ال node التي تحتوى على البيانات 40 في القائمة المتصلة (اذا وجدت)، و يتم ذالك عن طريق تمرير مؤشرين احدهما سابق p والأخر لاحق r.**

و يتم ذالك عن طريق الأوامر التالية

```
p=H;
while (p! Null && p->data!=40)
{ r=p;
  p = p->link;
}
```

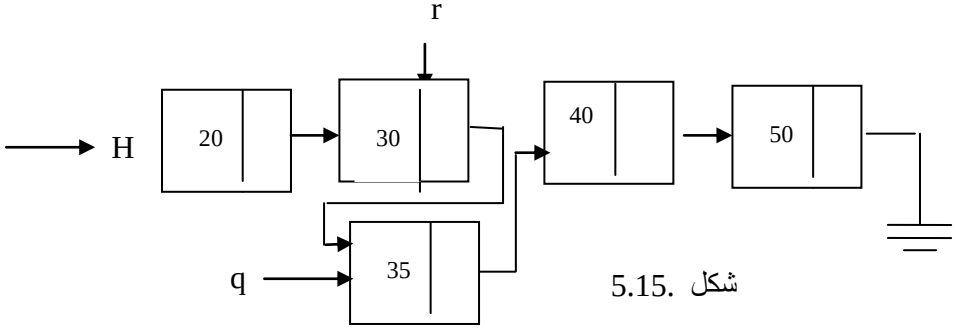


شكل 5.14.

ثم يتم ربط **ال node التى تحتوى على 35 (بعد تكوينها عن طريق المؤشر q) في القائمة المتصلة** شكل 5.15.

و يتم ذالك عن طريق الأوامر التالية

```
if (p!= NULL)
{
q->link= r->link;
r->link= q;
q->link= p;
}
```

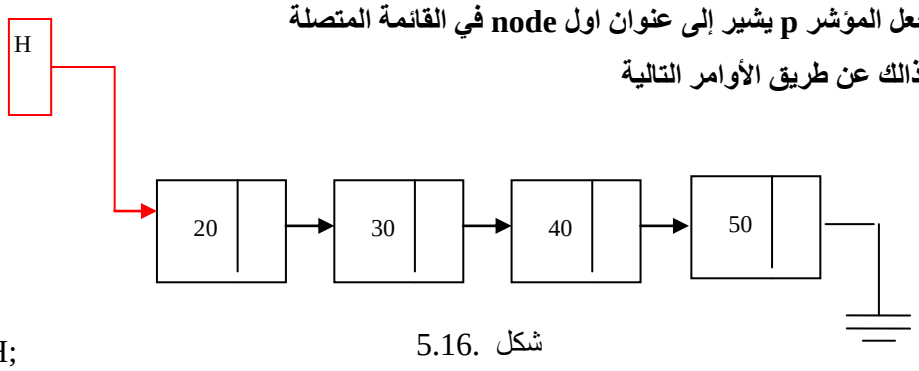


شكل 5.15.

ج- إضافة node إلى بداية القائمة 5 . 2 . 4

فلو كان لدينا قائمة متصلة في الذاكرة على شكل 5.16.، **ونريد إضافة، node تحتوى على البيانات 10 إلى بداية القائمة المتصلة، فيتم عمل الخطوات التالية**

1. **جعل المؤشر p يشير إلى عنوان اول node في القائمة المتصلة و يتم ذالك عن طريق الأوامر التالية**

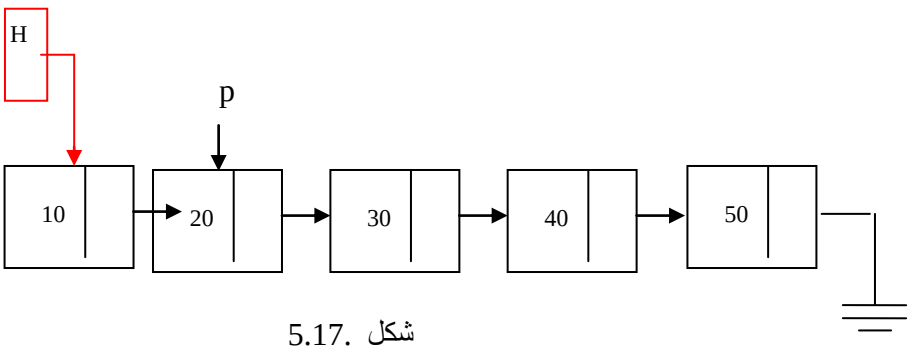


شكل 5.16.

```
p=H;
```

2. **ثم يتم ربط ال node التى تحتوى على 10 (بعد تكوينها عن طريق المؤشر q) في القائمة المتصلة** شكل 5.17. **و يتم ذالك عن طريق الاوامر التالية**

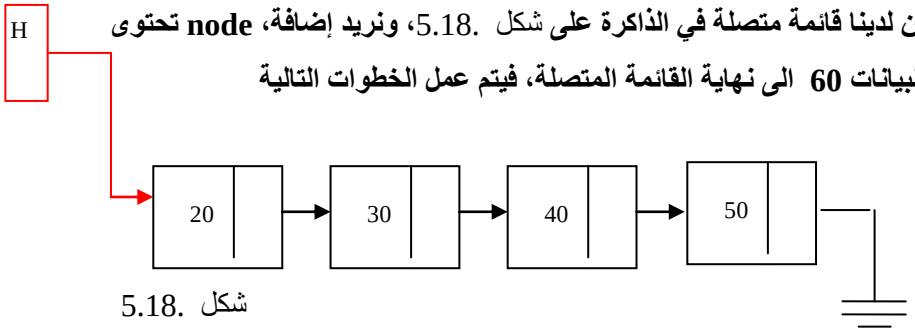
```
if (p!= NULL)
{
 q->link= p;
 p=q;
 }
```



شكل 5.17.

د- إضافة node إلى نهاية القائمة 5. 2 . 5

فلو كان لدينا قائمة متصلة في الذاكرة على شكل 5.18.، **ونريد إضافة، node تحتوى على البيانات 60 الى نهاية القائمة المتصلة، فيتم عمل الخطوات التالية**



شكل 5.18.

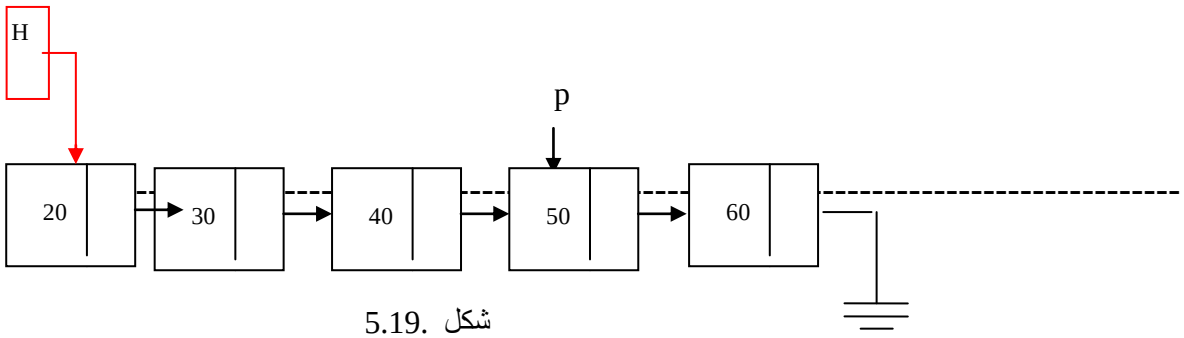
1. **جعل المؤشر p يشير إلى عنوان أول node في القائمة المتصلة**
2. **تمرير المؤشر p إلى نهاية القائمة المتصلة (إذا وجدت)**

و يتم ذالك عن طريق الأوامر التالية

```
p=H;
while (p->link! Null)
  p = p →link;
```

ثم يتم ربط ال **node التى تحتوى على 60 (بعد تكوينها عن طريق المؤشر q) في القائمة المتصلة** شكل 5.19. **و يتم ذالك عن طريق الأوامر التالية**

```
if (p!= NULL)
{
p->link= q;
p->link= NUll;
}
```

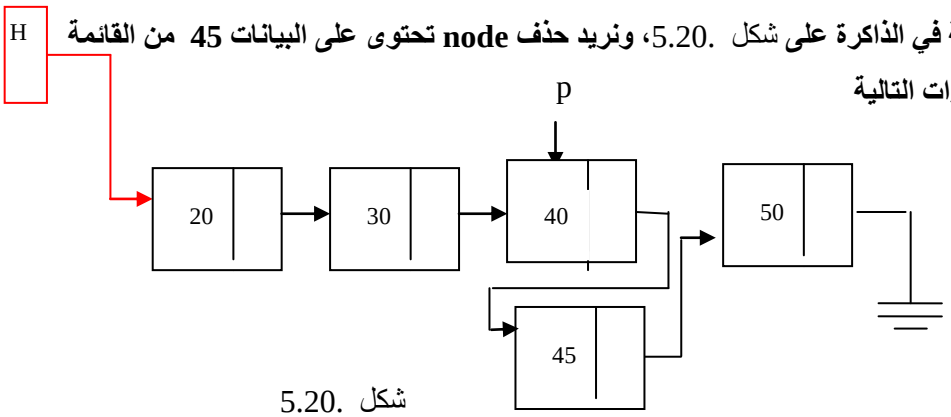


شكل 5.19.

6. 2 . 5 حذف node من قائمة متصلة

أ- 5.2.6.1 حذف node من بعد node في القائمة

فلو كان لدينا قائمة متصلة في الذاكرة على شكل 5.20.، ونريد حذف node تحتوى على البيانات 45 من القائمة المتصلة، فيتم عمل الخطوات التالية



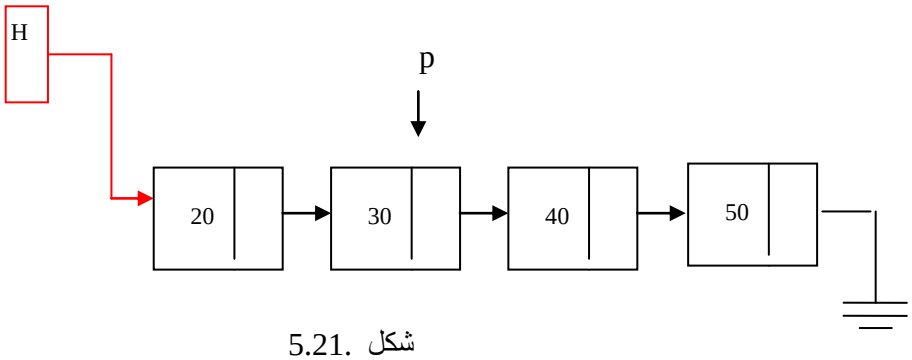
شكل 5.20.

1. جعل المؤشر p يشير الى عنوان اول node في القائمة المتصلة
2. البحث عن ال node التي تكون قبل ال node التي تحتوى على البيانات 45 في القائمة المتصلة (اذا وجدت)، و يتم ذالك عن طريق تمرير مؤشر.و يتم ذالك عن طريق الاوامر التالية

```
p=H;
if (p!= NULL)
while (p->link != Null &&  p->link->data!=45)
   p = p -> link;
```

3. ثم يتم فصل ال node التى تحتوى على 45 ثم حذفها من القائمة المتصلة شكل 5.21. و يتم ذالك عن طريق الاوامر التالية

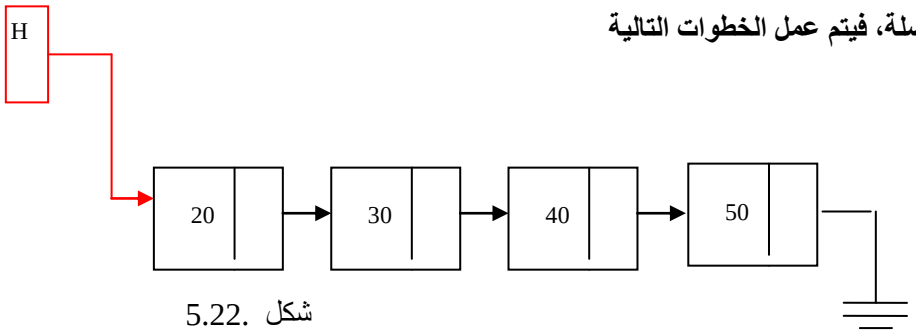
```
if (p!= NULL)
  {
   q= p->link;
   p->link= q->link;
   q->link=NULL;
   delete(q);
  }
```



شكل 5.21.

ب- حذف node من قبل node من القائمة 2 . 6. 2 . 5

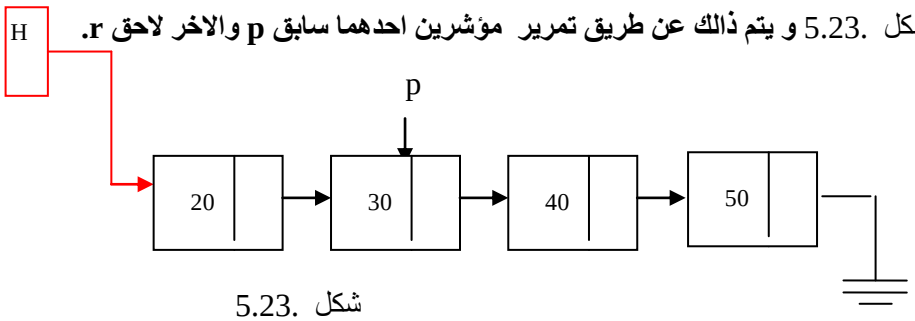
فلو كان لدينا قائمة متصلة في الذاكرة على شكل 5.22.، **ونريد حذف node قبل ال node التي تحتوى على البيانات 40 في القائمة المتصلة، فيتم عمل الخطوات التالية**



شكل 5.22.

1. **جعل المؤشر p يشير الى عنوان اول node في القائمة المتصلة**
2. **البحث عن ال node التي تكون قبل ال node التي تحتوى على البيانات 40 في القائمة المتصلة (اذا وجدت)،**

شكل 5.23. **و يتم ذالك عن طريق تمرير مؤشرين احدهما سابق p والاخر لاحق r.**



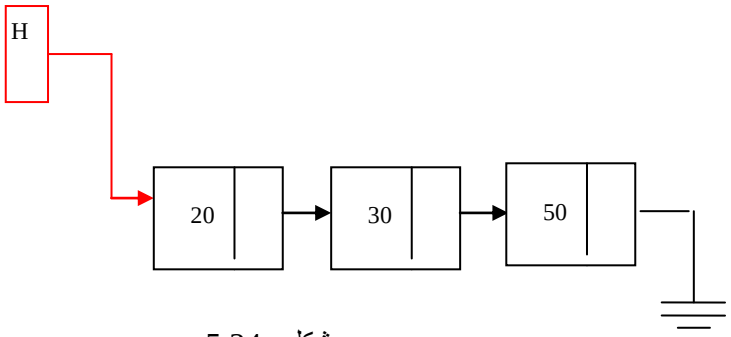
شكل 5.23.

و يتم ذالك عن طريق الاوامر التالية

```
p=H;
while (p! Null && p->data!=40)
{ r=p;
  p = p -> link;
}
```

3. ثم يتم ثم يتم فصل ال **node ثم حذفها من القائمة المتصلة،** شكل 5.24. **و يتم ذالك عن طريق الاوامر التالية.**

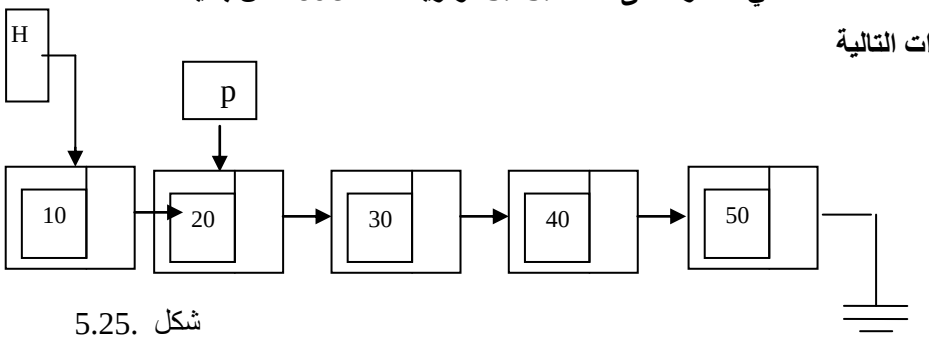
```
if (p!= NULL)
{
q->link= r->link;
r->link= q;
q->link= p;
}
```



شكل 5.24.

ج- حذف node من بداية القائمة

فلو كان لدينا قائمة متصلة في الذاكرة على شكل .5.25، **ونريد حذف node من بداية القائمة المتصلة، فيتم عمل الخطوات التالية**



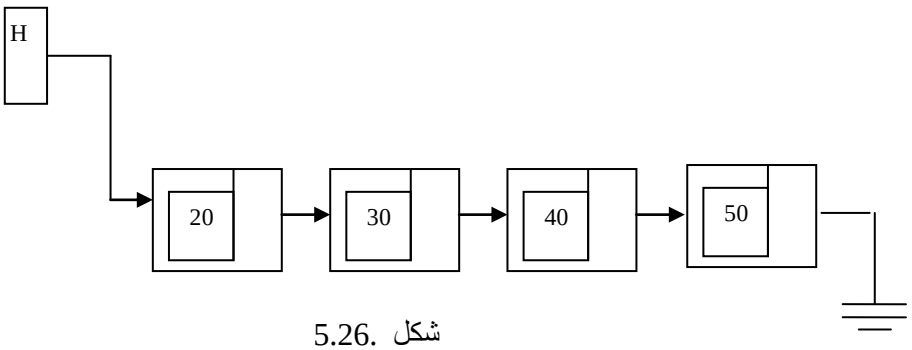
شكل .5.25

1. **جعل المؤشر p يشير الى عنوان اول node في القائمة المتصلة و يتم ذالك عن طريق الامر التالي**

```
p=H;
```

2. ثم يتم فصل ال **node التى في بداية القائمة المتصلة** شكل .5.26 **و يتم ذالك عن طريق الاوامر التالية**

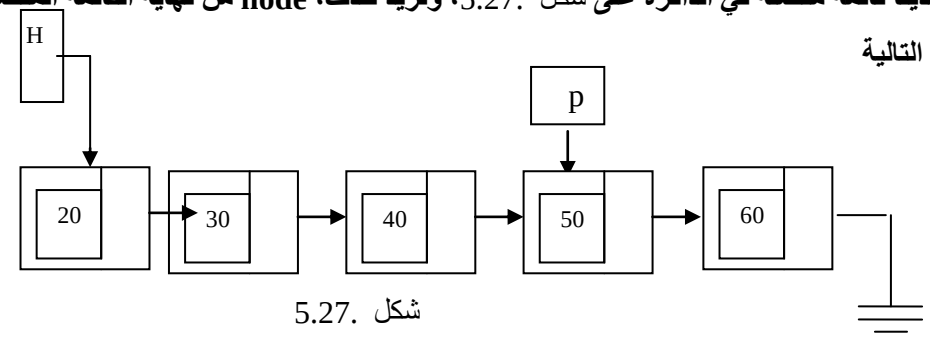
```
if (p!= NULL)
{
 H=p->link;
 p->link =NULL;
 delete(p);
}
```



شكل .5.26

د- حذف node من نهاية القائمة

فلو كان لدينا قائمة متصلة في الذاكرة على شكل .5.27، **ونريد حذف، node من نهاية القائمة المتصلة فيتم عمل الخطوات التالية**



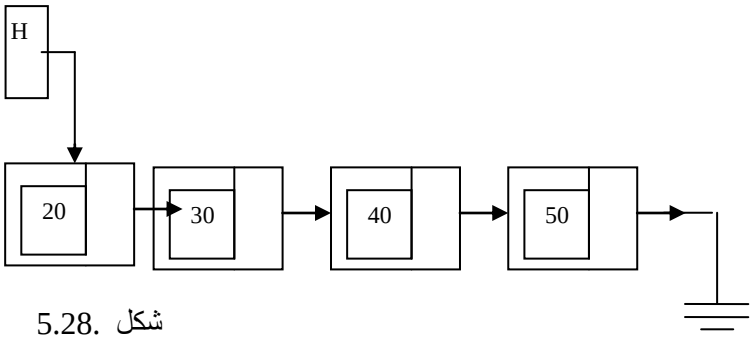
شكل .5.27

1. **جعل المؤشر p يشير الى عنوان اول node في القائمة المتصلة**
2. **تمرير المؤشر p الى node التي قبل نهاية القائمة المتصلة. و يتم ذالك عن طريق الاوامر التالية**

```
p=H;
while (p->link! Null)
  p = p -> link;
```

3. يتم فصل ال **node التى في القائمة المتصلة** شكل .5.28 **و يتم ذالك عن طريق الاوامر التالية**

```
if (p!= NULL)
{
q=p->link;
p->link= NULL;
delete(q);
}
```



شكل .5.28

التعامل مع القوائم المتصلة المتغيرة عن طريق الدوال

عند التعامل مع القوائم المتصلة المتغيرة، يفضل أن نتعامل مع الدوال لإضافة، أو لحذف nodes، و على ذالك يمكن أن تضاف الدوال الخاصة بالإضافة والحذف، فيتم كتابة بعض الدوال الخاصة بالإضافة والحذف ....لخ، ثم بعد ذالك يتم كتابة البرنامج الذي يستدعي تلك الدوال، والبرنامج التالي يوضح ذالك.

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  }*h;
void conslist( );
void printlist(node *);
node *getn( );
node *search(node *,int);
node *searchp(node *p, node *q);
void adda(node *p);
void addb(node *h);
void deln(node *p);
int count(node *p);
main( )
{
 int s;
  h=NULL;
  do{
          cout<<endl<<"----------------------------"<<endl;
          cout<<" 1 : construct linked list 20 nodes "<<endl;
          cout<<" 2 : add node after known value "<<endl;
          cout<<" 3 : add node before known value "<<endl;
          cout<<" 4 : delet known value  linked list "<<endl;
          cout<<" 5 : print the contents of the list"<<endl;
          cout<<" 6 : count no "<<endl;
          cout<<" 7 : exit "<<endl;
          cout<<" enter your selection "<<endl;
          cout<<endl<<"----------------------------"<<endl;  cin>>s;
           switch(s)
           {
            case 1 :  conslist( ); break;
            case 2 :  adda(h);  break;
            case 3 :  addb(h);  break;
            case 4 :  deln(h);   break;
            case 5 :  printlist(h);  break;
            case 6 :  cout<<endl<<" The no. of  "<<count(h)<<endl; break;
            case 7 : cout<<endl<<" by by ";  exit(0);
            default  : cout<<" out of range "<<endl;  break;
            }
```

توصيف محتويات ال nodes المستخدمة في القائمة المتصلة

تصريح عن الدوال المستخدمات في القائمة المتصلة

```
  } while(s!=7); cout<<endl<<" by by ";
 return 0 ;
}
 //-----------------------------------
          void addb(node *r)
          {
          node *p,*q,*t;      int i;
          if(r==NULL)
                  {
                  cout<<endl<<" pardon "<<endl;   return ;
                  }
          cout<<endl<<" enter int no to be added before "<<endl;   cin>>i;
          p=search(r,i);
          if(p==NULL)
                  cout<<endl<<" pardon "<<endl;
         else
         {
          if(p==h)
          {
               q= getn();
           q->next=h;  h=q;
          }
          else
          {
          q= getn();  p=searchp(r,p);  t=p->next; p->next=q;  q->next=t;
          }
        }
return ;
}
int count(node *p) //-----------------------------------
     {
         int i=0;
         while (p!=NULL)
          {
          p=p->next;  i++;  }
 return i;}
 //-----------------------------------------------
void conslist( )
 {
         int i;
         node *p,*q;
 p=new node;   p->next= NULL;
 p->data=rand()%1000;  h=p;
 for(i=1;i<20;i++)
 {
         q= getn();  p->next=q;   p=q;  q->next=NULL;
 }
 p=h;  return ;
}
node *getn()
```

هذه الدالة تقوم بإضافة node الى قبل node معروفة القيمة في القائمة المتصلة

هذه الدالة تقوم بعد ال nodes الموجودة بالقائمة المتصلة

هذه الدالة تقوم بتكوين قائمة متصلة مكونة من 20 nodes، الاضافة من النهاية

هذه الدالة تقوم بإيجاد موقع ل node ثم تولد بيانات وتضعها في محتويات ال node التكوين القائمة المتصلة

```
 {
   node *q ;
q= new node;
q->data =rand()%1000; q->next=NULL;  return q;
}
 //-----------------------------------------------
          void printlist(node *p)
          {
           int i=0;
           if(p==NULL)
           {
                  cout<<" empty list "<<endl;   return ;
           }
           while (p!=NULL) {
                                cout<<" ["<<setw(3)<<p->data<<" ]";
                                 if((i+1)%10==0)
                                  cout<<endl;
                                p=p->next;   i++; }
           return ;
           }
 //-----------------------------------------------
          node *search(node *p, int x)
          {
          if(p==NULL)
           {
                  cout<<" empty list "<<endl;
                  return NULL;
                  }
          while (p!=NULL&&p->data!=x)
                    p=p->next;
                  return p;
           }
           //-----------------------------------------------
           node *searchp(node *p, node *q)
          {
          if(p==NULL)
                  return NULL;
          while (p!=NULL&&p->next!=q)
                    p=p->next;
           if(p!=NULL)
                  return p;
           else return NULL;
           }
           //-----------------------------------------------
          void adda(node *h)
          {
           node *p,*q,*t;
           int i;
           if(h==NULL)
                  {
```

هذه الدالة تقوم بطباعة محتويات ال nodes الموجودة بالقائمة المتصلة

هذه الدالة تقوم بعملية البحث عن قيمة معينة في ال nodes الموجودة بالقائمة المتصلة

هذه الدالة تقوم بعملية البحث عن node معينة بين ال nodes الموجودة بالقائمة المتصلة

هذه الدالة تقوم بإضافة node الى بعد node معروفة القيمة في القائمة المتصلة

```
            cout<<endl<<" pardon "<<endl; return ;
            }
 cout<<endl<<" enter int no to be after "<<endl;   cin>>i;
 p=search(h,i);
 if(p==NULL)
            cout<<endl<<" pardon "<<endl;
t=p->next;  q= getn();  p->next=q;  q->next=t;   return ;
}
//-----------------------------------------------
void deln(node *t)
{
int x ;
 node *p,*q;
 p=t;
if(p==NULL)
 {                                                    هذه الدالة تقوم بحذف node معروفة
        cout<<" empty list "<<endl;  return ;          القيمة من القائمة المتصلة
  }
        cout<<endl<<" enter int no to be deleted "<<endl;
 cin>>x;
while (p!=NULL&&p->data!=x)
          {q=p;
           p=p->next;
           }
if(p!=NULL)
    {
        if(p==h)
          h=h->next;
        else
          q->next=p->next;
        p->next=NULL;  free(p); cout<<" deleted "<<endl;
     }
        else  cout<<" not found "<<endl;
return ;
}
```

5 . 3 بناء هيكل لقائمة متصلة على هيئة Stack وبحجم متغير

من الممكن أن يكون بناء هيكل Stack على حجم متغير، مثل القائمة المتصلة، وكما ذكرنا سابقا، فإن stack يعرف على النحو:

A stack is an ordered collection of items into which new items may be inserted and from which items may be deleted at one end, called the top of the stack.

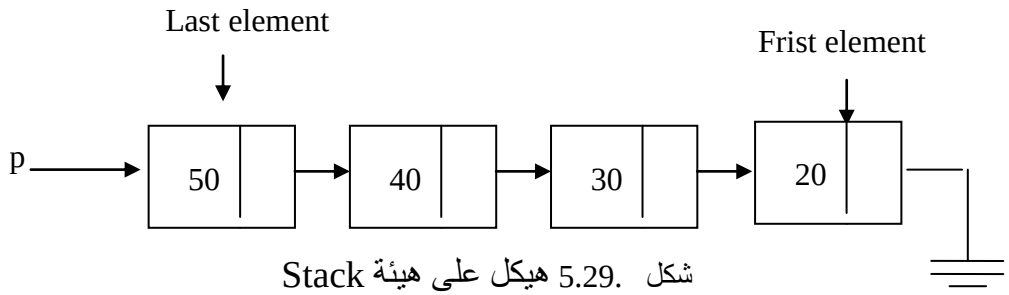
وعليه، فإنه يمكن تشبيه ال stack بأنبوبة طويلة مفتوحة من جهة ومغلقة من الجهة الأخرى.

عملية إدخال المعلومات أو إخراج المعلومات تتم فقط من الجهة المفتوحة للstack . أي أن stack عبارة عن قائمة متصلة تتم فيها اضافة اي node من نهاية القائمة، وعلى ذالك نبدأ بكتابة السجل الذي سوف يحتوي مكونات ال stack شكل .5.29.

```
struct Stack
{
int data;
Stack *next;
} s;
```

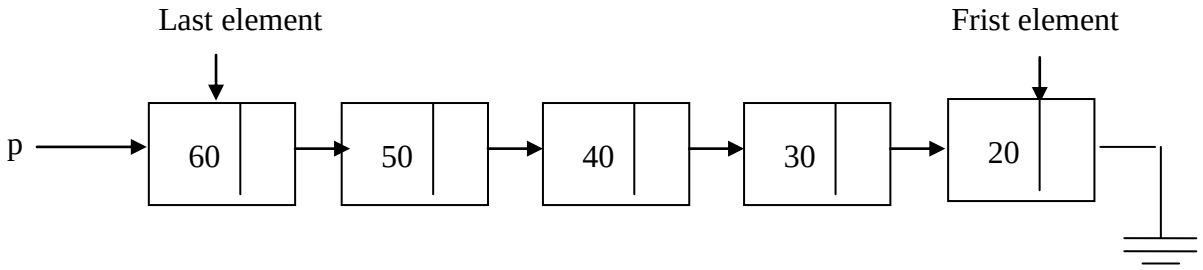
هيكل Stack على هيئة مصفوفة

مؤشر Stack



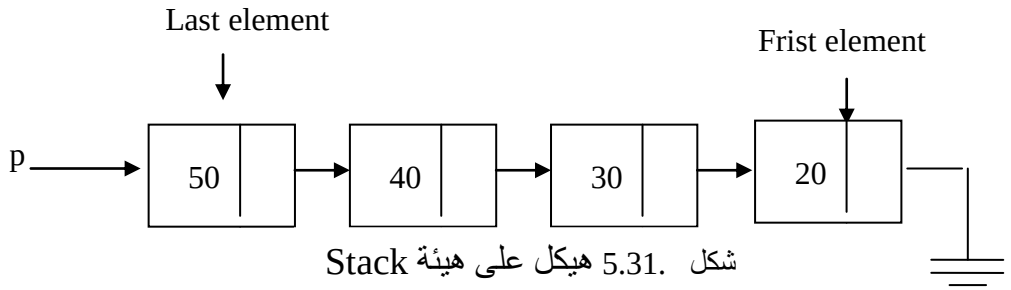
شكل .5.29 هيكل على هيئة Stack

عند إضافة node إلى stack يكون الشكل على النحو التالي شكل .5.30



شكل .5.30 هيكل على هيئة Stack

وعند حذف node من stack يكون الشكل على النحو التالي شكل 5.31.



شكل 5.31. هيكل على هيئة Stack

والبرنامج التالي لتكوين قائمة متصلة مكونة من n nodes و يطبع المخرجات في ملف اسمه "da2.cpp"

```
#include<fstream.h>
#include<iomanip.h>
#include<stdlib.h>
struct Stack
 {
  int data;
  Stack *next;
  }*S;
main()
{
 int i,n;
    fstream Dataf1;
     Dataf1.open("da2.cpp",ios::out);
  S=NULL;
  //-----------------------------------
   Stack *p;
    Dataf1<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
     cout<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
     cin>>n;
   S=new Stack;
   S->next= NULL;
   S->data=rand()%1000;
   Dataf1<<endl<<" during generation"<<endl;
   cout<<endl<<" during generation"<<endl;
   cout<<" ["<<setw(3)<<S->data<<"]";
   Dataf1<<" ["<<setw(3)<<S->data<<"]";
  for(i=1;i<n;i++)
  {
    p=new Stack;
   p->next= S;
   p->data=rand()%1000;
   Dataf1<<" ["<<setw(3)<<p->data<<"]";
   cout<<" ["<<setw(3)<<p->data<<"]";
     if((i+1)%10==0)
    {
            cout<<endl;
     Dataf1<<endl;}
```

```
  S=p;
 }
  p=S;
  cout<<endl<<"_____"<<endl;
  Dataf1<<endl<<"_____"<<endl;
   cout<<endl<<" after generation"<<endl;
   Dataf1<<endl<<" after generation"<<endl;
   if(p==NULL)
     {
            Dataf1<<" empty list "<<endl;
    cout<<" empty list "<<endl;
            return 0;
            }
    i=0;
    while (p!=NULL)
    {
     cout<<" ["<<setw(3)<<p->data<<"]";
    Dataf1<<" ["<<setw(3)<<p->data<<"]";
     if((i+1)%10==0)
    {
            Dataf1<<endl;
     cout<<endl;  }
     i++;
     p=p->next;
     }
    cout<<endl<<" enter any no";
     cin>>i;
    Dataf1 <<endl<<" enter any no";
    Dataf1.close();
     return 0;
    }
```

عند تنفيذ هذا البرنامج سوف تظهر النتائج التالية

enter int the number of nodes to be constructed as stack

during generation

[346] [130] [982] [90] [656] [117] [595] [415] [948] [126]

[4] [558] [571] [879] [492] [360] [412] [721] [463] [47]

[119] [441] [190] [985] [214] [509] [252] [571] [779] [816]

_____

after generation

[816] [779] [571] [252] [509] [214] [985] [190] [441] [119]

[47] [463] [721] [412] [360] [492] [879] [571] [558] [4]

[126] [948] [415] [595] [117] [656] [90] [982] [130] [346]

enter any no

5.3.1 كيفية إدخال أو اخراج عنصر إلى أو من Stack

ذكرنا أن Stack ممكن تخيله أنبوبة طويلة ولذلك لا بد من عملية ما، تتواكب مع هذا التعريف و لكي نضع فيه عنصراً ما. أي أن العملية push هي الجديرة بوضع عنصرا ما داخل stack. كما أن العملية pop هي الجديرة بسحب آخر عنصر من داخل stack.

```
#include<fstream.h>
#include<iomanip.h>
#include<stdlib.h>
struct Stack
 {
  int data;
  Stack *next;
  }*Ps;
    fstream Dataf1;
 void count();
 void push(  int );
 int pop( );
main()
{
 int i,n,x;
Dataf1.open("da2.cpp",ios::out);
   Stack *p;
   Ps=NULL;
     Dataf1<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
      cout<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
      cin>>n;
   Dataf1<<endl<<" during generation"<<endl;
   cout<<endl<<" during generation"<<endl;
  for(i=0;i<n;i++)
  {
    x=rand()%1000;
   Dataf1<<" ["<<setw(3)<<x<<"]";
   cout<<" ["<<setw(3)<<x<<"] ";
    push(x);
      if((i+1)%10==0)
     {
            cout<<endl;
      Dataf1<<endl;}
     }
   p=Ps;
   cout<<endl<<"_____"<<endl;
   Dataf1<<endl<<"_____"<<endl;
    cout<<endl<<" after generation"<<endl;
    Dataf1<<endl<<" after generation"<<endl;
    if (p == NULL)
      {
            Dataf1<<" empty stack"<<endl;
     cout<<" empty stack "<<endl;
```

```
            return 0;
            }
    count( );
    cout<<endl<<" enter whow many nodes to be deleted  "<<endl;
    Dataf1 <<endl<<" enter whow many nodes to be deleted  "<<endl;
   cin>>n;
 for(i=0;i<n;i++)
 {
   x=pop( );
  Dataf1<<" ["<<setw(4)<<x<<"]";
  cout<<" ["<<setw(4)<<x<<"]";
     if((i+1)%10==0)
    {
            cout<<endl;
     Dataf1<<endl;}
 }
   count();
  cin>>n;
    Dataf1.close();
     return 0;
    }
   //-----------------------------------------
int pop ( )
{
Stack *p; int I;
if (Ps == NULL)
 { cout<<"empty stack" <<endl;  return -1;}
  p=Ps;
   Ps=Ps->next;
  I= p->data;
   delete(p);
  return I;
}
//-----------------------------------------
void push (  int x)
{
 Stack *p;
  p = new Stack;
  p->data = x;
 if(Ps == NULL)
  {Ps=p;  Ps->next=NULL;  return;}
  p->next=Ps;
   Ps=p;
   return;
  }
  //-----------------------------------------
  void count()
   {
   int i=0;
    Stack *p;
```

```
 p= Ps;
cout<<endl;
 Dataf1<<endl;
 while (p!=NULL)
 {
  cout<<" ["<<setw(3)<<p->data<<"]";
Dataf1<<" ["<<setw(3)<<p->data<<"]";
  if((i+1)%10==0)
  {
         Dataf1<<endl;
          cout<<endl;  }
  i++;
    p=p->next;
  }
  return;
}
```

عند تنفيذ هذا البرنامج سوف تظهر النتائج التالية

```
 enter int the number of nodes to be constructed as stack  45
during generation
 [346] [130] [982] [ 90] [656] [117] [595] [415] [948] [126]
 [  4] [558] [571] [879] [492] [360] [412] [721] [463] [ 47]
 [119] [441] [190] [985] [214] [509] [252] [571] [779] [816]
 [681] [651] [995] [593] [734] [310] [979] [995] [561] [ 92]
 [489] [288] [466] [664] [892] [863] [766] [364] [639] [151]
 [427] [100] [795] [812]
```

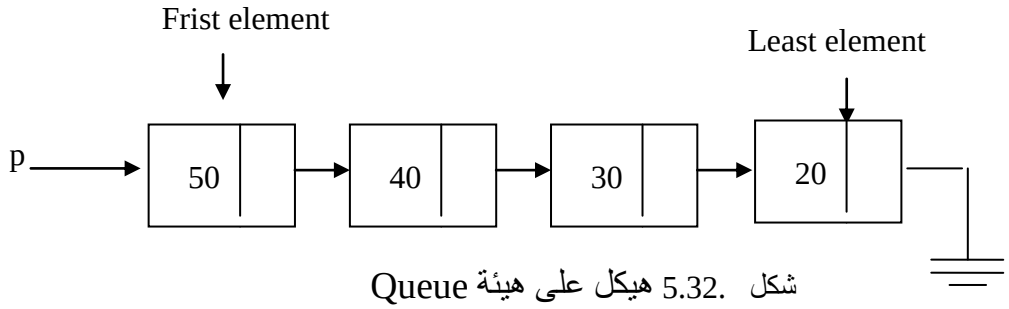
```
 after generation

 [812] [795] [100] [427] [151] [639] [364] [766] [863] [892]
 [664] [466] [288] [489] [ 92] [561] [995] [979] [310] [734]
 [593] [995] [651] [681] [816] [779] [571] [252] [509] [214]
 [985] [190] [441] [119] [ 47] [463] [721] [412] [360] [492]
 [879] [571] [558] [  4] [126] [948] [415] [595] [117] [656]
 [ 90] [982] [130] [346]
 enter whow many nodes to be deleted   5
   [ 812] [ 795] [ 100] [ 427] [ 151]
 [639] [364] [766] [863] [892] [664] [466] [288] [489] [ 92]
 [561] [995] [979] [310] [734] [593] [995] [651] [681] [816]
 [779] [571] [252] [509] [214] [985] [190] [441] [119] [ 47]
 [463] [721] [412] [360] [492] [879] [571] [558] [  4] [126]
              [948] [415] [595] [117] [656] [ 90] [982] [130] [346]
```

5 . 3. 2 الطابور بإستخدام القوائم المتصلة Queue

كما سبق وأن عرفنا الطابور بأنه نوع من أنواع هياكل البيانات المغلفة، ممكن أن نتخيل أن Queue الطابور عبارة عن أنبوبة مفتوحة الطرفين، شكل .5.32،

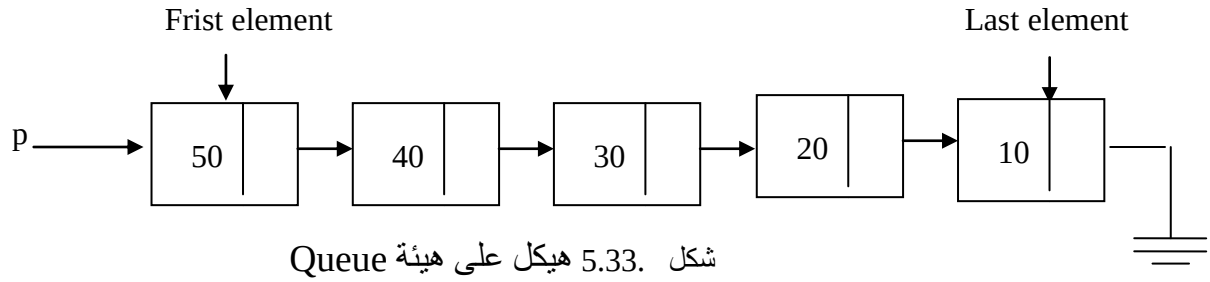
عملية إدخال المعلومات إلى الطابور تتم من الخلف وعملية إخراج المعلومات من الطابور تتم من الأمام، أي أننا نحتاج إلى مؤشرين أحدهما يؤشر للأمام ، ومؤشر آخر يؤشر للخلف.

A queue is an ordered collection of items from which items may by deleted at one end (called front) and into which items may by inserted at the other end (called rear)



شكل .5.32 هيكل على هيئة Queue

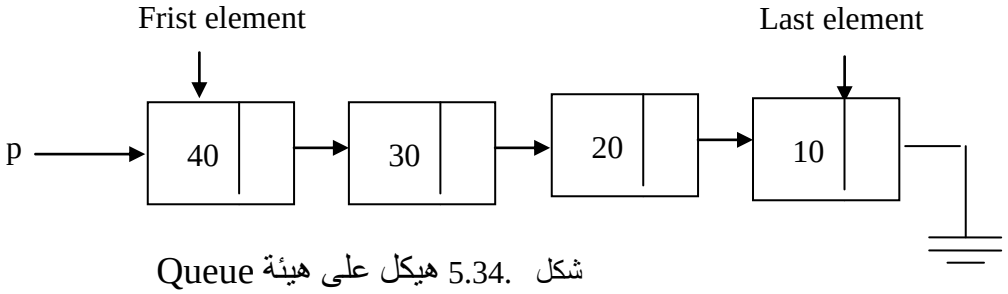
عند اضافة node إلى Queue **يكون الشكل على النحو التالي** شكل .5.33



شكل .5.33 هيكل على هيئة Queue

أي أن مؤشر الخلف انتقل خطوة إلى الخلف لان الإضافة تكون من الاخير

وعند حذف node من Queue **يكون الشكل على النحو التالى** شكل .5.34



شكل .5.34 هيكل على هيئة Queue

أي أن مؤشر الأمام انتقل خطوة إلى الخلف لأن الحذف يكون من الأمام

والبرنامج التالي لتكوين قائمة متصلة مكونة من n nodes و يطبع المخرجات في ملف اسمه "da2.cpp"

```
#include<fstream.h>
#include<iomanip.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  }*S;
main( )
{
 int i,n;
    fstream Dataf1;
     Dataf1.open("da2.cpp",ios::out);
  S=NULL;
  //-----------------------------------
   node *p,*q;
    Dataf1<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
     cout<<endl<<" enter int the number of nodes to be constructed as stack "<<endl;
     cin>>n;
   S=new node;
   S->next= NULL;
   S->data=rand()%1000;
   cout<<endl<<" during generation"<<endl;
   Dataf1<<endl<<" during generation"<<endl;
   cout<<" ["<<setw(3)<<S->data<<"]";
   Dataf1<<" ["<<setw(3)<<S->data<<"]";
   q=S;
  for(i=1;i<n;i++)
  {
    p=new node;
   q->next= p;
   p->data=rand()%1000;
   p->next= NULL;
   Dataf1<<" ["<<setw(3)<<S->data<<"]";
   cout<<" ["<<setw(3)<<p->data<<"]";
     if((i+1)%10==0)
    {
            cout<<endl;
           Dataf1<<endl;}
   q=p;
  }
   p=S;
   cout<<endl<<"_____"<<endl;
   Dataf1<<endl<<"_____"<<endl;
   cout<<endl<<" after generation"<<endl;
   Dataf1<<endl<<" after generation"<<endl;
    if(p==NULL)
```

```
 {
        Dataf1<<" empty list "<<endl;
 cout<<" empty list "<<endl;
        return 0;
        }
 i=0;
 while (p!=NULL)
 {
  cout<<" ["<<setw(3)<<p->data<<"]";
 Dataf1<<" ["<<setw(3)<<p->data<<"]";
  if((i+1)%10==0)
 {
        Dataf1<<endl;
 cout<<endl;  }
 i++;
  p=p->next;
  }
 cout<<endl<<" enter any no";
  cin>>i;
 Dataf1 <<endl<<" enter any no";
 Dataf1.close();
  return 0;
  }
//-------------------------------------
```

وعند تنفيذ هذا البرنامج سوف تظهر النتائج التالية

enter int the number of nodes to be constructed as stack

during generation
[346] [130] [982] [90] [656] [117] [595] [415] [948] [126]
[4] [558] [571] [879] [492] [360] [412] [721] [463] [47]
[119] [441] [190] [985] [214] [509] [252] [571] [779] [816]

---

after generation
[346] [130] [982] [90] [656] [117] [595] [415] [948] [126]
[4] [558] [571] [879] [492] [360] [412] [721] [463] [47]
[119] [441] [190] [985] [214] [509] [252] [571] [779] [816]

enter any no

5 . 3. 2 linked list with Stacks and Queues

كنا قد تحدثنا عن queues, stacks وعرفنا كيفية إستخدامهما وتم كتابة الدوال الخاصة بهما ولكن ممكن أن نبني stacks and queues مستخدمين **linked list**.

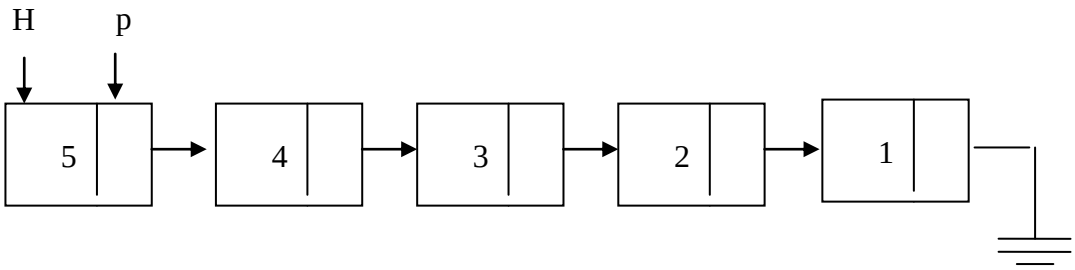
5 . 3. 3 كيفية تكوين stack بإستخدام .linked list

ممكن أن نبني stacks مستخدمين **linked list**.وذالك بجعل (عملية إدخال المعلومات) اضافة node (أو إخراج المعلومات) حذف node تتم فقط من جهة واحدة فقط.

اكتب برنامجا يكوين قائمة متصلة مكونة من 5 nodes على هئة **stack**، شكل 5.35. ثم يضع في كل node رقم تكوينها، ثم يطبع المخرجات .

الحل

```
#include<iostream.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  };
main( )
{
 int i;
node *p,*H;
H= new node;
H->data=1;
H->next=Null;
for(i=2 ;i<6;i++)
 {
   p= new node;
   p->data=i;
   p->next=H;
   H= p;
  }
  while (p! Null)
    {
      cout<< p->data;
      p = p-> next;
    }
return 0;
}
```



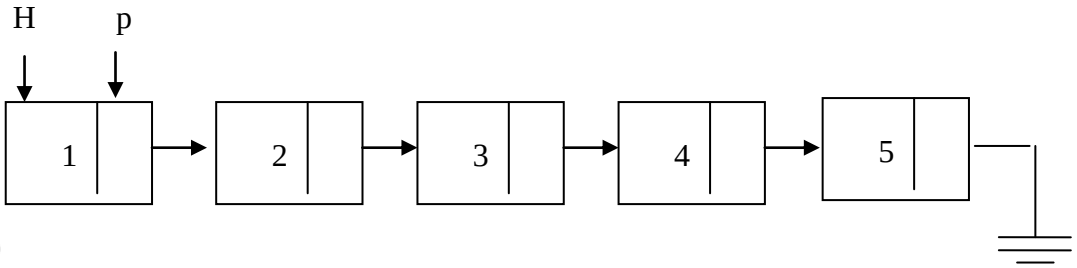
شكل 5.35. هيكل على هيئة **stack**

مخرجات هذا البرنامج سوف تكون على هذا النحو : 5 4 3 2 1

5 . 3. 4 كيفية بناء queue بإستخدام linked lists

ممكن أن نبني queues مستخدمين **linked list**. وذالك بجعل (عملية إدخال المعلومات) اضافة node من جهة الأمام (أو إخراج المعلومات) حذف node تتم من الخلف شكل 5.36.

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
struct node
 {
  int data;
  node *next;
  };
main( )
{
 int i;
node *p,*H, *q;
H= new node;
H->data=1;
q=H;
for(i=2 ;i<6;i++)
 {
   p= new node;
   p->data=i;
   q->next= p;
   q= p;
  }
 p->next= Null;
p=H;
  while (p! Null)
    {
      cout<< p->data;
      p = p-> next;
    }
return 0;
}
```



شكل 5.36 على هيئة **queue**

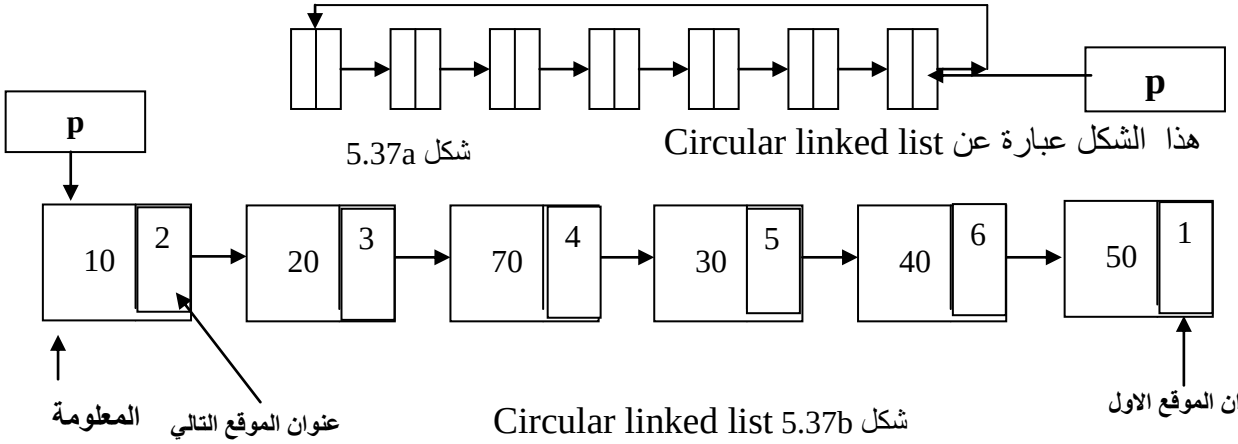
مخرجات هذا البرنامج سوف تكون على هذا النحو:

1 2 3 4 5

5 . 4. Circular Linked list

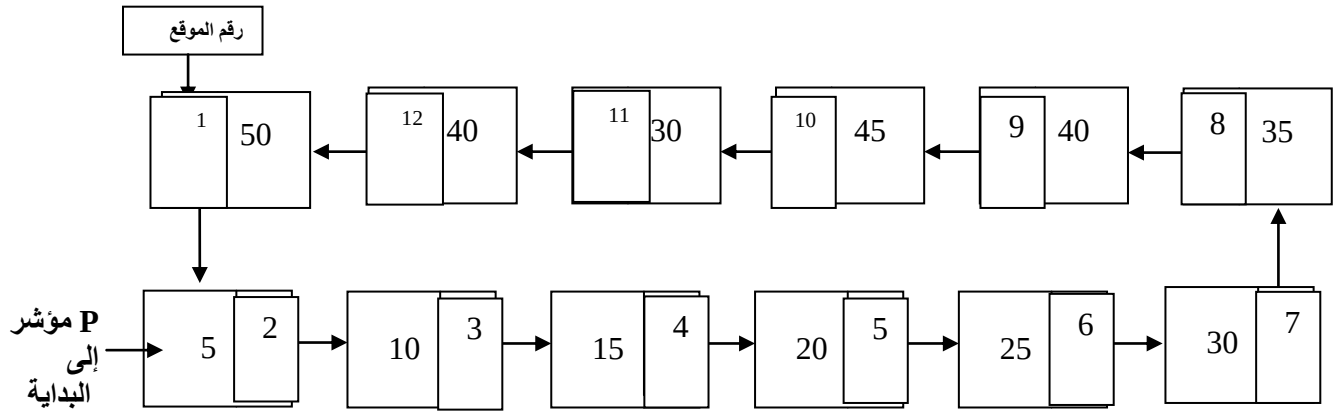
في كثير من التطبيقات نحتاج عادة العودة إلى ال node الأولى، وفي بعض التطبيقات نحتاج عادة العودة إلى ال node السابقة، ولكن في كل القوائم المتصلة السابقة لا نستطيع الرجوع خطوة واحدة، ولكي يتم العودة لا بد من وضع مؤشر في بداية linked list ثم نقدمه خطوة خطوة حتى نصل إلى ال node المطلوبة.

ولكن هناك فكرة أخرى هي Circular linked list، و فيها نجعل ال node الأولى تشير ال node الثانية، و ال node الثانية تشير إلى ال node الثالثة، وهكذا، وال node الاخيرة نجعلها تشير إلى ال node الأولى، و بهذه الطريقة نكون قد كونا Circular linked list ، كما في شكل 5.37:



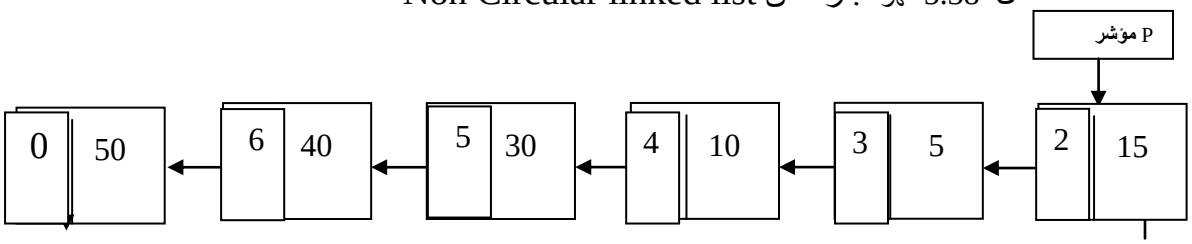
شكل 5.37b Circular linked list

P مؤشر إلى قائمة متصلة على نمط دائري، حيث أن ال node الأولى تشير ال node الثانية، و ال node الثانية تشير إلى ال node الثالثة، وهكذا، وال node الاخيرة تشير إلى ال node الأولى.



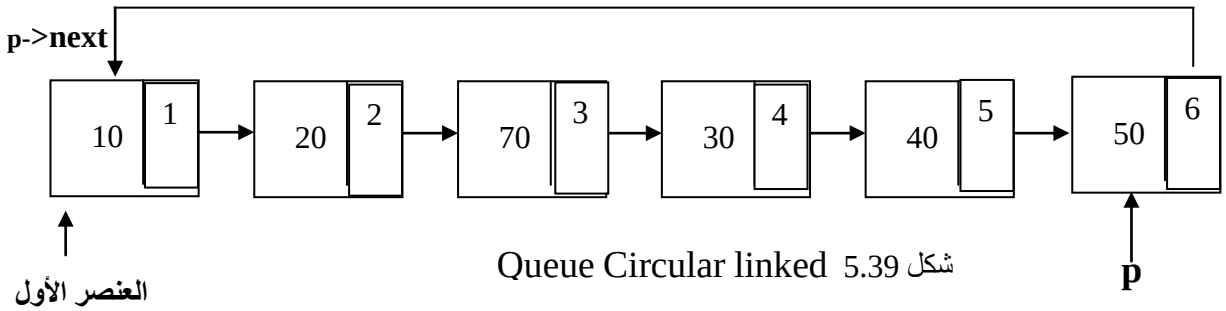
شكل 5.37c Circular linked list

أما الشكل 5.38 فهوعبارة عن Non Circular linked list



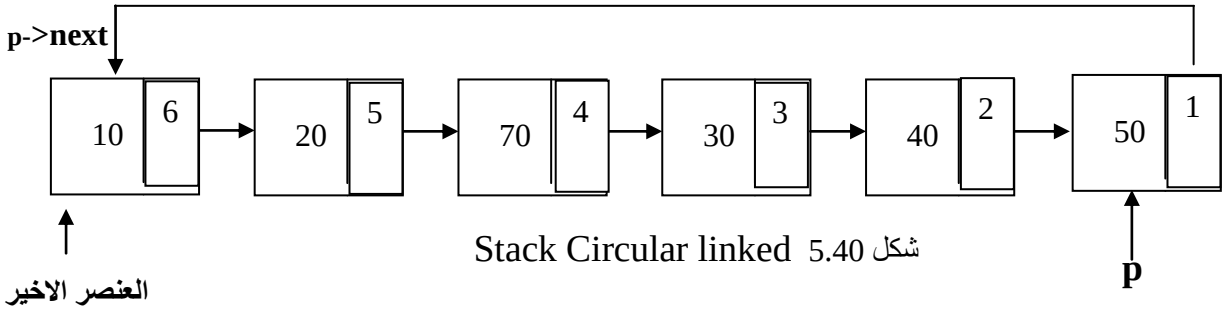
شكل 5.38 Circular linked list

الشكل 5.39 هوعبارة عن Stack Circular linked list **حيث أن p مؤشر إلى** قائمة متصلة على نمط دائري، حيث أن ال node الأولى تشير ال node الثانية، و ال node الثانية تشير إلى الnode الثالثة، وهكذا، وال node الاخيرة لا تشير إلى ال node الأولى بل تشير إلى نهاية القائمة .



شكل 5.39 Queue Circular linked

يمكن تكوين طابور على هيئة ، Circular linked list as a Queue شكل 5.39 ، وكذا يمكن تكوين مكدس على هيئة Circular linked list as a Stack شكل 5.40.



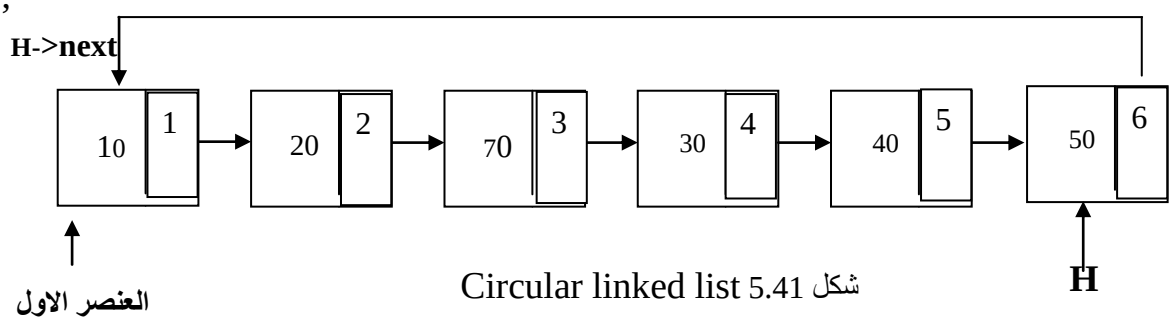
شكل 5.40 Stack Circular linked

```
/* this program to maintain a circular linked list as a queue  or a stack*/
# include<iostream.h>
# include<stdlib.h>
struct Cnode
{
int data;
Cnode *next;
}*H;
```

```
void addq ( int );
void adds ( int );
void del( );
void ptlist( );
main (  )
{
int x, n, i;
H=NULL;
cout<<" Enter the no.-of nodes to be constracted"<<endl;    cin>>n;
cout<<" if you want queue Enter 1 .if you want stack Enter 2 "<<endl;     cin>>i;
 if( i== 1)
 for (i=0; i< n; i++)
 {
  x=rand()%1000;
  addq ( x);
  }
 else
 if( i== 2)
 for (i=0; i< n; i++)
 {
  x=rand()%1000;
  adds ( x);
  }
 else return 0;
ptlist( );
cout<<endl<<" Enter the node to be deleted"<<endl;
cin>>x ;
for (i=0; i< x; i++)
 del( );
 ptlist( );
cin>>x ;
return 0;
}
//-----------------------------------------------
void addq( int no)
{
Cnode *p;
p= new Cnode;
p->data=no;
cout<<no<<" ";
if( H==NULL)
{
 H=p;
```

```
 p->next=p;
  return;
 }
p->next = H->next;
H->next=p;
H=p;
return;
}
```

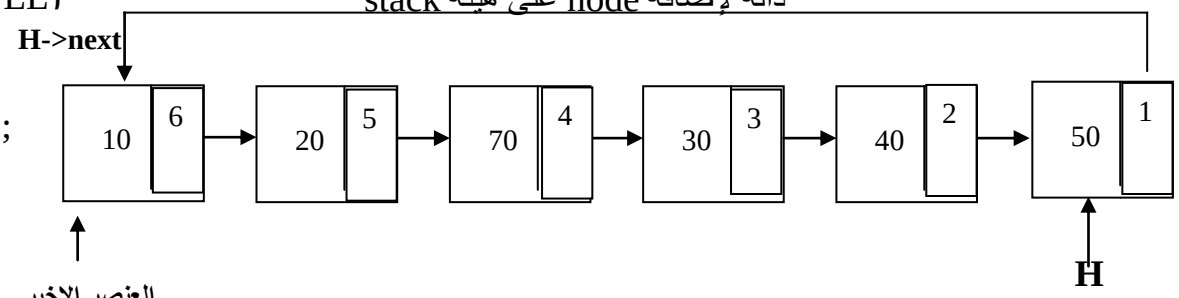
دالة لإضافة node على هيئة طابور



شكل 5.41 Circular linked list

```
//-----------------------------------------------
void adds( int no)
{
Cnode *p;
p= new Cnode;
p->data=no;
cout<<no<<" ";
if( H==NULL)
{
 H=p;
 p->next=p;
  return;
 }
p->next =
H->next=p;
return;
}
```

دالة لإضافة node على هيئة stack



شكل 5.42 Circular linked list

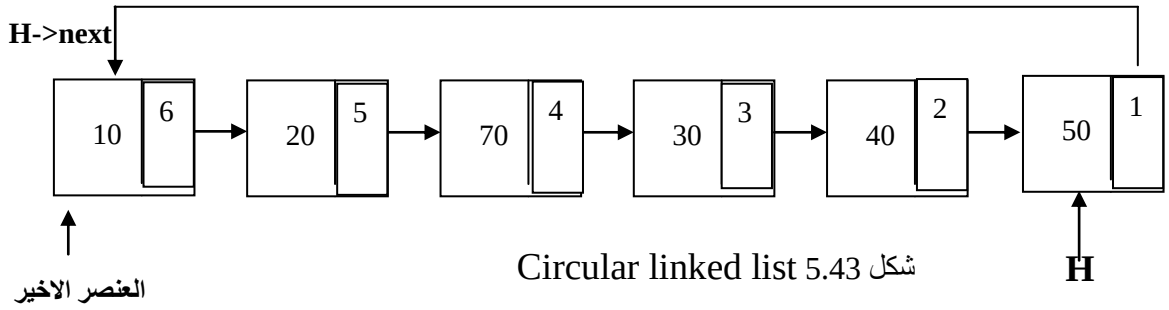
```
//-----------------------------------------------
void del( )
{
Cnode  *p;
if (H== NULL)
  { cout<<endl <<"empty list  ";
    return;
    }
    p=H->next;
  H->next= p->next ;
  p->next=NULL;
     free(p);
     return;
```

دالة لحذف node من الطابور أو stack

```
  }
//-------------------------------------------
 void ptlist( )
 {
  Cnode *p;
  if (H== NULL)
  { cout<<endl <<"empty list   ";
    return;
    }
 p= H->next;
 cout<<endl;
 while (p!= NULL && p != H )
  {
   cout<< p->data<<" ";
   p= p->next;
   }
  if(H!= NULL)
 cout<<H->data<<endl;
 return;
}
```

دالة لطباعة محتوى ال node في الطابور أو stack

Circular linked list as a Stack

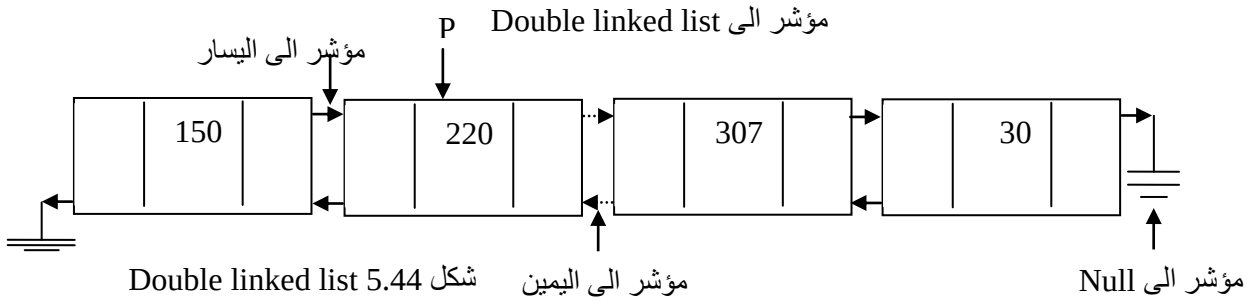


شكل 5.43 Circular linked list

5 . 5. Double Linked list

في بعض التطبيقات نحتاج العودة إلى ال node السابقة أو التي قبلها، ولكن في كل القوائم المتصلة المذكورة سابقا لا نستطيع الرجوع خطوة واحدة إلى الخلف، ولكن في طريقة ال Circular linked list، شكل 5.44 يمكن العودة إلى الخلف والوصول إلى ال node المطلوبة، ولكن لا بد من الوصول إلى نهاية القائمة linked list و من ثم يتم التقدم خطوة خطوة حتى يتم الوصول إلى ال node المطلوبة.

و طبعا هذا مكلف من حيث سرعة الوصول، ولكن هناك فكرة أخرى هي طريقة ال Double linked list، و فيها يتم جعل كل node تحتوي على حقل أو حقول للمعلومات، وكذاك كل node تحتوي على مؤشرين أحدهما يشير إلى اليمين سواء كان هناك node أم لا، و المؤشر الثاني يشير إلى اليسار ايضا سواء كان هناك node أم لا وهكذا، و بهذه الطريقة نكون قد كونا Double linked list، كما في شكل 5.44:



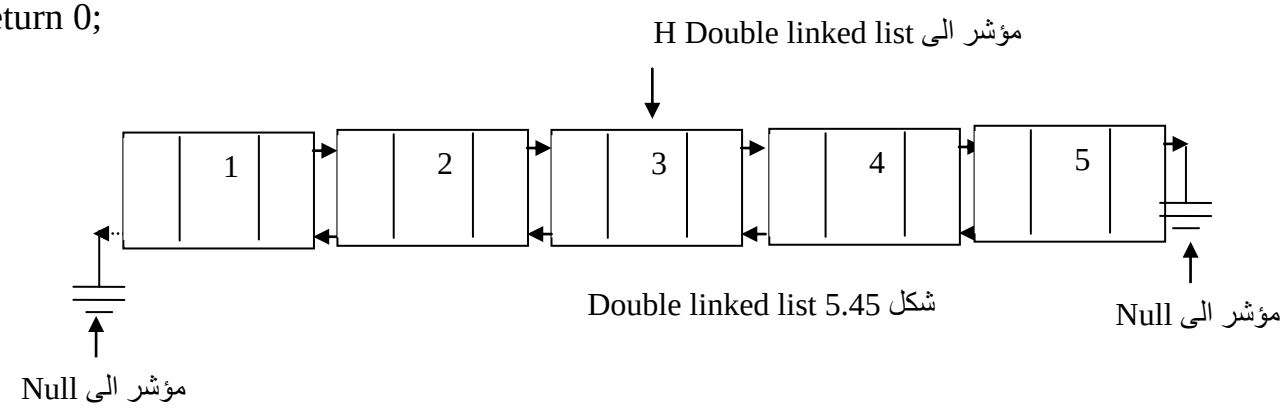
شكل 5.44 Double linked list

في ال Double linked list، أقل node تتكون من 3 حقول، حقل واحد للمعلومة، و حقلين للمؤشرين أحدهما يشير إلى اليمين والآخر يشير إلى اليسار.

و البرنامج التالي يكون 5 node شكل 5.45 ويربطهن على هيئة Double linked list.

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
struct Dnode
 {
  int data;
  Dnode *L,*R;
  };
main( )
{
 int i;
Dnode *p,*H, *q;
H= new Dnode;
```

```
H->data=1;
H->L=NULL;

q=H;
for(i=2 ;i<6;i++)
 {
   p= new Dnode;
   p->data=i;
   p->L= H;
   H->R=p;
   H= p;
  }
 p->R= Null;
  while (p! Null)
    {
      cout<< p->data;
      p = p-> R;
    }
return 0;
}
```



Double linked list 5.45 شكل

5 . 6. الأشجار Trees

ممكن أن تمثل البيانات على هيئة شجرة بحيث يكون هناك جذر ثم فروع ثم أوراق. والأوراق عبارة عن البيانات. وتعتبر الشجرة من هياكل البيانات المرنة والمفيدة في كثير من التطبيقات وخاصة في عمليات البحث، حيث من خلال بناء الأشجار يتم وضع البيانات فيها وفق نمط معين، وعند إذ يمكن الوصول بسرعة إلى المعلومة المراد البحث عنها أو الحكم بأن المعلومة المطلوبة غير موجودة. وسوف ندرس الشجرة الثنائية في هذا المقرر فقط Binary trees

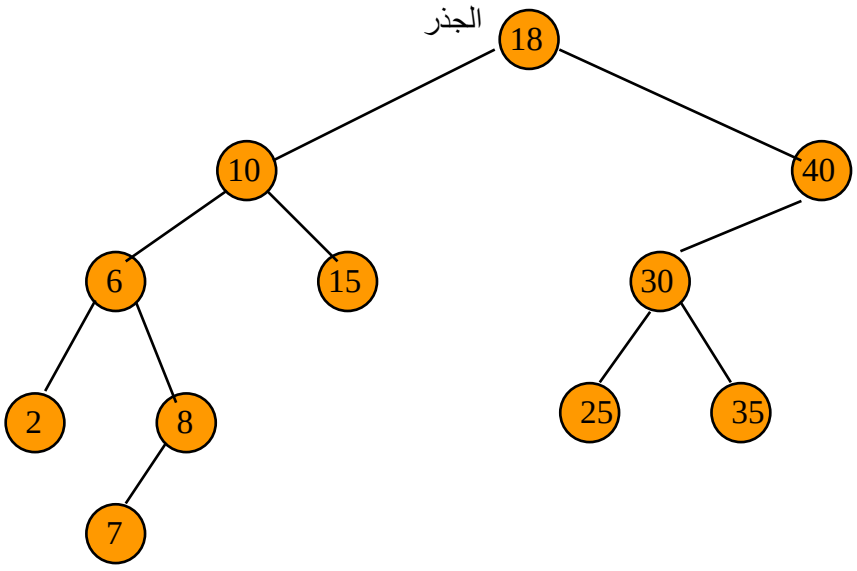
5 . 6. 1 Binary trees

الشجرة الثنائية عبارة عن مجموعة محدودة من ال nodes ، كل node يمكن أن يخرج منها على الأكثر اثنين ابناء، و كل node يمكن أن تجزأ على الأقل إلى ثلاث مجموعات غير متصلة.

المجموعة الأولى تحوي عنصرا واحدا أو أكثر، المجموعتان الآخريان تسميا الإبن اليمن والإبن الأيسر Left & Right ، و أول node تسمى الجذر (جذر الشجرة).

كل العناصر على يمين الجذر قيمهن أكبر من قيمة الجذر، و كل العناصر على يسار الجذر قيمهن اقل من قيمة الجذر.

وكذالك كل العناصر التي تقع على يمين أي عنصر x قيمهن أكبر من قيمة العنصر x، و كل العناصر على يسار العنصر x قيمهن اقل من قيمة العنصر x.



في الشكل أعلى توجد شجرة ثنائية تحتوي على 11 عنصرا، العنصر الأول يحتوي على القيمة 18 ويسمى جذر الشجرة الرئيسي أما العناصر 40, 30, 10, 6, 8 فتسمى جذور فرعية، وأما العناصر 35, 25, 15, 2, 7 أي العناصر التي ليس لها أبناء فتسمى الأوراق.

Traversal of a binary tree 5 . 6. 2

يمكن أن نمر على كل عنصر في الشجرة بثلاث طرق:

أ- preorder وفي هذه الطريقة نبدأ 1) بالجذر 2) الشمال 3) اليمنى.

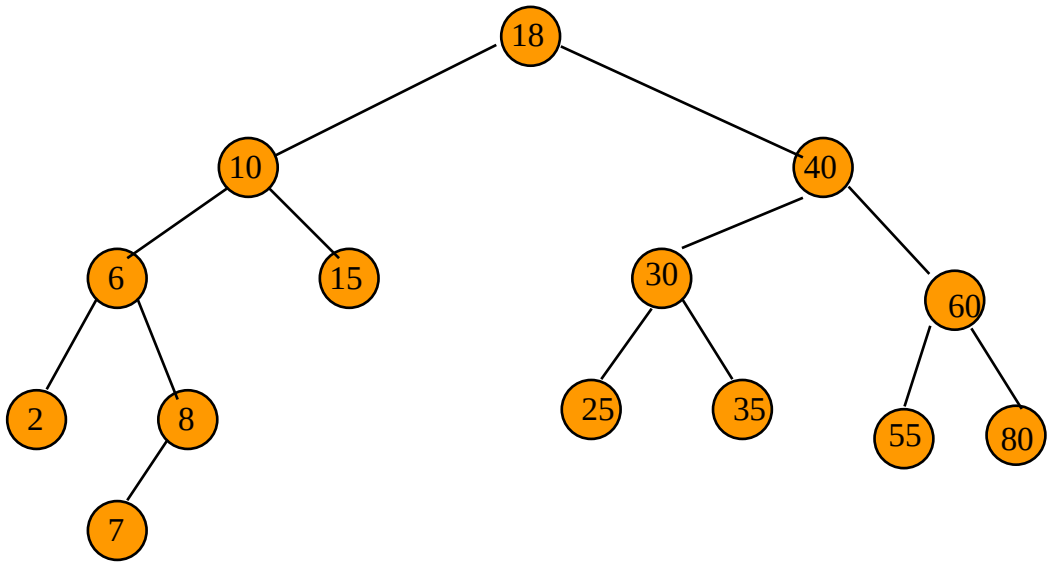
وعندما نصل إلى عنصر فنتعامل معه وكأنه يكون شجر ثنائية جديدة.

ب- post order وفي هذه الطريقة نبدأ 1) الشمال 2) اليمنى 3) الجذر.

ج- in order وفي هذه الطريقة نبدأ 1) الشمال 2) الجذر 3) اليمنى.

وعلى هذه الطرق الثلاث يمكن أن تتم عملية المرور على كل عناصر الشجرة السابقة على النحو:

70	60	50	56	65	67	80	76	90	85	95	Preorder
50	65	67	66	60	70	76	85	95	90	80	Post order
50	60	65	66	67	70	76	80	85	90	95	In order




```
/* program to implement a binary trem*/
#include<iostream.h>//
#include<iomanip.h>//
#include<stdlib.h>  //  :
struct Tree
 {
  int data;
  Tree *R,*L;
  }*Root;
void maktree(Tree *p);
Tree *getnt( );
Tree *search(Tree *,int);
Tree *search(Tree *p, Tree *q);
```

```
void addnt(Tree *p,Tree *q);
void prtin(Tree *p);
void prtps(Tree *p);
void prtpr(Tree *p);
main()
{
 int n,i;
  Root=NULL;
            n=rand()%100;
            for(i=0;i<n;i++)
                    maktree(Root);
            cout<<endl<<"----------------------------"<<endl;
            prtin( Root);
            cout<<endl<<"----------------------------"<<endl;
            prtps( Root);
            cout<<endl<<"----------------------------"<<endl;
            prtpr( Root);
            cin>>i;
            return 0 ;
            }
  //-----------------------------------
            void maktree(Tree *p)
            {
            Tree *t,*q;
            q=getnt();
                    cout<<q->data<<"----";
                    if(p==NULL)
                     Root=q;
                    else
                    {
                     t=search(p,q);
                     addnt(t,q);
                     }
                     return ;
                     }
  //-----------------------------------
           Tree *getnt( )
           {
            Tree *p;
            p=new Tree;
            p->R= NULL;
            p->L= NULL;
            p->data=rand()%1000;
            return  p;
           }
  //-------------------------------------------------
           void addnt(Tree *p,Tree *q)
  {
            if(p->data>q->data)
            p->L=q;
```

```
                else
                 p->R=q;
                 return;
        }
//-------------------------------------------------
        void prtin(Tree *p)
        {
         if(p==NULL)
                return ;
                prtin(p->L);
                cout<<p->data<<" ";
                prtin(p->R);
                }
        void prtps(Tree *p)
        {
         if(p==NULL)
                return ;
                cout<<p->data<<" ";
                prtps(p->L);
                prtps(p->R);
                }
        void prtpr(Tree *p)
        {
         if(p==NULL)
                return ;
                prtpr(p->L);
                prtpr(p->R);
                cout<<p->data<<" ";
                }
//-------------------------------------------------
        Tree *search(Tree *p, Tree *q)
        {
        Tree *f,*s;
        f=p;
         while (f!=NULL)
                {
                 s=f;
                 if(q->data>f->data)
                  f=f->R;
                 else
                        f=f->L;
                 }
                return s;
                }
```

س: اكتب المخرجات عند المرور على كل عنصر في هذه الشجرة مستخدما الثلاث الطرق المذكورة أعلى

/* traverse a binary search tree in a LDR &achion*/

ويمكن كتابة preorder , post order على نفس الخط.

الخلاصة

1. **يمكن تصميم مجموعات مختلفة من هياكل البيانات حسب الطلب، مثل السجلات والقوائم المتصلة و الطوابير والمكادس والاشجار.**
2. **يمكن جعل الهياكل تحتوي على مجموعات مختلفة من البيانات**
3. **حجم هذه الهياكل يحدد أثناء تنفيذ البرنامج**

والله الموفق والهادي إلى سواء السبيل ،،،

اسئلة عامة على مقرر هياكل البيانات

1)

أ) اكتب برنامجاً يولد 150 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية البعد، ثم يطبع هذه الأعداد كل عشرة أعداد في صف، ثم يطبع قيمة وموقع ثاني أصغر عدد في المجموعة .

ب) اكتب برنامجاً يولد 50 عددا من نوع int ثم يضع هذه الأعداد في مصفوفة ثنائيه (مكونة من5 صفوف و10 أعمده) ثم يطبع قيمة وموقع أصغر هذه الأعداد في المجموعة .

ت) اكتب برنامجاً يولد 50 عددا عشوائيا من نوع int ثم يضع هذه الأعداد في مصفوفة أحادية ، ثم يطبع هذه الأعداد كل10 أعداد في صف، ثم يقوم بترتيب هذه العناصر تصاعدياً ثم يطبع هذه الأعداد كل 5 أعداد في صف.

ث) اكتب برنامجاً يعرف مصفوفة مكونة من 7 صفوف و5 أعمدة، ثم يولد قيمة قيم عشوائية في كل صف ما عدى الصف الأخير والعمود الأخير يضع فيهم أصفارا، ثم يقوم البرنامج بوضع المجموع لكل صف في العمود الأخير والمجموع لكل عمود في الصف الأخير ثم يطبع هذه الأعداد كل صف في صفه.

2) بإستخدام الدوال: اكتب برنامجاً يكون struct (سجل طالب) يعرف فيه رقم الطالب ، جنس الطالب ، المواد التي أخذها(عددهن 6 مواد)، المعدل، ثم يعرف هيكل مكون من 100 سجل طالب، يقوم البرنامج بتوليد قيم عشوائية مناسبة لتلك المواد و يحسب المعدل لكل طالب، ثم يطبع رقم الطالب الحاصل على أكبر معدل .

3) بإستخدام الدوال: أكتب برنامجاً يكون linked list تحتوي على 5 nodes كل node تحتوي على عنصر من نوع int و عنوان، ثم استدعى الدالة insertb لإضافة node إلى بداية القائمة (linked list) ، ثم استدعى الدالة inserta لإضافة عنصرا إلى نهاية القائمة (linked list) ، ثم استدعى الدالة remove لحذف عنصرا من القائمة (linked list)، ثم استدعى الدالة prt، لطباعة محتويات القائمة (linked list) ثم استدعى الدالة count لطباعة عدد ال nodes في ال (linked list)

4) ا- بإستخدام الدوال: اكتب برنامجاً يكون linked list على هيئة طابورا (queue) تحتوي على 5 nodes ثم يستدعي الدالة Insert لإضافة 5 nodes إلى الطابور، ثم يستدعي الدالة Prite لطباعة محتويات الطابور، ثم يستدعي الدالة (remove) لحذف node من الطابور، ثم يستدعي الدالة Prite لطباعة محتويات الطابور بعد الحذف.

ب- بإستخدام الدوال :اكتب برنامجاً يكون stack ثم يقوم باستدعاء الدالة push لإضافة 5 nodes إلى ال stack ثم يستدعي الدالة pop لحذف عنصراً من داخل ال stack و كذالك يستدعي الدالة prt لطباعة محتويات stuck بعد كل pop و بعد كل push .

5) بإستخدام الدوال: أكتب برنامجاً يكون شجرة ثنائية مكونة من 90 عنصرا ثم يستدعى الدالة search للبحث عن العنصر x في الشجرة ثم تقوم الدالة بطباعة الرسالة المناسبة.

بسم الله الرحمن الرحيم

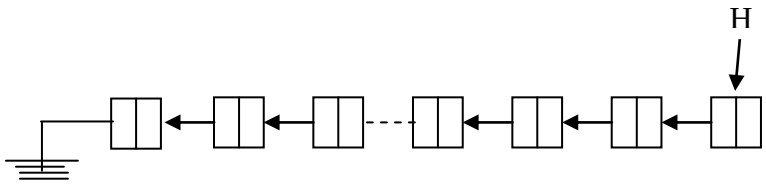
جامعة الأندلس
كلية الهندسة و تقنية المعلومات
أجب عن كل الأسئلة الآتية :

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
4	**8**	**12**	**16**	**20**	**24**	**28**	**32**	**36**	**40**
8	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**	**80**
16	**32**	**48**	**64**	**80**	**96**	**112**	**128**	**144**	**160**
32	**64**	**96**	**128**	**160**	**192**	**224**	**256**	**288**	**320**

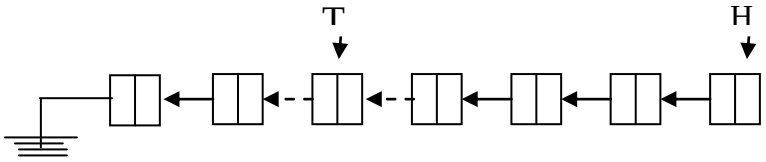
س$_1$ ا- كتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية كما هو موضح، ثم يطبع محتوياتها عناصر كل صف في سطر . (15 درجة)

س$_1$ ب- اكتب برنامجا يعرف مصفوفة أحادية مكونة من 200 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، قيمة كل رقم لا تزيد عن 999 ولا تقل عن 50 ثم يطبع محتويات المصفوفة (كل عشرة أعداد في صف)، ثم يطبع قيمة وموقع اكبر عناصر المصفوفة. (10 درجة)

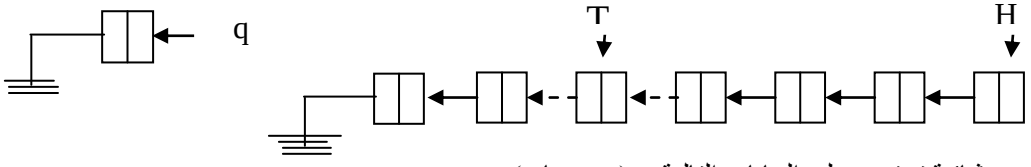
س$_2$ ا- اكتب برنامجا يكون قائمة متصلة Linked list على هيئة stack ومكونة من 100 nodes ثم يولد فيهن أرقاما عشوائية ثم يطبع محتويات ال stack (10 درجة)



س$_2$ ب- ا- اشرح (موضحا بالرسم وعبارات C++) كيف يمكن حذف ال node المشار إليها بالمؤشر T من القائمة المتصلة Linked list التي يشير إليها المؤشر H (5 درجات)



س$_2$ ج- - اشرح (موضحا بالرسم وعبارات C++) كيف يمكن إضافة ال node المشار إليها بالمؤشر q إلى قبل ال node المشار إليها بالمؤشر T في القائمة المتصلة Linked list التي يشير إليها المؤشر H (6 درجات)



س$_2$ د- ارسم شجرة ثنائية تحتوي على البيانات التالية: (4 درجات)

200 240 300 280 120 180 290 170 140 320 390 110 101 430 415 260 70 90 310 32

س$_3$ - ا **اكتب دالة تعد nodes الموجودة في قائمة متصلة ذات اتجاهين Doable Linked list ،مشار إليها بالمؤشر T. (10 p)**

س$_3$ ب **اكتب دالة تضيف node إلى شجرة ثنائية** يشار إليها بالمؤشر R .**(15 درجة)**

مع تمنياتي لكم بالتوفيق والنجاح

جامعة الأندلس
كلية الهندسة و تقنية المعلومات
أجب عن الأسئلة الآتية :

بسم الله الرحمن الرحيم

الزمن : 120 دقيقة التاريخ :
المادة : هياكل بيانات : المستوى الثاني
مدرس المادة : د/ صالح العسلي.
الاختبار النهائي :100%:

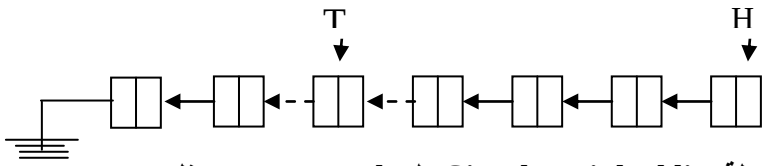
س1 ا اكتب التعريف اللازم بلغة C++ وذالك لتوصيف سجل لطالب بياناته كالتالي: اسم الطالب ، رقم الطالب، جنس الطالب، درجات المقررات الدراسية (5 مقررات)، المعدل العام، حالة الطالب، ثم كون هيكل للطلاب يتكون من 100 سجل. (5 درجات)

س1 ب اكتب بلغة C++ برنامجا يكون مثل المصفوفة التالية ثم يطبع محتويات المصفوفة عناصر كل صف في سطر (15)

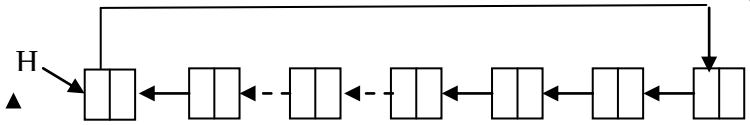
5	7	9	11	13	15	17	19	22
5	7	9	11	13	15	17	19	22
5	7	9	11	13	15	17	19	22
5	7	9	11	13	15	17	19	22

س-2 اكتب برنامجا يعرف مصفوفة أحادية مكونة من 200 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة بعكس الترتيب (كل 7 أعداد في صف)، ثم يطبع في سطر اخر قيمة وموقع اصغر عنصر . (20 درجة)

س3 اكتب برنامجا يكون قائمة متصلة Linked list تحتوى على 100 nodes علما بان كل node تتكون من حقلين هما info, *next، ثم يجعل المؤشر H يشير إلى بدايتها ثم يقوم البرنامج بحذف ال node المشار إليها بالمؤشر T (p 20)



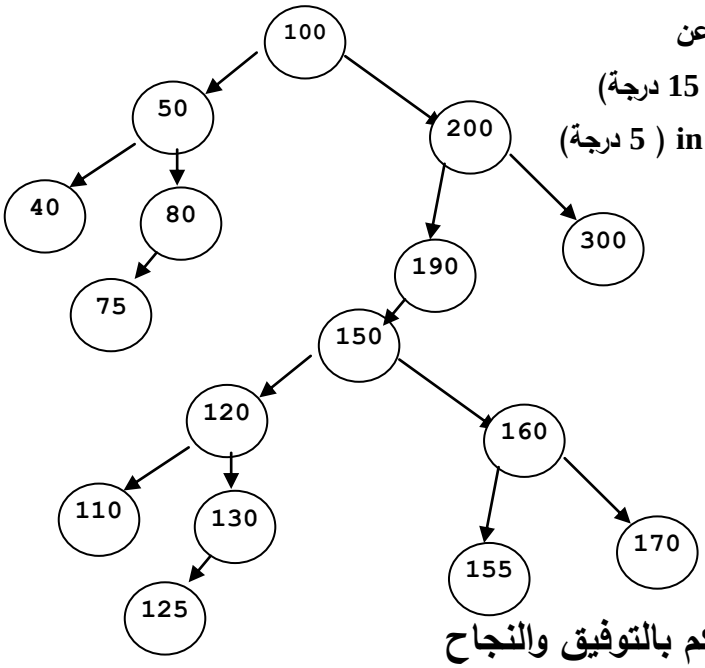
س—13– اكتب دالة تضيف node إلى قائمة متصلة Circular Linked list كل node تتكون من حقلين no, next (كما في الشكل التالي) . (10 درجات)



س- 3–2 اكتب دالة تحذف node من قائمة متصلة Circular Linked list (على هيئة طابور) . (10 درجات)

س4–ا p مؤشرا إلى جذر شجرة ثنائية، اكتب دالة تبحث عن العنصر x في شجرة ثنائية ثم تطبع الرسالة المناسبة. (15 درجة)

س4–ب–اكتب محتويات الشجرة الثنائية بطريقة in order (5 درجة)



تمنياتي لكم بالتوفيق والنجاح

الجمهورية اليمنية
جامعة الأندلس
كلية الهندسة و تقنية المعلومات
اجب عن كل الأسئلة الآتية

الزمن : 90 دقيقة: التاريخ :
المادة : هياكل بيانات : الاختبار النصفي :25%
مدرس المادة : د/ صالح العسلي.
اسم الطالب:

س1 ا- أكمل الفراغ التالي: (2 درجة)

يعتبر ال struct من هياكل البيانات، ومن مميزاته ----------

ب- اكتب التعريف اللازم بلغة C++ وذالك لتوصيف الهيكل التالي: (5 درجات)

Name	No	Six	Salary	Taxe	Net-salary
الاسم	الرقم	الجنس	المرتب	الضريبة	صافي المرتب

ج اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية ثم يطبع محتويات المصفوفة عناصر كل صف في سطر. (13 درجة)

3	**6**	**9**	**12**	**15**	**90**	**21**	**24**	**27**
27	**30**	**33**	**36**	**39**	**90**	**45**	**48**	**51**
51	**54**	**57**	**60**	**63**	**90**	**67**	**72**	**75**
75	**78**	**81**	**84**	**87**	**90**	**93**	**96**	**99**
3	**6**	**9**	**12**	**15**	**90**	**21**	**24**	**27**
27	**30**	**33**	**36**	**39**	**90**	**45**	**48**	**51**

س2 - اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 100 صفاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة (كل 8 عناصر في صف) ثم يطبع في صف جديد قيمة وموقع اصغر عناصر المصفوفة. (15 درجة)

س3 - اكتب برنامجا يكون قائمة متصلة Linked list على هيئة طابور queue مكونة من 70 nodes ، كل node تتكون من حقلين no, next ثم يضع في كل node رقما عشوائيا ، ثم يطبع محتويات الطابور ،(كل 7 عناصر في صف) ثم يحذف node من الطابور. (15 درجة)

جامعة الأندلس
كلية الهندسة و تقنية المعلومات
اجب عن كل الأسئلة الآتية

نموذج
1

الزمن : 120 دقيقة: التاريخ : 2011 /2/
المادة : هياكل بيانات : الاختبار النهائي
:60%
مدرس المادة : د/ صالح العسلى.

5	5	5	5	5
5	5	5	5	5
5	5	5	5	5

س1 -ا اكتب التعريف اللازم بلغة C++ وذالك لتوصيف الهيكل التالي
و يضع فيه القيم الموضحة (5 درجات)
س1ب- اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية ثم يطبع محتويات المصفوفة عناصر كل صف في سطر. (15 درجة)

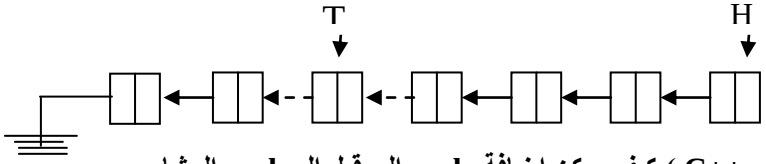
6	9	12	15	90
30	33	36	39	90
54	57	60	63	90
78	81	84	87	90
102	105	108	111	90
126	129	132	135	90

س2 -ا اكتب التعريف اللازم بلغة C++ وذالك لتوصيف الهيكل التالي (5 درجة)

Name	No	Six	Salary	Taxe	Net-salary
الاسم	الرقم	الجنس	المرتب	الضريبة	صـــــافي المرتب

س2 ب اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 120 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة (كل 6 عناصر في صف) ثم يطبع في صف جديد قيمة وموقع اصغر عناصر المصفوفة. (15 درجة)

س3 –ا اشرح (موضحا بالرسم وعبارات C++) كيف يمكن حذف ال node المشار إليها بالمؤشر T من القائمة المتصلة Linked list التي يشير إليها المؤشر H (6 درجة)



س3 –ب اشرح (موضحا بالرسم وعبارات C++) كيف يمكن إضافة node إلى قبل ال node المشار إليها بالمؤشر T في القائمة المتصلة Linked list التي يشير إليها المؤشر H و الموضحة بالشكل السابق (8 درجات)

س3 –ج ارسم شجرة ثنائية تحتوي على البيانات التالية: (6 درجات)

250 240 300 280 120 180 290 170 140 320 390 110 101 430 260 70 90 310 32

س4 -ا اكتب برنامجا يكون قائمة متصلة Linked list على هيئة مكدس stack مكونة من 170 nodes ، كل node تتكون من حقلين no, next ثم يضع في كل node رقما عشوائيا ، ثم يطبع محتويات المكدس ،(كل 7 عناصر في صف). (10)
س4 ب اكتب دالة تضيف node إلى قائمة متصلة ذات اتجاهين Doable Linked list ،مشار إليها بالمؤشر T. (10)
س5 ا اكتب دالة تضيف node إلى قائمة متصلة Circular Linked list، على هيئة Stack مشار إليها بالمؤشر T علما بان كل node تتكون من حقلين هما no, next . (10 درجة)
س5 ب p مؤشرا إلى جذر شجرة ثنائية، اكتب دالة تبحث عن العنصر x في الشجرة الثنائية (x من نوع int) ثم تطبع الرسالة المناسبة. (10 درجة)

وفقكم الله وسدد خطاكم

جامعة صنعاء
كلية الحاسوب وتقنية المعلومات
اجب عن كل الأسئلة الآتية

الزمن : 120 دقيقة: التاريخ : 2011 /3/
المادة : هياكل بيانات : اختبار تكميلي
100%:
مدرس المادة : د/ صالح العسلى.

س1-ا اكتب التعريف اللازم بلغة C++ وذالك لتوصيف الهيكل التالي و يضع فيه القيم الموضحة (5 درجات)

10	10	10	10
10	10	10	10
10	10	10	10

س1ب- اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية ثم يطبع محتويات المصفوفة عناصر كل صف في سطر. (15 درجة)

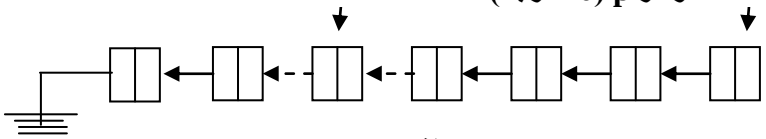
6	9	12	15	10
30	33	36	39	10
54	57	60	63	10
78	81	84	87	10
102	105	108	111	10
126	129	132	135	10

س2-ا اكتب بلغة C++ التعريف اللازم وذالك لتوصيف هيكل يحتوي على البيانات اللازمة لطالب (5 درجة)

Name	No	Six	Subjects	Average	state
الاسم	الرقم	الجنس	المواد	المعدل	الحالة

س2 ب اكتب بلغة C++ برنامجا يعرف مصفوفة أحادية مكونة من 180 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة (كل 5 عناصر في صف) ثم يطبع في صف جديد قيمة وموقع اصغر عناصر المصفوفة. (15 درجة)

س3 –ا اشرح (موضحا بالرسم وعبارات C++) كيف يمكن حذف ال node المشار إليها بالمؤشر q من القائمة المتصلة Linked list التي يشير مؤشر p (10 درجة)



س3 –ب اشرح (موضحا بالرسم وعبارات C++) كيف يمكن إضافة node إلى قبل ال node المشار إليها بالمؤشر q في القائمة المتصلة Linked list التي يشير إليها المؤشر p و الموضحة بالشكل السابق (10 درجات)

س4 -ا اكتب برنامجا يكون قائمة متصلة Linked list على هيئة طابور queue مكونة من 170 nodes ، كل node تتكون من حقلين no, next ثم يضع في كل node رقما عشوائيا ، ثم يطبع محتويات الطابور ،(كل 8 عناصر في صف). (12 درجة)

س4 ب ارسم شجرة ثنائية تحتوي على البيانات التالية: (8 درجات)

255 240 300 280 120 180 290 30 170 140 325 390 110 101 430 415 260 70 90 32 310 . (10 درجة)

س5 -ا اكتب دالة تضيف node إلى قائمة متصلة Circular Linked list، على هيئة طابور queue مشار إليها بالمؤشر T علما بان كل node تتكون من حقلين هما no, next . (10 درجة)

س5 ب p مؤشرا إلى جذر شجرة ثنائية، اكتب دالة تبحث عن العنصر T في الشجرة الثنائية (T من نوع int) ثم تطبع الرسالة المناسبة. (10 درجة)

وفقكم الله وسدد خطاكم

جامعة العلوم والتكنولوجيا
كلية العلوم والهندسة
قسم علوم الحاسوب والمعلوماتية
اجب عن كل الأسئلة الآتية:

فرع الحديدة

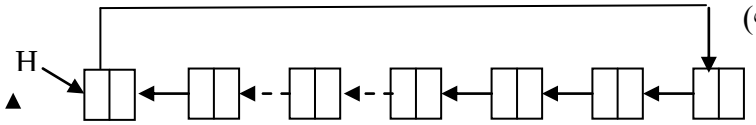
الزمن : 120 دقيقة : التاريخ :2010
المادة : هياكل البيانات : الفصل الثاني
الاختبار النهائي 60% المستوى الثاني
اسم الطالب:

س2 اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية ثم يطبع محتويات المصفوفة عناصر كل صف في سطر. (10 درجات)

3	6	9	12	15	90	21	24	27
27	30	33	36	39	90	45	48	51
51	54	57	60	63	90	67	72	75
75	78	81	84	87	90	93	96	99
3	6	9	12	15	90	21	24	27
27	30	33	36	39	90	45	48	51

س2 ـ ب- اكتب برنامجا يعرف مصفوفة أحادية مكونة من 200 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة بعكس الترتيب (كل 8 أعداد في سطر)، ثم يطبع في سطر آخر قيمة وموقع اصغر عنصرا في المصفوفة. (10 درجات)

س3ـ1- اكتب دالة تضيف node إلى قائمة متصلة Circular Linked list كل node تتكون من حفلين (no, next كما في الشكل التالي) . (10 درجات)

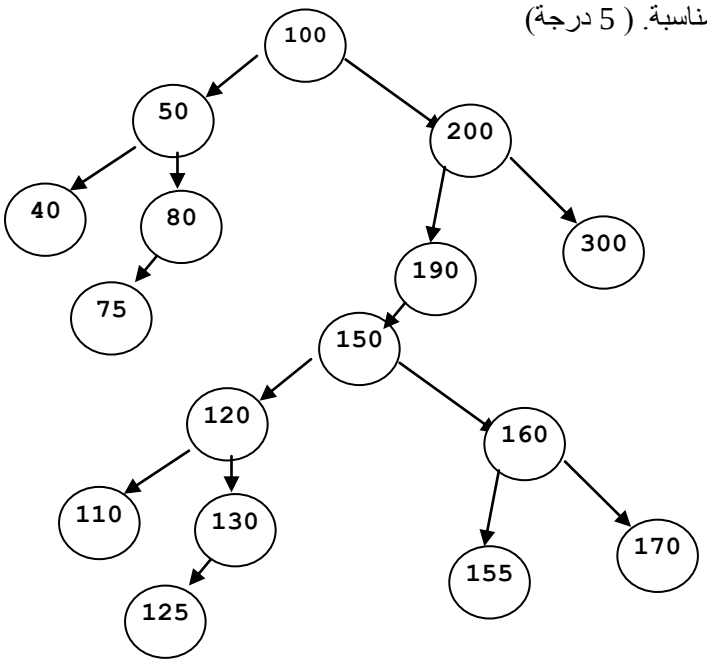


س- 3-2 اكتب دالة تحذف node من القائمة المتصلة Circular Linked list (كما في الشكل السابق) . (5 درجات)

س4ـ- p مؤشرا إلى قائمة متصلة Double Linked list اكتب دالة تبحث عن العنصر x في Double Linked list ثم تطبع الرسالة المناسبة (10 درجة)

س5 1- p مؤشرا إلى جذر شجرة ثنائية، اكتب دالة تبحث عن العنصر x في شجرة ثنائية ثم تطبع الرسالة المناسبة. (10 درجة)

2- اكتب محتويات الشجرة الثنائية بالطرق المناسبة. (5 درجة)



تمنياتي لكم بالتوفيق والنجاح

جامعة العلوم والتكنولوجيا
كلية العلوم والهندسة
قسم علوم الحاسوب والمعلوماتية
اجب عن كل الأسئلة الآتية:

فرع الحديدة

الزمن : 120 دقيقة : التاريخ :2010
المادة : هياكل البيانات : الفصل الثاني
الاختبار ا التكميلي %100 المستوى الثاني
اسم الطالب:

س$_1$ ا اكتب التعريف اللازم بلغة C++ وذالك لتوصيف سجل لطالب بياناته كالتالي:

اسم الطالب ، رقم الطالب، جنس الطالب، درجات المقررات الدراسية (5 مقررات)، المعدل العام، حالة الطالب، ثم كون هيكل للطلاب يتكون من 100 سجل. (5 درجات)

س$_1$ ب اكتب التعريف اللازم بلغة C++ وذالك لتوصيف node لتكوين شجرة ثنائية. (5 درجات)

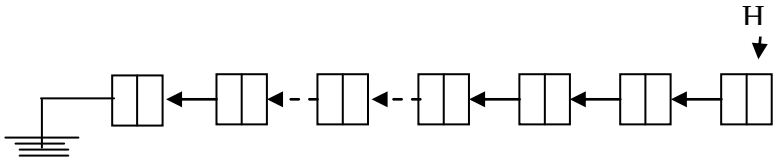
س$_1$ ج ارسم شجرة ثنائية تحتوى على البيانات التالية 50 85 95 135 35 130 110 90 80 120 100 70 140 115 . (10 درجات)

س$_2$ - اكتب بلغة C++ برنامجا يعرف مصفوفة ثنائية و يضع فيها القيم التالية ثم يطبع محتويات المصفوفة عناصر كل صف في سطر. (20 درجة)

27	**30**	**33**	**36**	**39**	**90**	**45**	**48**	**51**
51	**54**	**57**	**60**	**63**	**90**	**67**	**72**	**75**
75	**78**	**81**	**84**	**87**	**90**	**93**	**96**	**99**
3	**6**	**9**	**12**	**15**	**90**	**21**	**24**	**27**
27	**30**	**33**	**36**	**39**	**90**	**45**	**48**	**51**

س$_3$ - اكتب برنامجا بلغة C++ يعرف مصفوفة أحادية مكونة من 500 موقعاً، من نوع int ،ثم يولد في هذه المصفوفة أرقاما عشوائية، ثم يطبع محتويات المصفوفة (كل 7 أعداد في سطر)، ثم يطبع في سطر آخر قيمة وموقع اصغر عنصر في المصفوفة. (20 درجة)

س-4- اكتب برنامجا يكون قائمة متصلة Linked list تحتوى على 100 nodes (كل node تتكون من حقلين (no, next ثم يضع في كل node من هذه القائمة المتصلة Linked list رقما عشوائيا، ثم يجعل المؤشر H يشير إلى بداية القائمة المتصلة (كما في الشكل التالي) ثم يطبع محتويات هذه القائمة المتصلة. (20 درجة)



س-5- p مؤشرا إلى جذر شجرة ثنائية، اكتب دالة تبحث عن العنصر x في الشجرة الثنائية ثم تطبع الرسالة المناسبة. (20 درجة)