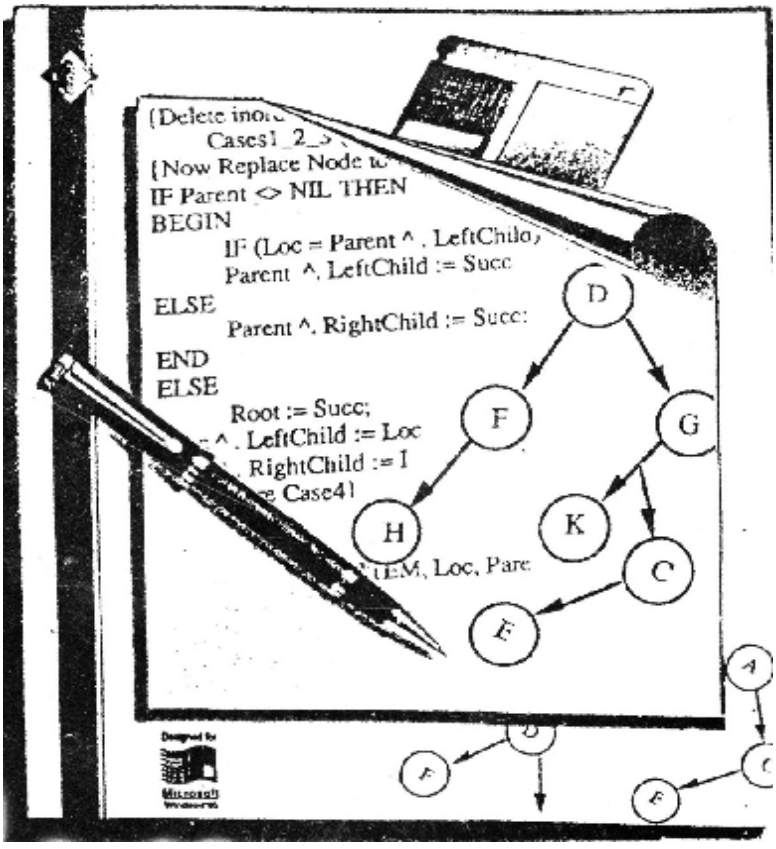
كلية الدراسات المتوسطة
College of Intermediate Studies
1417-1996
Al - Azhar University - Gaza
جامعة الأزهر - غزة

ملخص

تركيب البيانات و تصميم الخوارزميات

إعداد المهندس: رائد خضير



2020

(الوحدة الأولى)

تراكيب البيانات تناقش الطرق المختلفة لتنظيم البيانات داخل الحاسوب .

§ أسباب اختيار الطريقة المناسبة لتنظيم البيانات :

1. اختيار الطريقة المناسبة لتمثيل البيانات يؤدي إلى استخدام خوارزميات حل أكثر كفاءة .

البرنامج = تركيبة بيانات + خوارزمية الحل

2. اختيار الطريقة المناسبة لتمثيل البيانات يجعل البرنامج أيسر للكتابة حيث تكون خوارزمية الحل أوضح و أقرب إلى الواقع مما يؤدي إلى تقليل الوقت والجهد اللازمين لكتابة البرنامج .

3. وضوح ومنطقية طريقة الحل تؤدي إلى برامج واضحة يسهل فهمها وتعديلها عند الحاجة في وقت قصير .

§ البيانات و أنواعها Data And Data Types :

البيانات : هي مجموعة من القيم وظيفتها التعبير عن الكينونات التي نتعامل معها والأحداث التي نعيشها .

فالبيانات العددية عندما نتعامل مع الأرقام ، و البيانات الرمزية عندما نتعامل مع الحروف والكلمات ، و البيانات المنطقية عندما نتعامل مع بيانات تحتمل إجابات الصواب True و إجابات الخطأ False . فهذه جميعا يمكن أن نصطلح على تسميتها بإسم **أنماط بيانية بسيطة Simple data types**.

- المصطلح **مركب أو تركيب بياني Data Structure** يشير بشكل مباشر إلى مجموعة من عناصر البيانات التي لها تنظيم خاص .
- **ميزات التراكيب البيانية :**
  1. التعامل مع كل عنصر منها على حدة كما لو كان متغيرا مستقلا .
  2. طريقة تنظيم العناصر تؤثر مباشرة على طريقة الوصول إلى العنصر الواحد .

- **التراكيب البيانية البدائية Primitive :**

1) **الأعداد الصحيحة** (**Integer**) : يتم تخزين الأعداد الصحيحة في ذاكرة الحاسوب باستخدام إحدى طريقتين :
الأولى هي الطريقة الثنائية (Binary) ، و الثانية هي الطريقة العشرية (Decimal)
و من أساليب التخزين المتبعة لهذه الغاية طريقة المكمل الثنائي 2's Complement (هذا بالنسبة للطريقة الثنائية) .

أما الطريقة العشرية فإنها تستخدم الشيفرة العشرية المكونة من أربع خانات والتي تعرف بـــ Binary coded Decimal .

2) **الأعداد الحقيقة** (**Real**) : هي تلك الأعداد التي تشتمل على فاصلة عشرية .

3) **الرموز** (**Characters**) : هي الحروف و الأعداد من صفر إلى تسعة و العلامات الخاصة

4) **القيم المنطقية** (**Logical**) : إما True أو False .

§ أهم العمليات التي تجري عليها هي عملية الإسناد و العمليات البولية الثلاث المعروفة
NOT , OR , AND

§ ***ومن أهم طرق تمثيل القيم المنطقية*** :

a) استخدام خانة ثنائية واحدة (bit) حيث يعبر عن القيمة الإيجابية بالواحد ، والقيمة السلبية بالصفر .

b) استخدام الموضع التخزيني بكامله (WORD) ، يتم تخزين القيمة صفر في الموضع بكاملة للتعبير عن False ، و تعبئة الموضع بكامله بالقيمة واحد للتعبير عن True .

c) استخدام بايت واحد من الموضع Word . لأن الموضع Word قد يحتوي أكثر من بايت فيتم تخزين القيمة المنطقية في أصغر وحدة معنونة .

5) **مؤشرات الربط** (**Pointers**) : يقوم المترجم بحجز أماكن في الذاكرة للمتغيرات .

v التركيب المنطقي و الفيزيائي للبيانات Logical and physical structure of data :

- يتم التعامل مع فريقين :

الأول هو المبرمج والذي ينظر إلى البيانات من زاوية خاصة ، **والفريق الثاني هو الحاسوب** الذي يدرك معنى البيانات بصورة أخرى مختلفة ، فنظرة المبرمج إلى البيانات هي التصور المنطقي ، أما نظرة الجهاز هي التصور الفيزيائي .

- **إذا تجاوزنا اختلافات تمثيل البيانات في الذاكرة تبعا لنوع البيانات فإن نظرة الحاسوب إلى البيانات تكاد تكون واحدة مهما اختلفت أساليب التنظيم و مستوياته ، فالذاكرة مكونة من مجموعة من المواضع المعنونة تتبع بعضها بعضا على نسق واحد ، فهي أشبه ما تكون بمصفوفة ذات بعد واحد .**

v العلميات الأساسية التي تتم على التراكيب البيانية Data structure operations

- أهم العلميات التي تتم على التراكيب البيانية :

1) الاستقصاء Searching .

2) الاستعراض Traversal : ومن الحالات التي تقتضي القيام بمثل هذه العملية : **C** طباعة محتويات أحد التراكيب البيانية ، **A** و القيام بتعداد القيم البيانية المخزنة ، **B** و البحث عن أصغر أو أكبر قيمة .

3) الفرز Sorting .

4) الدمج Merging .

5) الإضافة و الحذف Insertion and Deletion .

6) التحديث Updating.

(الوحدة الثانية)

V مبادئ تحليل الخوارزميات :

t أهمية تحليل الخوارزميات :

الخوارزميات هي الخطوات اللازمة لحل مسألة ما ، وقد تكتب هذه الخوارزمية باللغة العربية أو الإنجليزية .

t المقاييس التي نستخدمها بالمقارنة بين الخوارزميات :

1ـ مقدار وقت الحاسوب اللازمة لتنفيذها .

2ـ مساحة ذاكرة الحاسوب التي تحتاج إليها الخوارزمية .

3ـ وضوح الخوارزمية وبساطتها .

t **وقت التنفيذ الحقيقي يعتمد على عوامل ليس لها علاقة بجودة الخوارزمية منها :**

1) **سرعة الحاسوب** : إذ أن الوقت الحقيقي يعتمد على سرعة الحاسوب المستخدم لتنفيذ البرنامج .

2) **كمية البيانات** : إذ أنه من الطبيعي أن يعتمد الوقت الحقيقي على كمية البيانات المراد معالجتها ، فكلما ازدادت كمية البيانات ازداد وقت التنفيذ .

3) **عوامل أخرى** لها علاقة بطريقة تنفيذ الخوارزمية .

t تحليل الخوارزمية رياضيا :

مثال : خوارزمية تطبع عناصر مصفوفة وتجد مجموع هذه العناصر :

```
sum : = 0 ;
for i = 1 to n do
begin
    sum : = sum + A [ i ] ;
   write ( sum ) ;
end ;
```

تستطيع التعبير عن عدد خطوات هذه الخوارزمية بدلالة اقتران رياضي بدلالة n ، فهناك خطوة واحدة قبل الدوران تنفذ مرة واحدة و خطوتان داخل الدوران تنفذان n من المرات ، لذا فإن عدد الخطوات اللازمة $T(n)$ لتنفيذ الخوارزمية : $T(n) = 1 + 2n$

ـ هناك طريقة رمزية (Notation) تدعى طريقة ترميز O الكبيرة لاستخلاص العوامل المهمة من التعبير عن كفاءة الخوارزمية .

مثال : استخدم طريقة ترميز O الكبيرة لتحليل الخوارزمية التالية :

```
for k: = 1 to N * N do
  begin
     writeln (k) ;
       for j: = 1 to N div 2 do
        begin
           m: = k * j ;
           writeln ( k:3 , j:3 , m ) ;
    end ;
end ;
```

الحل : الدورانين متداخلين ، فالعمليات الحرجة هي العمليات داخل الدورانين ، و لكون الدوران الداخلي يتكرر $N / 2$ و الدوران الخارجي يتكرر ^{2} من المرات فإن عدد مرات تنفيذ العمليات الحرجة هو $N / 2 * N^2$ ، و عليه فإن الخوارزمية هي $O (N^3)$.

t مقاييس تنفيذ مميزة و متكررة :

ـ أكثر هذه المقاييس شيوعا :

```
O ( 1 ) < O ( log n ) < O ( n ) < O ( n log n )
        < O ( n2 ) < O ( n3 ) < O ( 2n )
```

1ـ **الزمن الثابت** (Constant computing time) : زمن الحوسبة الثابت يشار إليه بالعلاقة (1) O

أي أن الوقت المطلوب للعملية لا يتغير .

2ـ **الزمن الخطي** (Liner time) : الخوارزمية التي يشار إليها بالعلاقة (n) O هي خوارزميه ذات زمن يتناسب طرديا مع حجم المشكلة ، أي مع أي زيادة تحدث على حجم البيانات .

3ـ **الزمن اللوغاريتمي** (Logarithmic time) : الخوارزمية ذات الكفاءة اللوغاريتمية و التي نشير إليها بالعلاقة (log n) O تقوم بعمل أكثر من الخوارزمية ذات الوقت الثابت ، وتحتاج وقتا أطول منها ولكنه أقل من الوقت الذي تستغرقه خوارزمية ذات وقت خطي .

(الوحدة الثالثة)

V التراكيب التجريدية :

- فوائد استخدام التراكيب التجريدية :

1. في حالة تغيير طريقة تمثيل و تنفيذ تركيبة بيانات معينة فإن البرامج التطبيقية الكبرى التي تستخدم تركيبة البيانات هذه لن تتأثر ، أي أنها لن تحتاج إلى عمل تعديلات عليها طالما أن العمليات المعرفة أصلا على تركيبة البيانات مازالت معرفه .
2. الاستخدام المتكرر للبرامج (Code Reuse) بدون الحاجة إلى كتابتها مرة أخرى .
3. تسهيل عمل فرق التصميم و البرمجة .

- التراكيب التجريدية :

هي تراكيب البيانات التي تستطيع استخدامها من خلال عمليات معينة معرفة على شكل إجراءات و اقترانات على هذه التراكيب ، من دون الحاجة إلى معرفة تفاصيل تمثيل هذه التراكيب . على سبيل المثال : الأرقام الصحيحة في لغة الباسكال هي نوع من أنواع البيانات التجريدية بالنسبة لنا ، إذ أننا نستخدم هذا النوع من خلال عمليات معرفة عليها مثل : الجمع و الضرب والقسمة و الطرح و بدون أن نضطر إلى معرفة كيفية تمثيل الأعداد الصحيحة فعليا .

- تصميم التراكيب التجريدية وتنفذيها :

لإنشاء تركيبه تجريدية جديدة يجب المرور بمراحل . في المرحلة الأولى هي مرحلة التعريف حيث يتم التعريف بدقة تركيبة البيانات الجديدة ، ونحدد مواصفاتها قبل أن نفكر بتفاصيل البناء / أما المرحلة الثانية هي مرحلة التمثيل حيث نحدد تفاصيل التمثيل المناسبة كاستخدام المصفوفات أو القوائم المتصلة .

- تركيبة البيانات لها ثلاث عوامل يجب أخذها بعين الاعتبار :

1) إعطاء وصف للعناصر المكونة لتركيبة البيانات .
2) إعطاء وصف للعلاقة بين العناصر .
3) إعطاء وصف للعمليات التي يتم تنفيذها على تركيبة البيانات .

v المجموعة باعتبارها تركيبة بيانات تجريدية (The Set Abstract Data Type) :

المجموعة هي عدد من الأشياء وغير المكررة المأخوذة من مدى (Universe) محدد من الأشياء ويحدد المدى بأصغر قيمة (First Value) و أكبر قيمة (Last Value) يمكن أن تنتمي إلى المجموعة . و لا يشترط أن تكون عناصر المجموعة مرتبة بأي ترتيب .

t العمليات التي تنفذ على المجموعات :

1- إنشاء المجموعة (SetCreate) : تنشأ كمجموعة خالية ويجب استخدام هذه التعليمة قبل استخدام المجموعة لأول مرة . وتحتاج في هذه المرحلة إلى كتابة ترويسة الإجراء الذي سينفذ هذه العملية ، و الترويسة هي :

```
Procedure SetCreate(var S:SetType)
```

2- الانتماء إلى المجموعة (IsElementof) : تقر فيما إذا كان عنصر ما منتميا إلى المجموعة S أم لا ، و الترويسة هي :

```
function IsElementOf(S:SetType;E:Universe):boolean;
```

3- تعيين المجموعات (SetAssign) : تعين هذه العملية قيمة المجموعة S إلى مجموعة أخرى T أي أن نجعلهما متساويتين . و الترويسة هي :

```
Procedure SetAssign(S:SetType,var T:SetType);
```

4- التأكد من أن المجموعة خالية (SetEmpty) : تستخدم لفحص مجموعة E هل هي خالية أم لا . الاقتران البولياني يعيد القيمة True إذا كانت المجموعة خالية ، و False إذا لم تكن خالية . و الترويسة هي :

```
function SetEmpty(S:SetType):boolean;
```

5- التأكد من مساواة المجموعات (SetEqual) : تستخدم لفحص فيما إذا كانت المجموعتان S , T متساويتين . و الترويسة هي :

```
function SetEqual(S,T:SetType):boolean;
```

6- فحص المجموعة الجزئية (SubSetof) : تستخدم لفحص فيما إذا كانت المجموعة S هي مجموعة جزئية في المجموعة T ، والترويسة هي :

```
function SubSetOf(S,T:SetType):boolean;
```

7- اتحاد المجموعات (Union) : تستخدم لإيجاد المجموعة T وهي مساوية لإتحاد المجموعتين S , R والترويسة هي :

```
procedure Union(R,S:SetType; var T : SetType);
```

8- تقاطع المجموعات (Intersection) : تستخدم لإيجاد المجموعة T وهي مكونة من العناصر الموجودة في كل من المجموعتين S , R والترويسة هي :

```
procedure Intersection(R,S:SetType ; var T : SetType);
```

9- فرق المجموعات (Difference) : تستخدم لإيجاد المجموعة T المكونة من العناصر الموجودة في المجموعة R وليست موجودة في المجموعة S والترويسة هي :

```
procedure Difference(R,S:SetType ; var T : SetType );
```

10- إضافة عنصر إلى المجموعة (AddElement) : تستخدم لإضافة العنصر U من المدى Universe إلى المجموعة S ، والترويسة هي :

```
procedure AddElement( var S:SetType ; U : Universe );
```

11- إزالة عنصر من المجموعة (RemoveElement) : تستخدم لإزالة العنصر U من المجموعة S ، والترويسة هي :

```
procedure RemoveElement( var S:SetType ; U : Universe );
```

12- طباعة عناصر المجموعة (SetDisplay) : يقوم بعرض العناصر المنتمية إلى المجموعة على شاشة العرض ، والترويسة هي :

```
procedure SetDisplay ( S:SetType );
```

t أمثلة على المجموعات :

1) إنشاء المجموعة (SetCreat) : إنشاء مجموعة خالية و تخزن القيمة False في كل موقع من مواقع المصفوفة :

```
procedure SetCreat(var S:SetType);
 var I: Universe ;
 begin
  for I:= FirstValue to Lastvalue do
      S[I] := false
 end ; {SetCreate}
```

2) الانتماء إلى المجموعة (IsElementOf) : إعادة القيمة البوليانية True إذا كان العنصر E منتميا إلى المجموعة S و القيمة False إذا لم يكون عنصرا في المجموعة :

```
procedure IsElementOf (s:SetType;E:universe):boolean;
 begin
  IsElementOf := S[E];
 end ; {iselement}
```

3) تعيين المجموعات (SetAssign) : أكتب برنامج لجعل قيمة المجموعة T مساوية لقيمة المجوعة S :

```
procedure SetAssign (S:SetType;var T:SetType);
 var
  I : Universe ;
 begin
  for I := FirstValue to LastValue do
      T[I]:= S[I];
end ; {SetAssign}
```

4) التأكد من أن المجموعة خالية (SetEmpty) : تكون المجموعة خالية إذا كانت كل القيم المخزنة في المصفوفة هي false و إلا فإنها لا تكون خالية ، و عليه يجب أن يعيد الاقتران القيمة false :

```
function SetEmpty(S:SetType):boolean;
 var
  I:Universe;
 begin
  SetEmpty:=true;
  for I:=FirstValue to LastValue do
      if S[I]=true then
          SetEmpty:=false;
  end;{SetEmpty}
```

5) التأكد من مساواة المجموعات (SetEqual) : أكتب برنامج لاقتران بولياني لفحص إذا كانت المجموعتان S , T متساويتين :

```
function SetEqual (S,T:SetType):boolean;
 var
  I:Universe;
 begin
  SetEqual:=true;
  for I:=FirstValue to LastValue do
      if S[I]<>T[I] then SetEqual:=false;
  end;{SetEqual}
```

6) فحص المجموعة الجزئية (SubsetOf): تكون المجموعة S مجموعة جزئية من T إذا كان كل موقع في المصفوفة الممثلة للمجموعة S يحتوي على القيمة true أيضا :

```
function SubsetOf(S,T:SetType):boolean;
 var
  I:Universe;
 begin
  SubsetOf:=true;
  for I:=FristValue to LastValue do
      if S[I] and not T[I] then
          SubsetOf:=false;
end;
```

7) اتحاد المجموعات (Union) : لإيجاد المجموعة T التي هي حاصل اتحاد المجموعتين S,R علينا أن نضع بها كل عنصر موجود في المجموعة R أو موجود في المجموعة S :

```
procedure Union(R,S:SetType;var T:SetType);
 var
  I:Universe;
 begin
  for I:=FirstValue to LastValue do
      T[I]:=R[I]or S[I];
end;{Union}
```

8) تقاطع المجموعات (Intesection).

9) فرق المجموعات (Difference).

10) إضافة عنصر إلى المجموعة (AddElement).

11) حذف عنصر من المجموعة (RemoveElement).

12) طباعة عناصر المجموعة (SetDisplay).

(الوحدة الرابعة)

v السلاسل الرمزية :

t السلاسل الرمزية باعتبارها تراكيب تجريدية Strings As Abstract Data Type :

السلسلة الرمزية String هي سلسلة من الرموز Characters مرتبه بطريقة خطية حيث أن لكل عنصر ترتيبا معينا .

t أمثلة على السلاسل الرمزية : ' Ali Ahmad '

' AbGt678&*( )# '

t العمليات المرتبطة بالسلاسل الرمزية :

1- إنشاء السلسلة الرمزية (StringCreate) : يستخدم قبل استخدام السلسلة لأول مرة بهدف تهيئة السلسلة الرمزية للإستخدام بطريقة صحيحة ، وطبيعة الخطوات التي يقوم بها تعتمد إلى حد كبير على طريقة تمثيل السلسلة الرمزية .

2- قراءة السلسلة الرمزية (ReadString) : إجراء يقوم بقراءة السلسلة الرمزية S من جهاز الإدخال . و الترويسة هي :

```
Procedure ReadString (Var S:StringType)
```

3- كتابة السلسلة الرمزية (WriteString) : إجراء يقوم بكتابة السلسة الرمزية S على جهاز الإخراج . و الترويسة هي :

```
Procedure WriteString (S:StringType)
```

4- إسناد السلاسل الرمزية (StringAssign) : عملية إسناد سلسلة رمزية S إلى متغير T وهو متغير سلسلي رمزي ، بحيث تصبح تلك السلسلة قيمة لذلك المتغير وللإجراء الترويسة التالية :

```
Procedure StringAssign(S:StringType;Var T:StringType)
```

5- إيجاد طول السلسلة الرمزية (StringLength) : وهي عملية تحسب عدد الرموز في السلسة S وهي دالة (اقتران) لها الترويسة التالية :

```
Function Length (S:StringType):StringLengthRange;
```

6- وصل السلاسل الرمزية (Concatenate) : عملية تستخدم لوصل سلسلتين بحيث تصبحا سلسلة واحدة تتكون من رموز السلسلة الأولى S مضافا إليها رموز السلسلة الرمزية T ، و الترويسة هي :

```
Procedure concatenate(Var S:StringType;T:StringType)
```

مثال : إذا كانت قيمة *First* هي *' Ali '* وقيمة *Second* هي *' Ahmad '* فإن نتيجة استدعاء *concatenate* هي :

```
Procedure concatenate(First,Second)
First= ' Ahmad Ali '
```

7- استخراج قطع من سلسلة رمزية (Substring) : هي عملية تستخدم لاستخراج نسخة عن مقطع في سلسلة رمزية ، ولعمل ذلك تحتاج إلى تحديد موقع أول حرف في المقطع START وعدد أحرف المقطع Len ، و الترويسة هي :

```
Procedure Substring (S:StringType Start: StringRange;
    Len: StringLengthRange ; Var T:String Type);
```

8- البحث عن سلسلة رمزية في سلسلة أخرى (StringSearch) : هي عملية تحتاج إليها لمعرفة فيما إذ كانت سلسلة رمزية معينة Sub تظهر في سلسلة أخرى Master كسلسلة جزئية ، وتحدد هذه العملية مكان وجود هذه السلسلة Sub في السلسلة الكلية Master وتعيد لنا رقم هذا المكان ، وإذا لم تظهر Sub في Master فإن القيمة المعادة هي 0 للإشارة إلى عدم وجود Sub في Master ، و الترويسة هي :

```
Function StringSearch ( Master,Sub:StringType;
      Start:StringRange):StringLengthRnge;
```

9- إدخال سلسلة رمزية في سلسلة أخرى (StringInsert) : وتستخدم هذه العملية عند الحاجة إلى إدخال سلسلة رمزية T في أخرى S ، بحيث تصبح سلسلة جزئية فيها ، وعلى المستخدم أن يحدد مكان الإدخال Place ، و الترويسة هي :

```
Procedure StringInsert (Var S:StringType;T:StringType;
Place : StringRange);
```

10- حذف مقطع من سلسلة رمزية (StringDelete) : وتستخدم لحذف مقطع من سلسلة رمزية ، وعلى المستخدم أن يحدد عند استدعاء هذه العملية الموقع START الذي يبتدئ عنده المقطع و عدد أحرف ذلك المقطع Number ، و الترويسة هي :

```
Procedure StringDelete(Var S:StringTypeStart StringRange;
Number:StringLengthRange);
```

11- فحص فيما إذا كانت سلسلتان رمزيتان متساويتين (StringEqual) : تعيد هذه العملية القيمة true إذا كانت السلسلتان S و T متساويتين ، وتعيد false إذا لم تكونا متساويتين . و الترويسة هي :

```
Function StringEqual (S, T:StringType):boolean ;
```

12- فحص فيما إذا كانت سلسلة رمزية تسبق أخرى حسب الترتيب الأبجدي : تعيد true إذا كانت السلسلة تسبق السلسلة T حسب الترتيب الأبجدي للحروف ، وحسب قيم ASCII لجميع الرموز ، و الترويسة هي :

```
LessThan
Function LessThan (S, T:StringType):boolean;
```

13- فحص فيما إذا كانت سلسلة رمزية تأتي بعد أخرى (GreaterThan) : وتعيد لنا هذه العملية true فيما إذا كانت السلسلة S تلي السلسلة T حسب الترتيب الأبجدي و الترويسة هي :

```
Function GreaterThan (S, T:String): boolean;
```

t تنفيذ عمليات السلالة الرمزية باستخدام أسلوب الطول الثابت :

1) إنشاء سلسلة رمزية StringCreate : يجعل طول السلسلة صفرا ، و الترويسة :

```
procedure StringCreat(var S:SetType);
begin
   S.Len:=0;
end ; {StringCreate}
```

2) قراءة السلسة الرمزية ReadString :

```
procedure ReadString ( var S:StringType);
var
  i:StringLengthRange ;
begin
  i:=0;
  while not eoln and (I<=MaxString)do
      begin
  i:=I+1
  read(s.data[i]);
      end;{while}
readln;
      s.Length:=I;
end; {ReadString}
```

يقوم هذا الإجراء بقراءة السلسلة من سطر واحد لذلك فهو يحتوي على دوران يتوقف إذا انتهى السطر المدخل أو إذا إزداد عدد الرموز المدخلة عند أكبر عدد مقبول MaxString و داخل الدوران نقرأ السلسلة رمزا رمزا ، وتخزن في المصفوفة الرمزية data وتزاداد قيمة العداد الذي يعد عدد الرموز المقروءة ، وعند انتهاء الدوران يخزن عدد الرموز المقروءة i في الحقل Length .

3) كتابة السلسلة الرمزية WriteString :

```
procedure WriteString (S: StringType);
   begin
      Wrtieln (S.data);
end;
```

4) إسناد السلاسل الرمزية StringAssign :

```
procedure StingAssign (S:StringType; Var T:StringType);
Var
     i = StringRange ;
begin
     for I = 1 to MaxString do
          T.data [I]: S.data [I];
     T.length: S.Length;
 end ; {StringAssign}
```

إجراء يقوم بنسخ المصفوفة S.data في T.data عنصرا عنصرا . ثم يجعل طول السلسلة T مساويا لطول السلسلة S .

5) إيجاد طول السلسلة الرمزية Length :

```
Function Length (S:StringType): StringLengthRange;
begin;
     length :=S.Length;
```

6) وصل السلاسل الرمزية Concatenate :

```
Procedure concatenate (Var S:StringType;T:StringType);
Var
     i:StringRange;
begin
     for I:=to T.length do
          S.data [I+S.Length]:T.data[I];
          S.Lenght:=S.Length+T.Length;
end;
```

يقوم بنسخ السلسة T في S إبتداء من آخر موقع في S ، ثم يقوم بتعديل طول السلسلة S بحيث يصبح مساويا للطول القديم زائد طول السلسلة T .

7) استخراج مقطع من سلسلة رمزية SubString :

```
Procedure substring (var S:StringType;Start:StartingRange;
                     len:StringLengthRange;
                     var T:StartingType);
Var
     i:integer;
begin
     for i:=1 to len do
          T.data[i]:=S.data [Start+i-1];
     T.length:=len;
end;
```

8) البحث عن سلسلة رمزية في سلسلة أخرى StringSearch :

```
Function StringSearch (Master,Sub:StringType;
                       Start:StringRange):
                                 StringLengthRange;
Var
      i,j:integer;
      found:boolean;
begin
      i=Start;
      found:=false;
      While (I<=Master.length) and not found do
           begin
           j:=1;
             While (Master.data[I+j-1]=Sub.data[j]) and
                     not found
                  do
                     if j= Sub.length then
                               found:true
                     else
                          j:=j+1;
                if not found then I:=I+1;
           end; {While}
if found then
      StringSearch:=I
else
      StringSearch:=0;
end; {StringSearch}
```

يقوم هذا الإجراء بمقارنة عناصر السلسلة Sub بعناصر السلسلة Master ابتداءً بالموقع Start في Master . فإن توافقت عناصرهما جميعا يتوقف الدوران ، ويعيد الإجراء موقع Sub في Master . و إن لم يتوافقا تعاد الكرة ولكن ابتداء من الموقع Sub+1 وهكذا ، حتى يتوافقا أو نتأكد من أن Sub لا يمكن أن تكون سلسلة جزئية في Master .

9) إدخال سلسلة رمزية في سلسلة أخرى StringInsert :

```
Procedure StringInsert (Var S:StringType;T:StringType ;
                        Place:StringRange;
Var
     i:integer;
begin
if(place<=S.Length)and
        (S.length+T.length<=Maxstring)then
     begin
         for i:S.legth+T.length downto place do
              S.data[i]:S.data[I-T.length];
         for i:=1 to T.length do
              S.data[Place+I-1]:=T.data[I];
         S.length:=S.length+T.length;
     end;{if}
end;{StringInsert}
```

يقوم هذا الإجراء بالتأكد أولا من أن عملية الإدخال عملية صحيحة بمعنى أن مكان الإدخال (أقل من طول S) وأن ناتج عملية الإدخال سيبقى سلسلة رمزية ذات طول مناسب أي يمكن أن تخزن في المصفوفة الرمزية ذات الطول Maxstring . إن لم يتحقق أي من هذين الشرطين فإن عملية الإدخال غير صحيحة ولا يتم تغيير أي شيء .

أما إن تحقق الشرطان فإن هناك ثلاث خطوات رئيسية يجب أن تنفذ :

أولا: يجب عمل فراغ في السلسلة S لوضع السلسلة T وذلك بإزاحة الرموز الموجودة في الموقع Place في S إلى اليمين بمقدار طول السلسلة T وهذا يتم في دوران for الأول .

ثانيا : يتم نسخ عناصر السلسلة T رمزا رمزا إلى السلسلة S . وهذا ما يتم من خلال دوران for الثاني .

ثالثا: يجب أن يتغير طول السلسلة S بحيث تصبح مساوية لطول السلسلة بعد الإضافة وهو الطول القديم زائد طول السلسلة T .

10) حذف مقطع من سلسلة رمزية StringDelete :

```
 Procedure StringDelete (Var S:StringType;Start:StringRange;
                                    Number:StringRange);
var
   i:integer;
begin
 if(Start<=S.length)and (Start+Number <= S.length+1)
then
      {the operation is legal}
   begin
     for i:=Start to MaxString-Number do
         S.data[i]:=S.data[i+Number];
     S.length:=S.length – Number;
   end;
end; {StringDelete}
```

11) فحص فيما إذا كانت سلسلتان رمزيتان متساويتين StringEqual :

```
Function StringEqual (S,T:StringType):boolean;
begin
      StringEqual:=S.data=T.data;
end;{Stringequal}
```

12) فحص فيما إذا كانت سلسلة رمزية تسبق أخرى حسب الترتيب الأبجدي :

```
Function LessThan(S,T:StringType):boolean;
begin
      LessThan:= S.data < T.data ;
end;
```

13) فحص فيما إذا كانت سلسلة رمزية تأتي بعد سلسلة أخرى :

```
Function GreaterThan (S,T:StringType):boolean;
begin
      GreaterThan:= S.data > T.data ;
end;
```

(الوحدة الخامسة)

٧ المكدسات Stacks :

- المبدأ العام الذي يقوم عليه المكدس هو أن الإضافة و الحذف يجريان عند طرف واحد فالمكدس ما هو إلا قائمة من القيم مخزنة بأسلوب معين من أساليب تمثيل القوائم و تعمل وفق مبدأ "من يأتي آخرا يغادر أولاً " . ومعنى ذلك أن آخر قيمة تضاف إلى المكدس هي أول قيمة تحذف منه ، ويوصف هذا بالكلمة (LIFO) Last In First Out ، و أهم خصائص المكدس : ـ

1. يتسم المكدس بالديناميكية و الحركة .
2. يوفر المكدس إمكانية الوصول المباشر ، إلا أن هذا الوصول محدود بعنصر واحد وهو قيمة المكدس ، كما أن الوصول إلى العناصر الأخرى مرهون بعمليات الإضافة والحذف push and pop ، فكلما زادت عمليات الحذف اقترب العنصر من القمة ، وكلما زادت عمليات الإضافة ابتعد العنصر عن القمة .
3. إذا تساوت عمليات الإضافة وعمليات الحذف فإن المكدس يصل إلى وضع يصبح معه خاليا من القيم .
4. طبيعة عمل المكدس لا تستلزم وجود أكثر من مؤشر واحد للإشارة إلى مقدمة المكدس Top .

- تمثيل المكدسات في الذاكرة يمكن أن يستخدم أحد الأسلوبين :

ـ **الأول** هو التمثيل التتابعي .

ـ **و الثاني** هو التمثيل المتصل .

t العمليات الخاصة بالمكدسات :

1. إنشاء مكدس StackCreate :

```
Procedure StackCreate (var S:Stack);
```

2. فحص حالة الامتلاء أو الفائض StackFull :

```
Function StackFull (S:Stack):Boolean ;
```

3. فحص إذا كان المكدس خاليا StackEmpty :

```
Function StackEmpty (S:Stack):Boolean;
```

4. عملية الإضافة Push :

```
Procedure Push (var S : Stack ; var Element : StackData);
```

5. عملية الحذف Pop :

```
Procedure Pop (var s : Stack ; var Element : StackData);
```

t تنفيذ العمليات الخاصة بالمكدسات :

1) إنشاء المكدس StackCreate :

```
Procedure stackCreate (var s:stakc)
    begin
        S.Top:=0
    end;
```

2) فحص حالة الفائض (الامتلاء) StackFull :

```
Function StackFull (S:Stack):boolean;
    begin
        S.Top:=0
    end;
```

3) فحص إذا كان المكدس خاليا StackEmpty :

```
Function StackEmpty (S:Stack):boolean;
    begin
       StackEmpty:=(S.Top=0)
    end;
```

4) عملية الإضافة Push :

```
Procedure Push (var S:Stack ; Element : StackData);
     begin (*Push*)
          If S.Top=N Then
             writeln ('the stack is full')
          else begin
               S.Top:=S.Top+1;
               S.Data [S.Top]:=element;
          end;
     end;
```

5) عملية الحذف Pop :

```
Procedure pop (var S:Stack ; var element : StackData );
     begin (*pop*)
          If S.Top=0 then
               Writeln ('Stack is empty')
          Else begin
               Element := S.Data[S.Top] ;
               S.Top:= S.Top – 1 ;
          end;
     end;
```