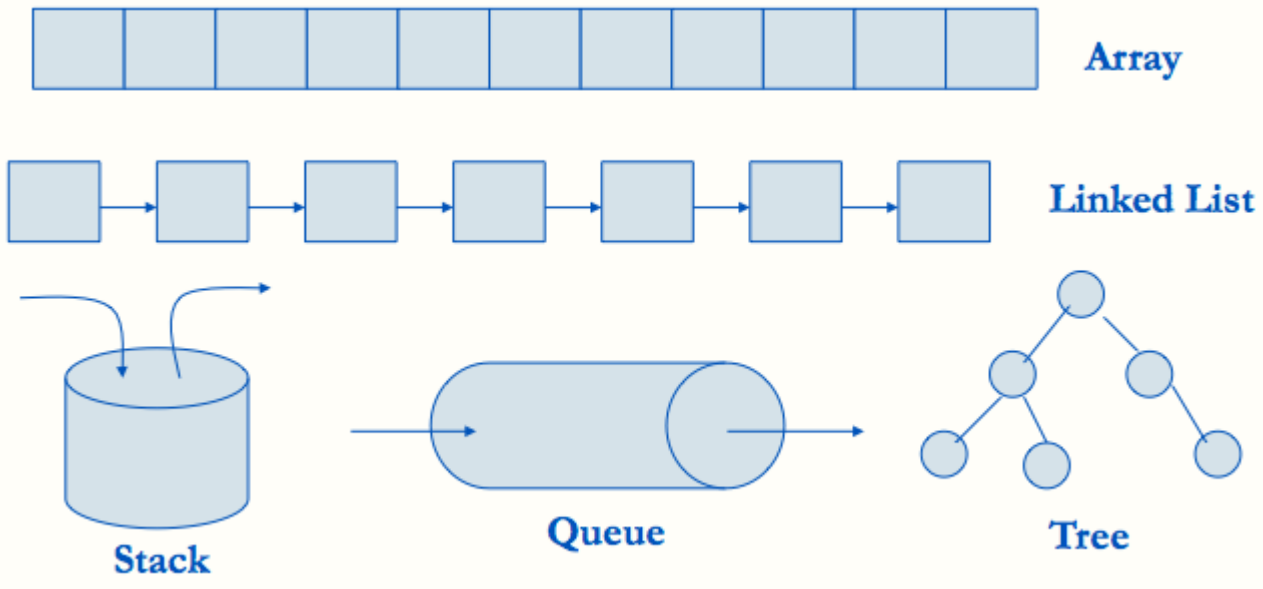
المعهد العالي للمهن الشاملة البركت

تراكيب (هياكل) البيانات

باستخدام Microsoft Visual Studio (VB.NET)

Data Structure Using VB.NET

(الجزء النظري)



التخصص: حاسوب

الفصل : الرابع

إعداد : أحمد محمد العربي الأنصاري

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

المقدمة ،،،،،،،

يعد مفهوما البيانات **Data** والمعلومات **Information** من المفاهيم الاساسية في دراسة الحاسبات واستخداماتها ، فإن بنية البيانات هي طريقة خاصة لتخزين و تنظيم البيانات في الكمبيوتر بحيث يمكن استخدامها بكفاءة تناسب أنواع مختلفة من هياكل البيانات و أنواع مختلفة من التطبيقات، وبعضها مخصص بدرجة عالية لمهام محددة. على سبيل المثال ، **(Tree)** بشكل خاص مناسبة تماما لتنفيذ قواعد البيانات ، في حين تنفيذ المترجم عادة ما يستخدم جداول **(Hash)** للبحث عن المعرفات، وتستخدم هياكل البيانات في كل برنامج تقريبا ، في هذا المنهج سيتم دراسة المفاهيم الاتية :

- مقدمة للمنهج ومراجعة أساسية .
- مراجعة سريعة للمصفوفات **Arrays** وأنواع **Data Structure** .
- تقنيات البحث **Searching Techniques** .
- بعض مشكلات القوائم المرتبة **Sorted List Matching Problem** .
- القوائم المرتبطة **Linked Lists** .
- الإستدعاء الذاتي **Recursion** .
- المتراصات و الطوابير **Stacks & Queues** .
- مقدمة عن الشجرة **Binary Trees** .
- خوارزميات الترتيب **Sorting Algorithms** .
- الخوارزميات الهندسية **Describing Geometric Algorithms** الاتية :
- - الرسوم البيانية وجداول التجزئة **Graphs & Hash Tables** .
- - خوارزميات الرسم **Describing Graph Algorithms**.

ملاحظة مهمة : يجب على كل الطلبة مراجعة منهج أساسيات البرمجة المدروس مسبقاً في الفصل الأول ومراجعة أهم المفردات الاتية : (الجمل الشرطية – الحلقات التكرارية والمتداخلة – الخوارزميات - المخرجات) ، ومراجعة منهج البرمجة الشيئية المدروس مسبقاً في الفصل الثالث،

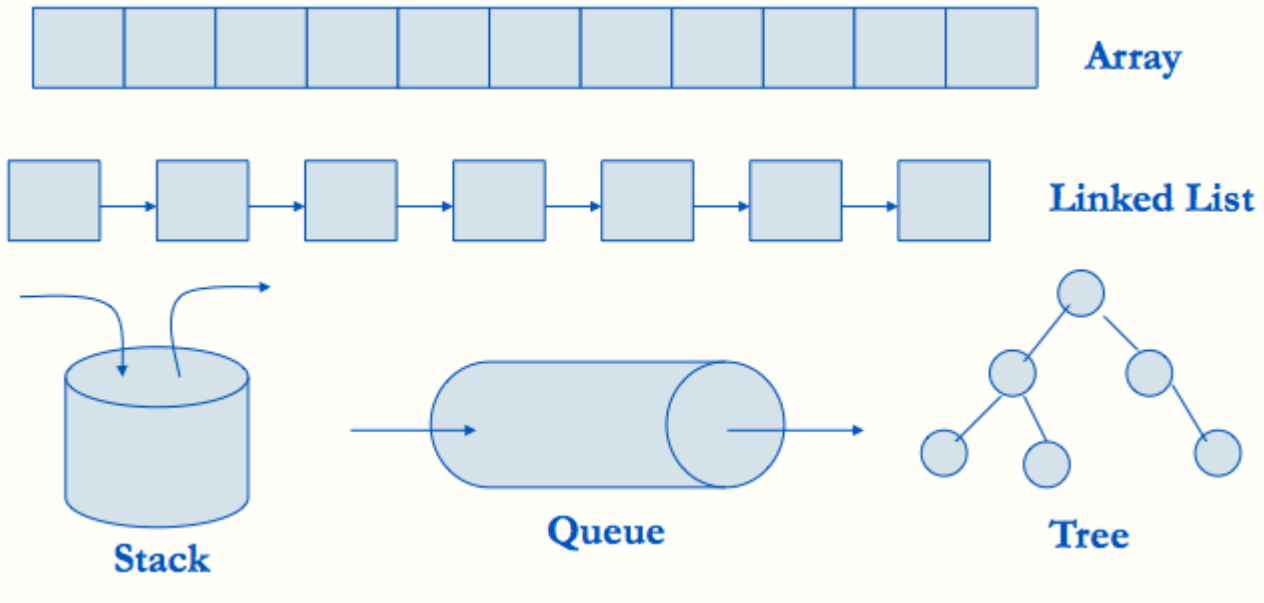
مبادئ تراكيب البيانات Elementary Of Data Structure:

تعريف تراكيب البيانات :

هي الأشكال التي يتم بواسطتها تنظيم البيانات في الذاكرة سواء كانت ذاكرة الحاسوب أو الأقراص المختلفة.

او هي طريقة محددة لجمع البيانات وترتيبها وتنظيمها في الحاسب الآلي بحيث أنه يمكن استخدامها بكفاءة.

من تلك الأشكال : **Stacks ,Trees , Hash ,Array ,Linked List, File , Record**



تعريف الصيغ الرياضية (الخوارزميات):

هي الطرق والنُسق التي يتم التعامل بواسطتها مع تراكيب البيانات ، وتتمثل في الإجراءات التي يتم بها معالجة البيانات كالتعليمات والأوامر ، وقد تكون كل الجمل التنفيذ البرمجية.

مفهوم عملية البرمجة : هي التكامل بين تراكيب البيانات والصيغ الرياضية للاستفادة من الحاسوب والمعلومات

يمكن تقسيم تراكيب البيانات إلى قسمين :

1- من حيث استخدامها للذاكرة :

- ذات الحجم الثابت **Static** مثل: المصفوفات **Arrays**
- ذات الحجم المرن **Dynamic** مثل: القوائم المرتبطة **Linked List**

2- من حيث تمثيلها شكلياً :

- تمثيلاً خطياً **Linear** مثل: المصفوفات **Arrays** والقوائم المرتبطة **Linked List**
- تمثيلاً غير خطي **Non-Linear** مثل: تركيبة الشجرة **Trees** أو التركيبة البيانية **Graphs**

العمليات التي نقوم بها على تراكيب البيانات :

- **العبور Traversing :** هو عبارة عن الوصول إلى كل عنصر داخل التركيبة سواء كانت مصفوفة أو قائمة مرتبطة
- **البحث Searching :** العثور على موقع عنصر ما داخل التركيبة.
- **الإضافة Insertion :** إضافة عنصر جديد إلى التركيبة.
- **الحذف Deletion :** إزالة عنصر من التركيبة.
- **الترتيب Sorting :** ترتيب بطريقة منطقية (هجائياً ، تصاعدياً ، تنازلياً)
- **الدمج Merging :** دمج عناصر تركيبة واحدة أو أكثر مع بعضها البعض.

تصنيف تنظيم البيانات (File Organization)

- **التنظيم التتابعي. (**Sequential Organization**)**
- **التنظيم النسبي.** (Relative Organization**)**
- **التنظيم التتابعي المفهرس. (**Indexed Sequential Organization**)**
- **التنظيم متعدد المفاتيح. (**Multi-Key Organization**)**

التمثيل الفيزيائي لهياكل البيانات: يتم تمثيل البيانات كما يلي :

للأعداد :

- (Binary using Sign).
- (Binary using Tow's-Complement).

للحروف:

- (EBCDIC / 8bits).
- (ASCII / 8bits).
- (Huffman Code).

لنصوص: باستخدام لمؤشر وعدد الحروف المكونة للنص.

أنواع البيانات (Types of Data)

لأنواع البيانات عدة مفاهيم ، والمعروف في أغلب البرامج أن أنوع البيانات هي المقصود بها نسق الذي يكون للبيان ، مثل الرقم الصحيح **Integer** والعشري **Decimal** والنص **String** والتاريخ.
يقصد بأنواع البيانات في تراكيب البيانات معنيين متلازمين، **المعنى الأول**:
هو عنصر البيان أو نسق البيان ، كالرقم الصحيح والعشري والنص ، وكذلك الأنواع التي يعرفها المستخدم **User Define** فعلى سبيل المثال يمكن تعريف نوع أرقام عشرية له قيمة نهائية معينة كحد أقصى لا يتجاوزه.
يمكن الحصول على أنواع أخرى من البيانات تكون ناتج عملية برمجية ، فالبرمجيات الحديثة تسمح بتكوين أجزاء برمجية متنوعة مثل الفئة **Class** والدالة **Function** ، بحيث تمرر لها أنواع بيانات فتقوم بمعالجتها بطريقة مدروسة مسبقاً ، ثم يصدر منها ناتج يمثل نوع بيان ما.
المعنى الثاني: يحدد العمليات الممكن تنفيذها على نوع البيان كالضرب (*) والقسمة (/) وجيب التمام Sin. فنوع بيان ما تنفذ عليه عمليات ما ولا تصلح على غيره

• أنواع البيانات الأولية (Primitives Type):

انواع البيانات الأولية هي الانواع التي ليست **Objects** ((الانواع **Objects** مثل **String** والتي تحتوي على صف من البيانات من النوع **char**)) هذه الانواع الأولية هي **boolean** والتي تأخذ قيمة اما **True** او. **False** والنوع **char** والذي يأخذ حرف هجائي وهو من الانواع الصحيحة غير اي ياخذ قيمة موجبة
واما الانواع الاخرى : **byte,short,int,long :**

• أنواع البيانات المركبة (Compound Type) :

هي أنواع مستمدة من أكثر من نوع أولي, ويمكن القيام بذلك بعدة طرق مختلفة, يطلق عليها وهي مجتمة تراكيب البيانات(أو هياكل البيانات) Data Structures, يجب أن تعلم أن هياكل البيانات تختلف عن البيانات الأولية.
مثلا: مصفوفة من أعداد صحيحة تختلف عن النوع الصحيح نفسه integer.
المصفوفات: تقوم بتخزين عدد من العناصر من نفس النوع وبترتيب معين.
السجلات(وتسمى أيضا تركيبات): هي أبسط أنواع هياكل البيانات.
الاتحاد: يقوم بتعريف عدة أنواع أولية بداخلة, يتميز أنه لايقوم بتخزين إلا قيمة واحدة لجميع المتغيرات طيلة تنفيذ البرنامج.
التعدادات: (Enumerated type) هي عبارة عن عدة ثوابت تحمل قيم مختلفة فيما بينها, يمكن أن تقارنها و أن تسندها إلى متغيرات أخرى.
النوع النصي :
الحروف: هي عبارة عن حرف أبجدي, أرقام, رموز, علامات.
نصوص: هي عبارة عن سلسلة حرفية, تستخدم في الجمل والكلمات.

• أنواع البيانات الاخرى :

المؤشرات والمراجع

المؤشرات : هي عبارة عن أماكن في الذاكرة تقوم بتخزين عناوين المتغيرات فقط, تشتهر بها لغة C/C++ التي تتميز عن بعض اللغات التي لاتدعم هذا النوع من البيانات.

المراجع: هي عبارة عن مؤشرات ثابتة, تقوم بتخزين عنوان المتغير, وبالتالي تصبح تؤشر على عنوانه في الذاكرة, أي أن كل تغير في المرجع هو في الحقيقة تغير في المتغير الأصلى, يمكن أن نسمي المرجع أحيانا اسم آخر للمتغير.

الدوال

تقوم الدالة بإعادة قيمة بنفس النوع التي تحمله, مثلا دالة **sine** نقوم بإعطائها الزواية وهي تقوم بإعادة جيب الزاوية.

• أنواع البيانات المجردة :

أي نوع لا يكون له هدف معين ليقوم بتنفيذه فهو نوع بيانات مجرد :
مثلا المكدس هو نوع من أنواع البيانات المجردة
أو مصفوفة(هي عبارة عن قطع في الذاكرة متجاورة مع بعضها لها نفس النوع)
أو قائمة مرتبطة (هي مجموعة من القطع في الذاكرة غير متجاورة تترابط مع بعضها بوساطة المؤشرات).

تعريفات أخرى لبعض انواع البيانات :

الملف File:

هو الحاوية التي تخزن فيها البيانات ، فعناصر البيانات يجب أن يكون لها مكان وعنوان يسمح بالوصول إليها في الذاكرة والتعامل معها وهو ما يقوم به الملف.

السجل Record:

تخزن البيانات في الملف على شكل وحدات متماثلة ، كل وحدة تتكون من عدد من عناصر البيانات.

المجال Field:

هو عنصر بيانات الذي يمثل عنصراً في سجل ، أي أن كل سجل يتكون من عدد من المجالات ، كل مجال له نوع بيان واحد.

المدخل Key:

هو مجال من مجالات السجل يستخدم لتحديد سجل ، وعند تحديد السجل عن طريق هذا المجال يمكن التعامل مع كل مجالاته الأخرى.

مدخل البحث Search Key:

هو مدخل خاص أي أنه مجالٌ يمكن بواسطته الوصول إلى سجل بعينه بسهولة ، بحيث يسمح بتمييزه من بين السجلات الأخرى.

الجدول التالي يوضح انواع بيانات Microsoft Visual Studio 2005

النوع Type	الحجم Size	القيم Values
Boolean	2 bytes	True or False
Byte	1 byte	0 to 255 (unsigned byte)
SByte	1 byte	-128 to 127 (signed byte)
Char	2 bytes	0 to 65,535 (unsigned character)
Short	2 bytes	-32,768 to 32,767
UShort	2 bytes	0 through 65,535 (unsigned short)
Integer	4 bytes	-2,147,483,648 to 2,147,483,647
UInteger	4 bytes	0 through 4,294,967,295 (unsigned integer)
Long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ULong	8 bytes	0 through 18,446,744,073,709,551,615 (unsigned long)
Decimal	16 bytes	0 to +/-79,228,162,514,264,337,593,543,950,335 with no decimal point. 0 to +/-7.9228162514264337593543950335 with 28 places
Single	4 bytes	-3.4028235E+38 to -1.401298E-45 (negative values) 1.401298E-45 to 3.4028235E+38 (positive values)
Double	8 bytes	-1.79769313486231570E+308 to - 4.94065645841246544E-324 (negative values) 4.94065645841246544E-324 to 1.79769313486231570E+308 (positive values)
String	variable	Depending on the platform, a string can hold approximately 0 to 2 billion Unicode characters
Date	8 bytes	January 1, 0001 0:0:00 to December 31, 9999 11:59:59 pm
Object	4 bytes	Points to any type of data
Structure	variable	Structure members have their own ranges

سلوك أنواع البيانات:

توجد أربعة عوامل تحدد بدقة سلوك المتغير وهي:

نوع البيان Data Type:

يحدد نوعية البيان من حيث تبعيته لفئة مثل الأرقام ، المحارف وغيرها.

مجال التعريف Scope defines: كل متغير يكون معرَّف في موضعين فقط:

داخل نطاق الإجراء **Procedure** أو الوظيفة **Function** أو الفئة **Class** الذي تم تعريفه فيه.

الإجراءات والوظائف التي تقع ضمن نطاقه أو المورثة فيها.

إمكانية الوصول Accessibility:

تتوقف على نوع التعريف ، فالمتغير المعرّف على انه محلي **Dim** أو **Private** فانه معرّف في نطاق الفئة ومكوناتها فقط ، وغير معرّف خارجها حتى وان تم توريثها. بينما لو تم تعريفه على انه عام **Public** فانه يكون معرّف فيها وفي مكوناتها وكذلك إذا ورثت الفئة. كما توجد إمكانيات أخرى مثل استخدام المتغير دون إمكانية التأثير في قيمته كما في تعريف النوع المحمي **Protected** ، وكذلك عند إرسال المتغير إلى إجراء أو وظيفة على انه قيمة.

فترة حياة Lifetime:

تُحدد الفترة الزمنية التي تنهي سريان فاعلية المتغير ، فعلى سبيل المثال المتغير المعرّف في إجراء ما يبدأ عمره الافتراضي مع استدعاء أو بدء الإجراء ولحظة انتهاء الإجراء تنتهي فاعلية المتغير.

تكرار الظهور والإرجاع (الاستدعاء الذاتي) Recursion

هو استخدام وظيفة بحيث تقوم باستدعاء نفسها بعدد محدد من المرات. يتم ذلك بإنشاء وظيفة تنجز عملية ما بحيث يوضع بها نظام تحكم يضبط لحظة إيقاف التنفيذ وفق عدد مدروس من المرات فقط.

عند الحاجة لتنفيذ المهمة لعدد من المرات يمكن تنفيذ ذلك باستدعائها في برنامج خارجي ، فمع تنفيذها تبدأ آلية التحكم تضبط عدد مرات التكرار إلى أن تصل الحد المضبوطة عليه فيتم الإيقاف من تلقاء نفسها. ففي هذه الحالة يتم الاستدعاء لعدد من المرات ويلحق كل استدعاء إرجاع النتيجة واستخدامها في الاستدعاء التالي وهكذا

تحليل الصيغ الرياضية Algorithm analysis:

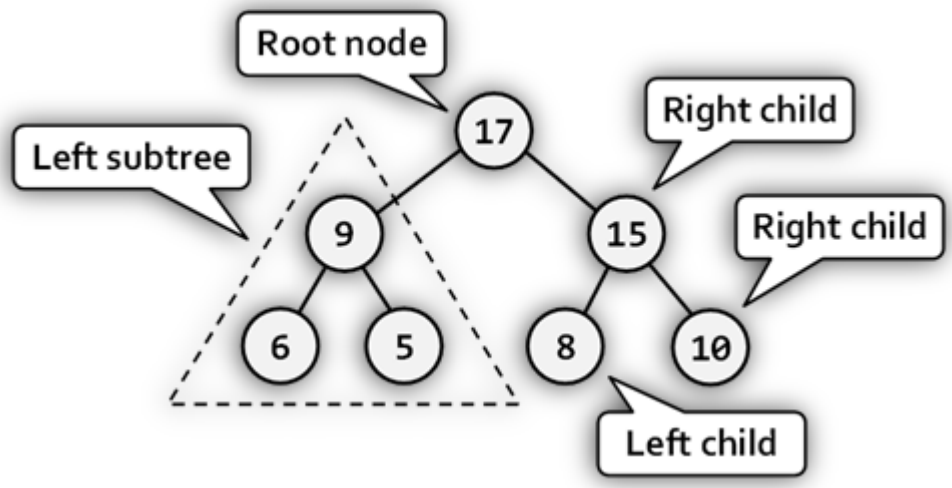
المفهوم العادي للصيغ الرياضية يركز على جانب استخدام العمليات الرياضية مثل الجمع ، الطرح ، الدوال ، المقارنات ، والتخصيص وغير ذلك. إلا أن المعنى في تحليل الصيغ الرياضية يركز على البرمجيات بصفة عامة ، بحيث تشمل الصيغ الرياضية العادية بالإضافة للجمل البرمجية الأخرى التي يحتويها التطبيق أو المشروع ، وبذلك تشمل الصيغ الرياضية المفهوم العادي والتعليمات المكونة لجمل البرنامج كلها.

في الجزء العملي أمثلة عن انواع البيانات ومجالات التعريف والاستدعاء باستخدام VB.net

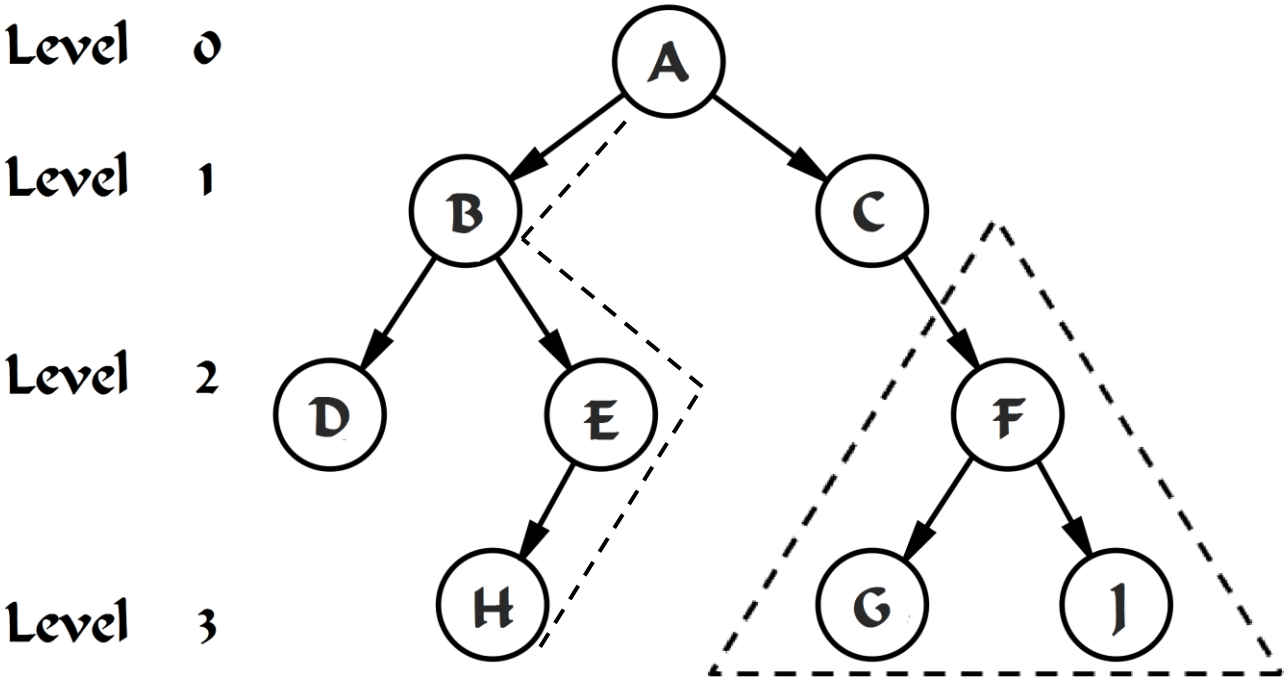
،،

الشجرة (Tree) :-

هي واحدة من الهياكل (تراكيب) الغير خطية (**Non-Linear**) **Hierarchical** ، تتكون الشجرة من مجموعة عقد **Nodes** يصل بينها أسلوب ربط ، في أعلى الشجرة عقدة الجزر وهي عقدة الاب بالأسفل منها عقد الابناء والأوراق.



مكونات الشجرة :-



في الشكل السابق شجرة ثنائية (**Binary Tree**) جزرها **Root** العقدة (**A**) في المستوى 0 ، العقدة (**A**) أب **Parent** للأبناء **Children** (**B , C**) في المستوى 1، ايضاً داخل الشجرة في المستوى 2 يوجد شجرة فرعية الاب فيها العقدة (**F**) ، كل خط يصل بين العقد هو اسلوب الربط **Edges** ، الخط المتقطع من العقدة (**A**) الى العقدة (**H**) يسمى المسار **Path** ، اي العقدة من غير ابناء تسمى ورقة (**Leaf**) ، العقد (**D , H , G , J**) أوراق **Leaves** .

أنواع الشجرة :

تنقسم الشجرة الى نموذجين :

1- نموذج الشجرة الثنائي (**Binary Tree**) : يكون للأب ابنين فقط الابن الايمن و الابن الايسر.

2- نموذج الشجرة المتعدد (**Multiway Tree**) : يكون للأب عدد ابناء أكثر من أثنين.

نموذج الشجرة ذات وصلات في اتجاه ليس الى الاسفل هو نموذج غير صحيح.

مميزات استخدام الشجرة:

سرعة العالية في البحث عن العناصر داخل الشجرة ، وسرعة العالية في تنفيذ عمليتي الإضافة والحذف.

تعريفات هامة:

الزيارة **Visiting**: هي العقدة التي يصل إليها التحكم خلال تنفيذ عملية.

العبور **Traversing**: هو زيارة جميع العقد في الشجرة .

المستويات **Levels**: هو عدد المرتبات التي تكون عليها العقد تنازلياً ، وذلك باعتبار مرتبة الجذر صفر.

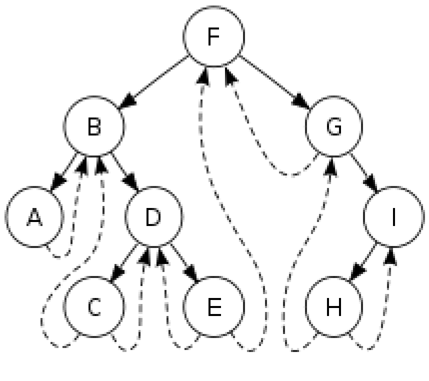
شروط نقل أوامر التحكم والبيانات في المسارات:

1- يمكن لكل عقدة أب أن تصدر أوامر تحكم إلى العقد أبنائها والعكس غير مسموح.

2- العقدة الابن يمكنها أن ترد على نتائج تنفيذ أوامر تحكم الصادرة من العقدة الأب.

3- البيانات تتحرك في الاتجاه من الأب إلى الابن فقط والعكس غير مسموح.

4- أي تعامل بين الأبناء في نفس المستوى يضعف التحكم في الشجرة.

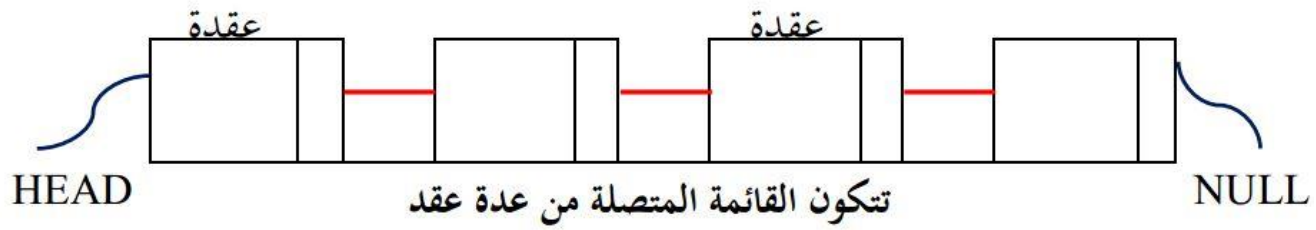


في الجزء العملي كافة العمليات على الشجرة باستخدام VB.net

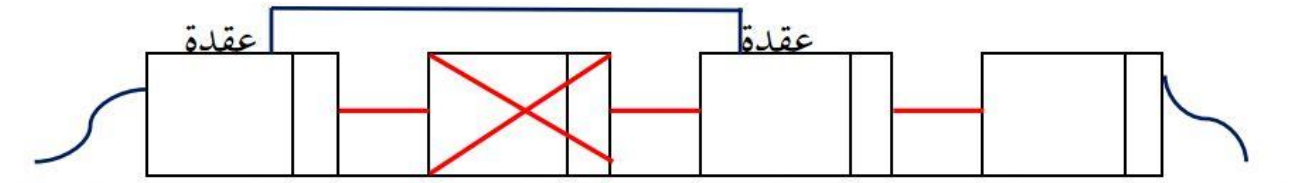
،،،

القوائم المتصلة Linked List :

هي عبارة عن هيكل (تركيب) بيانات مكون من عقد **Nodes** مرتبطة مع بعضها البعض ، ولها بداية ونهاية ، تسمى العقدة في القائمة المتصلة بالسجل .



تتكون القائمة المتصلة من عدة عقد

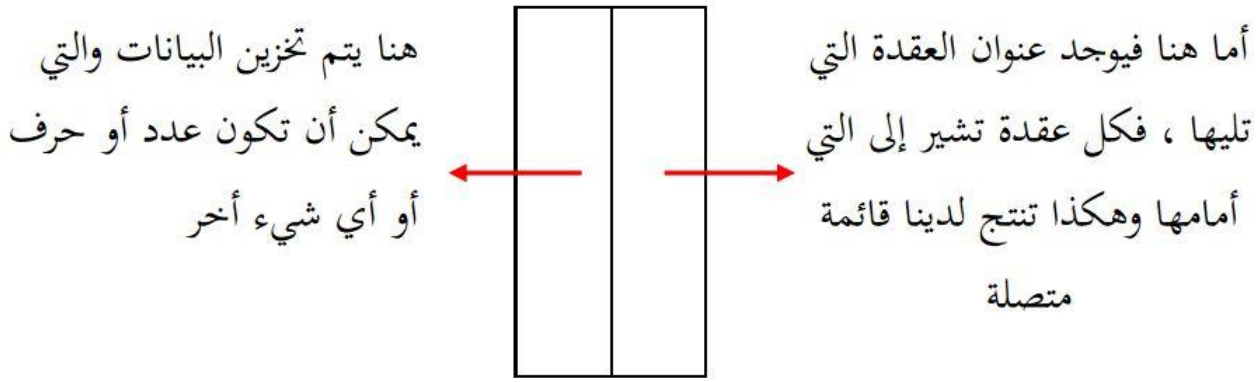


تتميز القائمة المتصلة بالقدرة على حذف العناصر أثناء تشغيل البرنامج

كل عقدة في القائمة تحتوي على الآتي :

1- بيانات **Data**

2- مؤشر إلى عقدة اخرى **Pointer to another node**



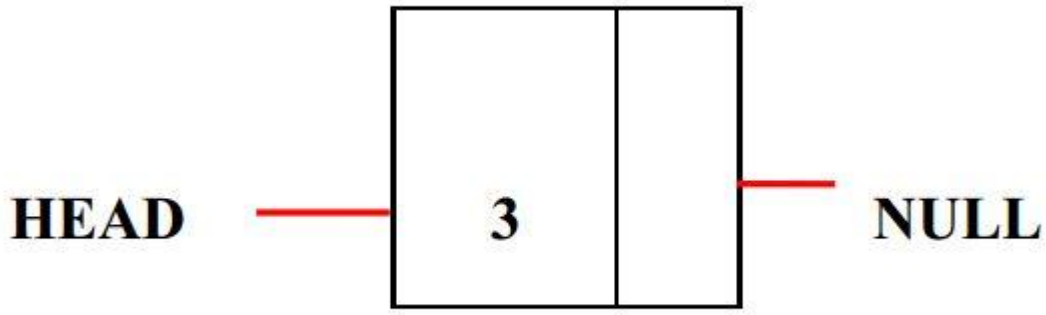
عقدة

Node

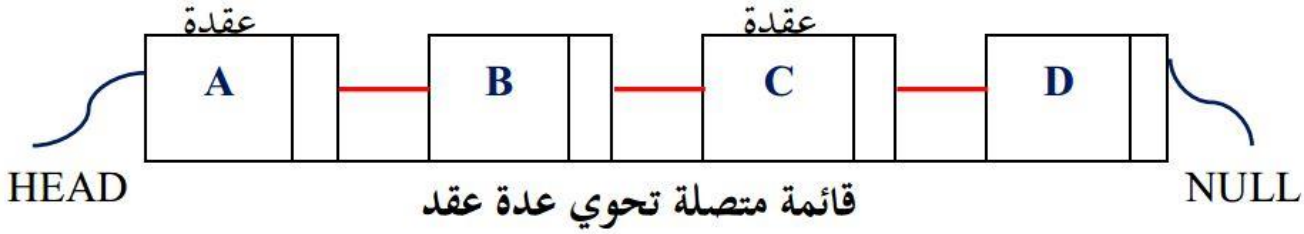
ترتبط العقد مع بعضها حتى تكون قائمة متصلة ، يكون المؤشر **Head** هو مؤشر لأول عقدة في القائمة المتصلة ، وإذا كانت القائمة خاليا فانه سيشير إلى **Null** والتي تعني فارغ او 0

أمثلة عن العقد والقوائم :

قائمة متصلة فارغة من العقد



قائمة متصلة تحوي عقدة واحدة



قائمة متصلة تحوي عدة عقد

أنواع القوائم المتصلة :

يوجد عدة انواع اهمها :

1- القائمة المتصلة المفردة **Single Linked List**

2- القائمة المتصلة المزدوجة **Double Linked List**

3- القائمة المتصلة الدائرية **Circular List Linked**

استخدامات القوائم المتصلة :

على سبيل المثال تستخدم بشكل أساسي داخل المصارف ، فحساب كل شخص هو عبارة عن عقدة **Node** فهي تستخدم لحفظ البيانات بشكل مؤقت وتتعامل مع البيانات ، وتستطيع التطبيقات أن تشكل القائمة من قاعدة بيانات أو أن تحفظ داخلها ، ايضاً يمكن ان تجرى على العقد العمليات التي ترغبها ، تسطيع أن تضيف حساب شخص بأي مكان من القائمة ، وحذف حساب شخص متى ما أردت ومن اي مكان في القائمة ،تستطيع ترتيب الحسابات ،عرضها ، تحريرها وكذلك تمديد القائمة وإضافة عناصر جديدة بكل سهولة وكفاءة، على عكس المصفوفات التي تتطلب تحريك العناصر.

القوائم المتصلة في Microsoft Visual Studio :

يمكن تعريف قائمة متصلة فارغة من نوع بيانات عشرية كالاتي:

```
Dim Numbers As LinkedList(Of Double) = New LinkedList(Of Double)
```

وايضاً يمكن تعريف و إضافة بيانات لعقد قائمة متصلة كالاتي:

```
Dim Values As List(Of Double) = New List(Of Double)
   Values.Add(84.597)
   Values.Add(6.47)
   Values.Add(2747.06)
   Values.Add(282.924)
Dim Numbers As LinkedList(Of Double) = New LinkedList(Of Double)(Values)
```

العمليات على القائمة مرتبطة Fundamental Operations on a Linked List :

معرفة عدد العقد في القائمة :

```
MsgBox(" عدد العقد في القائمة = " & CStr(Numbers.Count))
```

إضافة عقدة **Node :**

```
Numbers.AddLast(148.24)
```

أو

```
Dim x As LinkedListNode(Of Double) = New LinkedListNode(Of Double)(148.24)
    Numbers.AddLast(x)
```

أو

```
x = New LinkedListNode(Of Double)(35.75)
    Numbers.AddLast(x)
```

البحث والعثور :

```
If Numbers.Contains(2222.06) = True Then
If Numbers.Find(2747.06) Is Nothing Then
```

وهناك عمليات مثلا الحذف والوصل لأول عقدة أو العقدة الاخيرة وغيرها ،،،:

```
Numbers.First.Value
Numbers.Last.Value
Numbers.AddBefore(x, Number)
Numbers.AddAfter(x, Number)
Numbers.RemoveFirst()
Numbers.RemoveLast()
Numbers.Clear()
```

في الجزء العملي كافة العمليات على القوائم المتصلة باستخدام VB.net

،،

التركيب Structure Define :

هو عبارة عن مجموعة من العناصر التي تربطها مع بعضها البعض علاقات وثيقة وهذه العناصر من الممكن أن تكون من بيانات مختلفة النوع (**Integer,String,Decimal**) وتخزن هذه المعطيات في الذاكرة بشكل متتالي تحت اسم واحد ، ويستخدم لتمثيل بيانات شيء ما ، مثل سجل بيانات طالب ، سجل بيانات سيارة وهكذا،،،،

فمثلاً التركيب التالي يوضح سجل بيانات طالب :

لدينا المعلومات التالي: رقم الطالب ، اسم الطالب ، المدينة ، التخصص ، تاريخ التسجيل.، تستخدم الكلمة **Structure** لإنشاء التركيب كما يلي :

```
Structure Student
  Dim Number As Intger
  Dim Name As String
  Dim city As String
  Dim special As String
  Dim BDate As Date
End Structure
```

يتم تعريف متغير من التركيب السابقة ،،كما يلي :

```
Dim info As Student
```

ادخال البيانات كالاتي :

```
info.Number = 7669943
info.Name = "Ahmed Mohmed Alarbi Alansari"
info.city = "Ghat"
info.Special = "Computer Scienc"
info.BDate = #9/2/1989#
```

يمكن استدعاء وطباعة أي بيان داخل التركيب :

```
TextBox1.Text = info.Number
TextBox2.Text = info.Name
 Debug.WriteLine(info.city)
 Debug.WriteLine(info.Special)
MaskedTextBox1.Text = info.BDate
```

في الجزء العملي كل الامثلة عن التركيب باستخدام VB.net

''

المصفوفات Arrays :

هي عبارة عن تركيب متوال من عناصر من نفس النوع تكون منظمة في الذاكرة في أماكن متسلسلة ومتجاورة. وتعتبر المصفوفات هي الأكثر استخداماً في تراكيب البيانات،

خصائص المصفوفات:

1- تحمل عناصر من نفس النوع فقط
2- تحجز مكاناً في الذاكرة بمجرد إنشاءها
3- حجمها ثابت **Static** ولا يمكن تغييره بمجرد انشاءها
4- يمكن الوصول لعناصرها مباشرة
5- المصفوفات تعتبر كائنات **Objects**
6- يمكن نسخها بواسطة الطرق **Methods**
7- تنتج الخطأ **ArrayIndexOutOfBoundsException** عندما يشار إلى عنصر خارج مجالها

العمليات التي نقوم بها على المصفوفات :

1- الوصول للعناصر Accessing Elements

الوصول للعنصر في المصفوفة يعتبر لحظي وسريع جداً فبمجرد كتابة رقم الـ**Index** فقد اصبح بإمكانك تعديل وتحرير العنصر.

2- إضافة العناصر Inserting Elements

إضافة العناصر للمصفوفة يملك إحدى حالتين إما تضيف في البداية أو المنتصف وهنا تكون العملية بطيئة جداً وتحتاج لإزاحة كافة العناصر **Shifting**.

أما إضافة العناصر في آخر المصفوفة فهو لحظي وسريع ولا يحتاج إلى عمل **Shifting**.

3- حذف العناصر Removing Elements

حذف العناصر في المصفوفة هو نفس حالة الإضافة، فإذا كان الحذف من بداية أو منتصف المصفوفة فإنه يكون بطيء ويجب عمل **Shift** لباقي العناصر.

أما حذف عنصر من آخر المصفوفة فهو لحظي وسريع لأنه لا يحتاج إلى عمل **Shift**.

4- البحث عن العناصر Searching for Elements

البحث عن العناصر يعتمد على الـ**Algorithm** المستخدمة، وسنتحدث عنها لاحقاً.

المصفوفات في Microsoft Visual Studio :

يتم تعريف وادخال بيانات مصفوفة حسب نوع البيان بأحد الطرق الاتية :

```
Dim Ar() As Byte = {10, 20, 30, 40}
```

أو

```
Dim ar(3) As Byte
 ar(0) = 10 : ar(1) = 20 : ar(2) = 30
```

العمليات على المصفوفات Fundamental Operations in Arrays :

طباعة و اخراج بيانات مصفوفة :

```
    For i = 0 To 3
     Debug.WriteLine(ar(i))
    Next
TextBox1.Text=ar(2)
```

العمليات باستخدام الدالة Array :

```
Array.Sort(ar)
```
ترتيب المصفوفة تصاعدياً من اصغر إلى أكبر أو هجائياً

```
Array.Reverse(ar)
```
عكس ترتيب المصفوفة بحيث القيمة الاولى تصبح في الاخير وهكذا،،

```
Array.BinarySearch(ar)
```
بحث عن عنصر بالطريقة الثنائية

```
Array.Copy(ar,x,int32)
```
نسخ عناصر من مصفوفة الى اخرى بطول 32 بت أو 64 بت

```
Array.Find(ar, AddressOf x)
```
بحث عن عنصر يطابق شروط محددة وإرجاع

القوائم (المصفوفات) Array List :

```
Dim Lar As New ArrayList(4)
Dim i As Byte

Lar.Add("Ahmed")
Lar.Add("Mohamed")
Lar.Add("ALarabi")
Lar.Add("ALansari")

For i = 0 To 2
     Debug.WriteLine(Lar(i))
Next
```

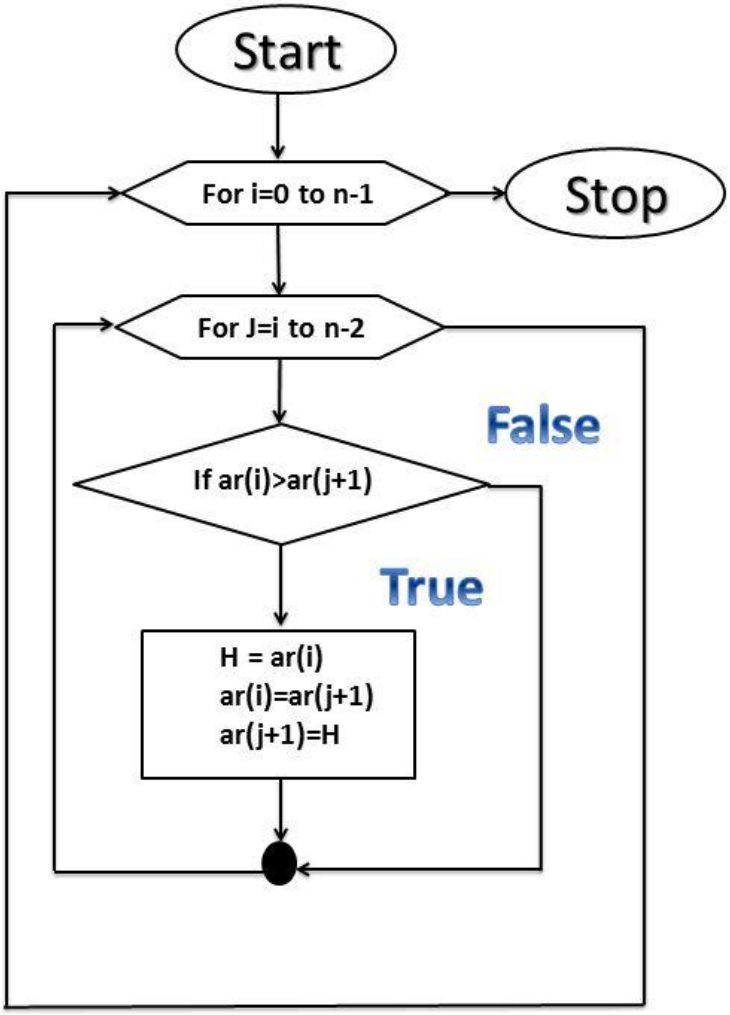
خوارزميات الترتيب والبحث Sorting Algorithms :

عملية البحث : يمكن بحث عن اي عنصر في المصفوفة بعدة طرق منها :

البحث الثنائي **BinarySearch**
البحث الخطي **LinearSearch**

عملية الترتيب : يمكن ترتيب العناصر في المصفوفة تنازلياً أو تصاعدياً أو استبدال قيم بعدة طرق واساليب منها :

طريقة الاختيار **Selection Sort**
طريقة الدمج **Merge Sort**
طريقة الفقاعة **Bubble Sort**
طريقة الاضافة **Insertion Sort**



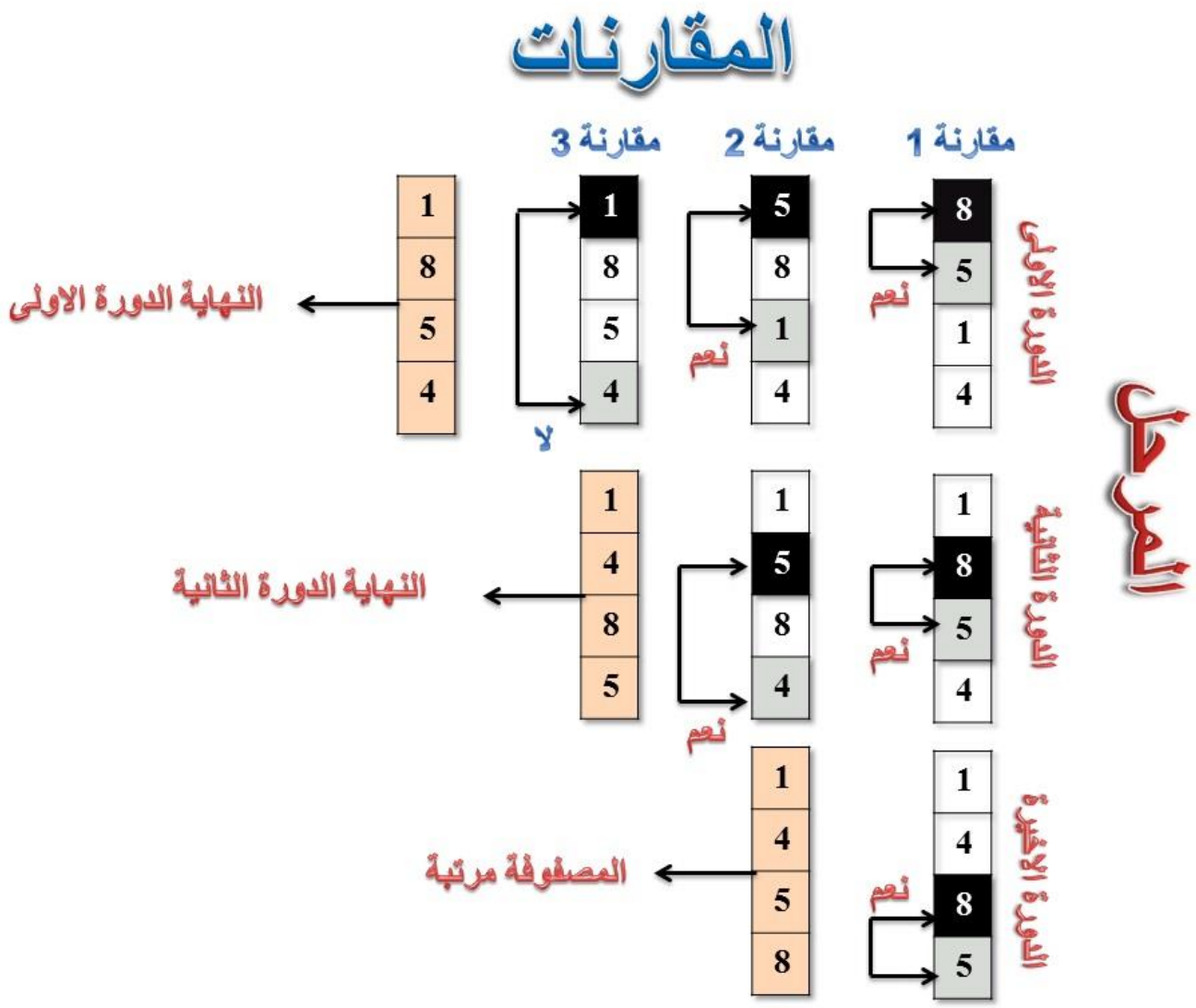
شرح خوارزمية الترتيب بطريقة Bubble Sort في المصفوفة :

هي تعمل على رفع العنصر الأكبر كفقاعة الهواء التي ترتفع إلى أعلى وذلك بترتيب العناصر بتتابع. أي نقوم مقارنة العنصرين الأول والثاني، نحتفظ بالعنصر الأكبر في مكان أخر في الذاكرة ، ونبدل الأماكن إذا كانا غير مرتبين. نقوم بهذه العملية إلى آخر عنصر. بعد ذلك نعيد العمليات إلى أن المكان ما قبل الأخير وهكذا،،،!!

لترتيب N عناصر في المصفوفة A ، عدد المقارنات سيكون. $\sum_{i=0}^{N} x_i$:

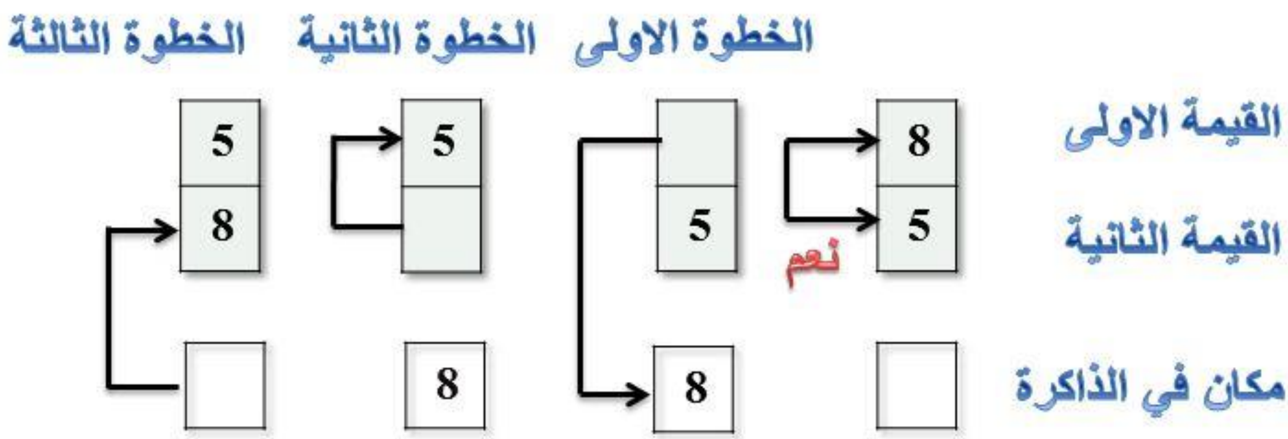
تعقيد الخوارزم هو $\mathcal{O}(n^2)$ في المعدل ،، و $\mathcal{O}(n)$ في الحالة المثلى.

مثال : ترتيب مصفوفة بيانات (8 ، 5 ، 1 ، 4) بطريقة Bubble Sort كالاتي :



في شكل السابق لا حظ في الدورة الاولى يتم مقارنة القيمة الاولى مع الثانية ،في حال القيمة الثانية أصغر تتم عملية الاستبدال ، وبعدها مقارنة القيمة الاولى مع الثالثة ،وهكذا،، الى القيمة الاخيرة ،في حال القيمة الثانية أكبر من الاولى تبقى المصفوفة على نفس الترتيب بدون استبدال.

عملية الاستبدال تتم في الخطوات الاتية :
كما ذكرنا سابقاً في حال القيمة الثانية أصغر يتم نقلها الى مكان في الذاكرة ثم نقل القيمة الاولى مكان القيمة الثانية ،وبعد ذلك نقل القيمة الثانية الى مكان القيمة الاولى كالاتي:



برمجياً يتم تنفيذ بالخطوات الاتية :

```
Dim ar() As Byte = {8, 5, 1, 4}
 Dim i, j, H As Byte

 For i = 0 To Ar.Length - 1
   For j = i To Ar.Length - 2

     If Ar(j + 1) < Ar(i) Then
        H = Ar(i)
        Ar(i) = Ar(j + 1)
        Ar(j + 1) = H
     End If

   Next
 Next
```

$x(0)$ $x(i)$ $\sum_{i=0}^{i=n} x(i)$

$$\sum_{i=0}^{i=n} x(i) = 1 + 2 + 3+,,,,,+(n-1) = x(n^2)$$

في الجزء العملي كافة العمليات على المصفوفات باستخدام VB.net

،،

المتراصات (المكدس) Stacks:

تعريف المكدس: هو عبارة عن نموذج خاص لتخزين البيانات بآلية ثابتة وإخراجها بشكل مؤقت و المتراصة عبارة عن بيانات متسلسلة ومتلاصقة في الذاكرة وتوضع فيها البيانات بآلية الداخل أولاً هو الخارج أخيراً أو الداخل أخيراً خارج أولاً :

Last in First Out(LIFO) أو First in Last Out(FILO)

المتراصات لها عدة استخدامات فمثلاً في المعالج الميكروي وذلك لتنفيذ العمليات حسب أولوياتها.

العمليات على المتراصات (المكدسات) Fundamental Operations in Stacks :

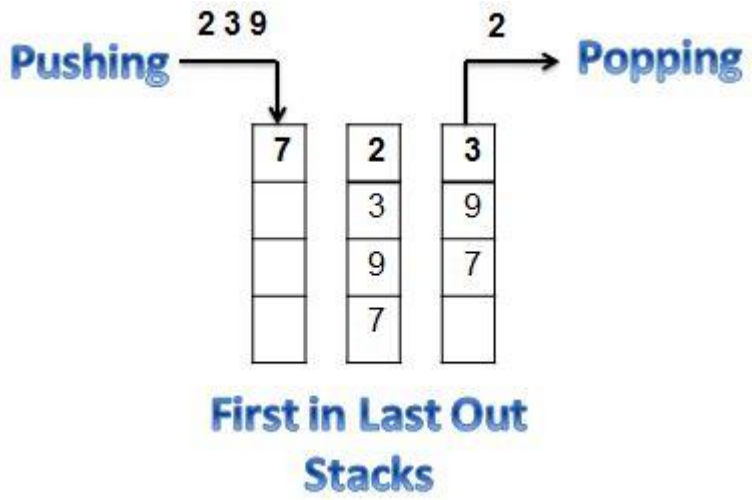
هناك أربعة عمليات اساسية تتم على المكدس وهي :

1- إضافة عنصر إلى المكدس
2- حذف عنصر من المكدس
3- اختبار حالة المكدس فيما إذا كان ممتلئاً أم لا
4- اختبار حالة المكدس فيما إذا كان فارغاً أم لا

وكل العمليات السابقة تعتمد على مؤشر واحد هو **Top** فعندما يكون المكدس فارغاً فإن **top=-1** وعند إدخال **Pushing** أول بيان يصبح في أعلى المتراصة ،ويزداد المؤشر بمقدار واحد إي **top++** وكل ما أدخلنا قيمة فان المؤشر يزداد بمقدار واحد إلى أن يمتلئ المكدس، وعند إدخال بيان ثاني يأخذ المكان في أعلي المتراصة والأول يحل في المرتبة الثانية وهكذا. في المتراصة يمكن الوصول إلى بيان واحد فقط وهو الذي في الأعلى ، وعملية الوصول إليه تمثل عملية إخراجه أو حذفه وتسمى **Popping** وعملية الإخراج تتم بإنقاص قيمة المؤشر بمقدار واحد في كل مرة إلى أن تصل قيمة المؤشر 1- يكون المكدس فارغاً

مثال : تخزين بيانات **(7 ، 9 ، 3 ، 2)** في مكدس :

```
Dim Stc As New Stack(4)
 Dim i As Byte

  Stc.Push(7)
  Stc.Push(9)
  Stc.Push(3)
  Stc.Push(2)

For i = 0 To 3
Debug.WriteLine(Stc.Pop)
Next
```



في الجزء العملي كافة العمليات على المتراصات باستخدام VB.net

طوابير البيانات Queues:

تعريف الطابور: هو عبارة عن نوع من هياكل البيانات الخطية لتخزين البيانات بشكل مؤقت. الطوابير تشبه المكدسات فهي عبارة عن بيانات متسلسلة ومتلاصقة ومتقاربة بشكل خطي في الذاكرة وليست على مواقع متفرقة ، الفارق يكمن في أن التنظيم المتبع لإدخال البيانات وإخراجها هو الداخل أولاً الخارج أ أولاً والداخل أخراً خارج أخراً كالاتي :ومرتبة كالاتي:

Last in Last Out(LILO) أو **First in First Out(FIFO)**

يحتوي الطابور على مؤشرين :

1- مؤشر الرأس ويسمى **Head or Front** ويتم عن طريقه حذف العناصر من الطابور .

2- مؤشر الذيل ويسمى **Tail or Rear** ويتم عن طريق هذا المؤشر إضافة عناصر إلى الطابور.

يستخدم الطابور في حال تنظيم التعامل مع عمليات لها نفس الأولوية حيث التعامل و حسب أسبقية وصولها. مثلاً طابور الانتظار لأفراد في المؤسسات المصرفية أو تصطف السيارات عند الإشارات المرورية على شكل طابور.

أنواع الطوابير :

1- **طابور خطي Linear Queue :** له حجم محدود وشرط امتلائه أن تكون قيمة الذيل تساوي حجم المصفوفة (العناصر) .

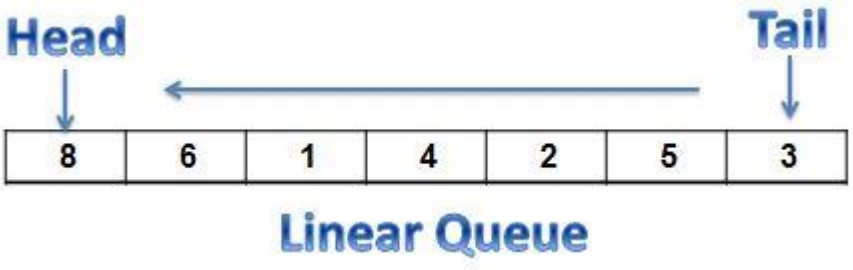
في بداية الطابور الخطي يكون الرأس والذيل لا يؤشران إلى أي موقع وعندها يكون الطابور فارغاً ، وفي هذه الحالة يكون **Front= -1** و **Rear=-1** ، عند إدخال **Enqueue** أول بيان يصبح في أول الطابور وتصبح قيمة الرأس و الذيل تساوي صفر. ، وعند إدخال بيان ثاني يأخذ المكان الثاني في الطابور ويحل في المرتبة الثانية في هذه الحالة تزداد قيمة الذيل بمقدار واحد ويبقى الرأس كما هو .

في عملية الإخراج أو حذف **Dequeue** تنقص قيمة الذيل بمقدار واحد ويبقى الرأس كما هو مع إزاحة الطابور لليسار مع كل عملية حذف .

2- طابور دائري Circler Queue : له حجم محدود وشرط إمتلائه يختلف عن الطابور الخطي وهو الرأس يساوي 1 والذيل يساوي حجم المصفوفة (العناصر) ،

أو بمعنى أخر الرأس = الذيل + 1

Front=1 and Rear=n حيث **n** تمثل حجم العناصر (المصفوفة) أو **Front=Rear+1**



Linear Queue

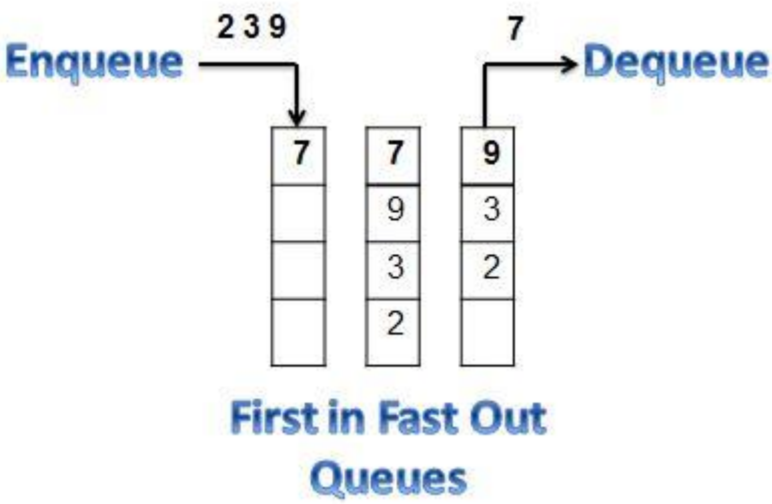
العمليات على الطوابير Fundamental Operations in Queues :

هناك أربعة عمليات اساسية تتم على الطابور وهي :

1- إضافة عنصر إلى الطابور
2- حذف عنصر من الطابور
3- اختبار حالة الطابور فيما إذا كان ممتلئاً أم لا
4- اختبار حالة الطابور فيما إذا كان فارغاً أم لا

وكل العمليات السابقة معتمدة على مؤشر الرأس و مؤشر الذيل ، سوء كان نوع الطابور خطي أو دائري ، كذلك اختبار حالة الطابور فارغ أو ممتلئ لا يعتمد على نوع الطابور .

مثال : تخزين العناصر **(7 ، 9 ، 3 ، 2)** في طابور بيانات :




```
Dim Que As New Queue(4)
 Dim i As Byte
  Que.Enqueue(7)
  Que.Enqueue(9)
  Que.Enqueue(3)
  Que.Enqueue(2)
For i = 0 To 3
  Debug.WriteLine(Que.Dequeue)
Next
```

في الجزء العملي كافة العمليات على الطوابير باستخدام VB.net

،،

الرسوم البيانية وجداول التجزئة Graphs & Hash Tables :

جداول التجزئة **HashTable** تعد من أهم وأسرع هياكل البيانات على الإطلاق ، وكثير من التطبيقات تستخدم مثل هذه البنيه مثل **Spell Checker** أو **Symbol Table** في المترجمات ، حيث تضمن لنا هذه البنيه الوصول السريع جدا لأي بيانات نريدها مهما كان حجم تلك البيانات ، بالإضافة الى ادخال البيانات يتم في سرعه كبيره (مثلها مثل المصفوفة) ،، اضافه الى ميزه السرعة هناك ميزه جيده بها وهي أنها سهله التطبيق حيث أننا سنطبق هذه البنيه من خلال مصفوفه عاديه أو **Vector**. بالرغم من ميزات هذه البنيه ، الا أنها يجب أن تستخدم في الأوقات المناسبة وقد لا تصلح لكل حاله ، لأن أداء هذه البنيه يقل تدريجيا كلما أمتلئت المصفوفه **Table** ، لذلك من البداية يجب أن نحدد حجم البيانات التي سوف توضع في هذه المصفوفه ونقوم بحجز مساحه مناسبه لهذا الحجم .

باستخدام **Hash Table** لا يمكن زياره جميع العناصر في المصفوفه (بمعنى أصح لا توجد فائده من هذه العملية **Visiting** لأن البيانات مخزنه في أماكن عشوائية بلا ترتيب) ، وبالتالي لا يمكنك مثلا اجراء عمليه ايجاد القيمة الأكبر في هذه البنيه **Find Max** ، وفي حال أردت أن تستخدم بنيه توفر لك هذه الميزة مثلاً **Binary Tree** . يفضل أن تستخدم في حال كانت البيانات التي تتعامل معها ثابته ولا تتغير ، وتريد الوصول لها في كل مره بشكل سريع ، مثلا البرامج التي تستخدم القاموس **Dictionary** يمكن أن تقرأ هذه القاموس وتضعه في بنيه **Hash Table** في الذاكرة وقت تشغيل البرنامج. أحد أهم المفاهيم في **Hashing** بشكل عام هو كيف يمكن تحويل مجموعه من المفاتيح الى مواقع معينه في **Index** في المصفوفه ، في أبسط الأحوال لا تكون هناك أي عمليه تحويل ويكون المفتاح هو نفسه مباشره الموقع **Index** في المصفوفه ، لكن هناك حالات أخرى كثيره لا يكون هناك مفتاح من أساسه وبالتالي عمليه التحويل من أي قيمه الى موقع **index** سوف تتم باستخدام الدالة المعروفة **Hash Function** .

جداول التجزئة في Microsoft Visual Studio :

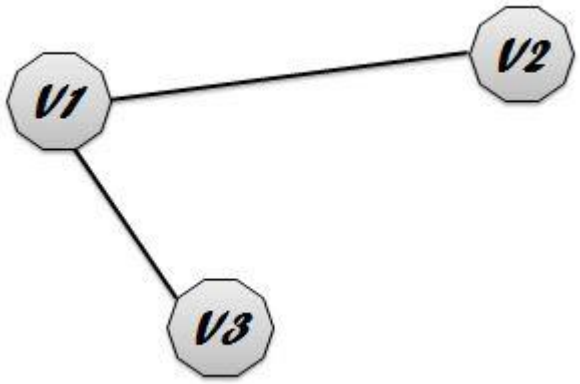
```
Dim openWith As New Hashtable()

  openWith.Add("txt", "notepad.exe")
  openWith.Add("bmp", "paint.exe")

  openWith.Remove("txt")
  openWith.Clone()
If Not openWith.ContainsKey("doc") Then
  Debug.WriteLine("Key ""doc"" is not found.")
End If
  openWith.Clear()
```

في الجزء العملي أمثلة على جداول التجزئة باستخدام VB.net

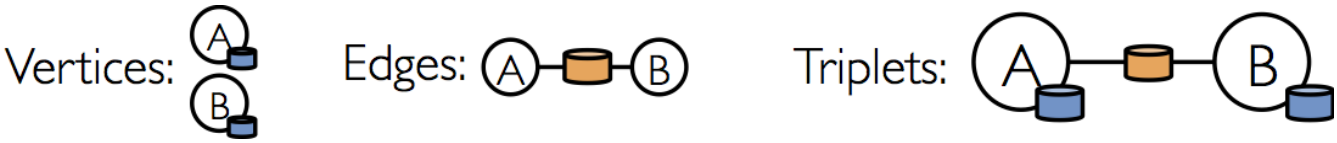
،،

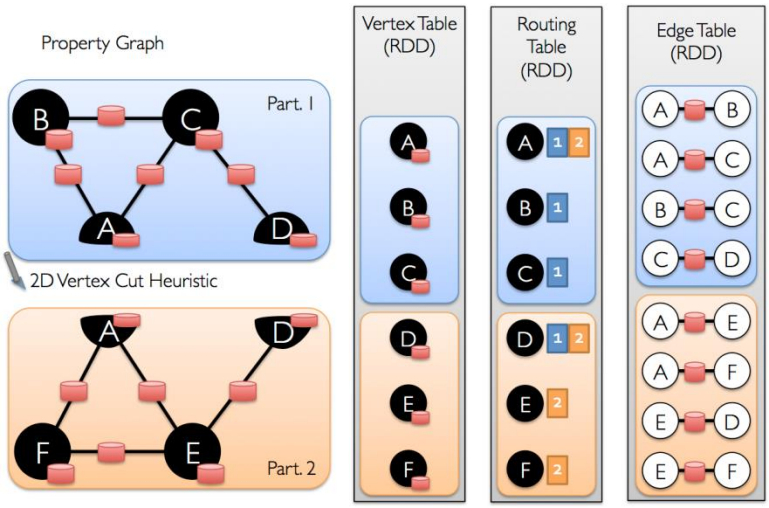
خوارزميات الرسم (المخططات) Graphs : النقطة هي وحدة الأساسية في الرسم ، تعتبر شاشة الحاسوب عبارة عن مجموعة من النقاط ، كل نقطة لها إحداثيات **(X , Y)** ، يمثل الرسم في الحاسوب بنقاط **V (Vertices)** وحواف **E (Edge)** ، فإذا كان لدينا النقاط التالية: **V = (v1,v2,v3)** يمكن إنشاء الحواف التالية:

E = {(v1,v2) , (v2,v3) }

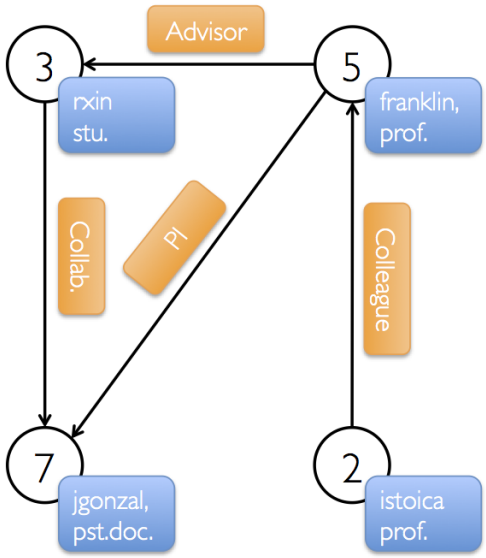


في شاشة الحاسوب تعتبر كل نقطة في الرسم إما مضاءة أو غير مضاءة ، وعليه فان الخط المستقيم هو عبارة عن عدد من النقاط المتسلسلة المضاءة ، ما يكوّن لدى المستخدم مظهراً بصرياً على شكل خط مستقيم. ،، الرسم ذو البعد الثلاثي **3D** هو في الأساس رسم ثاني **2D** إلا أن قدرات حاسبات الحديثة تجعل الشاشة تظهر ظلال وتدرجات لونية غاية في الدقة ، وذلك نتيجة لما توفره التقنيات الحديثة من قدرات في معالجة المعادلات المعقدة للرسم ، وكذلك قدرات بطاقات الشاشة الملونة والبرمجيات المتطورة.





Property Graph

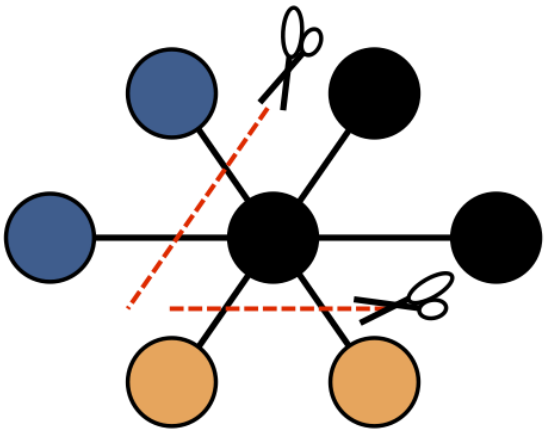


Vertex Table

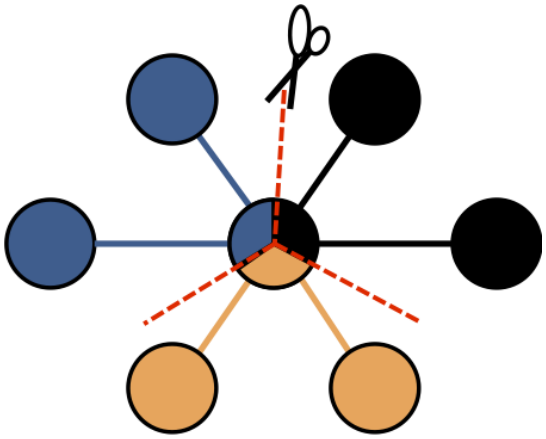
Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI



Edge Cut



Vertex Cut

المخططات Graphs في Microsoft Visual Studio :

```
Dim nVertices As Integer = 0
    vertices(nVertices) = New Vertex("A")
    nVertices += 1
Dim theGraph As New Graph
    theGraph.addVertex("CS1")
    theGraph.addEdge(0, 1)
    theGraph.TopSort()
    theGraph.DepthFirstSearch()
    theGraph.path()
```

```
Imports System.Drawing
Imports System.Threading
  Private m_GraphThread As Thread
  Private m_Y As Integer
   Private Sub DrawGraph()
     Dim y As Integer = m_Y
       Do
      PlotValue(y, m_Y)  :      y = m_Y
       Loop
    End Sub
Dim wid As Integer = picGraph.ClientSize.Width
Dim hgt As Integer = picGraph.ClientSize.Height
Dim bm As New Bitmap(wid, hgt)
Dim gr As Graphics = Graphics.FromImage(bm)
Dim m_Ym As Integer
Dim GRID_STEP As Integer = Convert.ToInt32(txtGridSpacing.Text)
       m_Ymid = hgt / 2
       gr.DrawImage(picGraph.Image, -1, 0)
       gr.DrawLine(Pens.Blue, wid - 1, 0, wid - 1, hgt - 1)
For i As Integer = m_Ym To Graph.ClientSize.Height Step GRID_STEP
    gr.DrawLine(Pens.LightBlue, wid - 2, i, wid - 1, i)
Next i
 For i As Integer = m_Ymid To 0 Step -GRID_STEP
    gr.DrawLine(Pens.LightBlue, wid - 2, i, wid - 1, i)
 Next i
   gr.DrawLine(Pens.White, wid - 2, old_y, wid - 1, new_y)
   picGraph.Image = bm
   picGraph.Refresh()    :      gr.Dispose()
```

في الجزء العملي مثال على الرسم Graph باستخدام VB.net

،،

في الختام اتمنى ان اكون قد وفقت في شرح هذا المقرر شرحا لا ملل فيه ولا تقصير وان يوفقنى الله واياكم لما فيه رضاه .

المدرس : أحمد محمد العربي الأنصاري

****

تمنياتي لكم بالتوفيق والنجاح

****