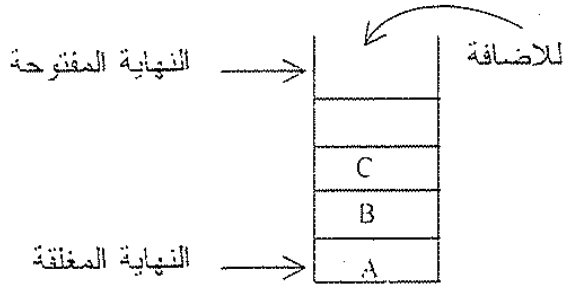
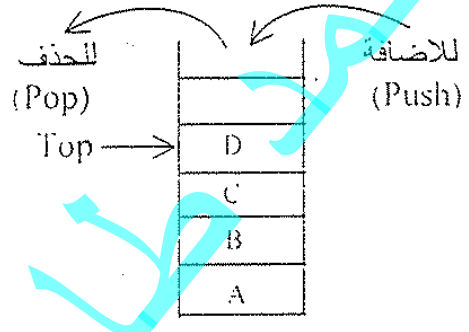
2-3 المكدس Stack

هو عبارة عن قائمة خطية تتم فيها عمليتي الاضافة والحذف من احدى نهايتي القائمة وتكون النهاية الاخرى مغلقة.



الشكل (3 – 1)

لنأخذ المكدس الموضح في الشكل اذ نجده يحتوي على العناصر A.B.C وعند اضافة عنصر جديد مثل (D) يجب ان تكون الاضافة من الجهة المفتوحة ليصبح الشكل كالاتي :



وعند حذف عنصر من المكدس يجب ان تستخدم نفس الجهة المفتوحة فقط، أي نستطيع ان ناخذ العنصر (D) ثم ناخذ العنصر (C) بالتتابع ولا نستطيع ان ناخذ العنصر (C) قبل ان ناخذ العنصر (D) ، مع ملاحظة ان العنصر (D) دخل اخيرا.

ولهذا نستطيع ان نلخص عمل المكدس بالعبارة الاتية :

(اخر من يدخل اول من يخرج) Last In First Out (LIFO)

كما انه لايمكن اخذ (حذف) عنصر من وسط عناصر المكدس الا بعد حذف (اخراج) العناصر التي تسبقه من جهة النهاية المفتوحة مع التاكيد على ان النهاية الاخرى مغلقة ولا تستخدم ابدا.

غادة محمد طاهر قادم

وتسمى عملية الاضافة الى المكدس (push) او (Insertion) وعملية الحذف من المكدس (pop) او (Deletion) .

مثال :

نفرض (S) تعني (Stacking) أي ترمز لعملية اضافة عنصر الى المكدس و (U) تعني (Unstacking) أي ترمز لعملية حذف عنصر من المكدس وكانت مجموعة المدخلات للمكدس بالترتيب R.N.Y.B.M, بين ما هي المخرجات بعد تنفيذ كل سلسلة من العمليات التالية :

أ / SSUUSUSUSU

ب / SSSUSUUSUU

الحل :

يقصد بترتيب المدخلات انه عند تنفيذ عملية ادخال عنصر الى المكدس فان اختيار العنصر يكون من تلك المدخلات بالتتابع أي ناخذ M اولا ثم B ، ... وهكذا، ولانستطيع اخذ العنصر N قبل العناصر السابقة له .

أ -

المدخلات	→	M	B			Y		N		R	
سلسلة العمليات	→	S	S	U	U	S	U	S	U	S	U
المخرجات	→			B	M		Y		N		R

ب -

المدخلات	→	M	B	Y		N			R		
سلسلة العمليات	→	S	S	S	U	S	U	U	S	U	U
المخرجات	→				Y		N	B		R	M

بإشراف محمد طاهر قاسم

مثال :

اذا كانت مجموعة مدخلات المكدس بترتيب 5.4.3.2.1 بين أيا من المخرجات المبينة ادناه صحيحة وفق اسلوب عمل المكدس ..

أ- 2 4 5 3 1

ب- 4 2 3 1 5

جـ- 4 5 1 2 3

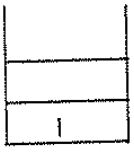
ء- 4 3 5 2 1

الحل :

الفرع أ: المخرجات المطلوبة (2.4.5.3.1)

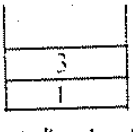
لاخراج العنصر (2) يجب اولا ادخال العنصرين 2.1 أي ان تسلسل تنفيذ العمليات هو SSU

أي ان محتويات المكدس تصبح



ولاخراج العنصر (4) بعد العنصر (2) يجب إدخال العنصرين 4.3 أي ان تسلسل تنفيذ العمليات في هذه الحالة هو SSUSSU وتصبح محتويات المكدس :

ولاخراج العنصر (5) بعد العنصر (4) يجب ادخاله اولا ثم اخراجه أي ان تسلسل تنفيذ العمليات يكون SSUSSUSU وتصبح محتويات المكدس :



وفق حالة المكدس الحالية يمكن اخراج العنصرين 1.3 بالتتابع أي ان تسلسل تنفيذ العمليات هو SSUSSUSUUU .

اذن يمكن الحصول على مثل هذه المخرجات اذا كان تسلسل العمليات بالصيغة الاخيرة مع الالتزام بترتيب المدخلات .

غادة محمد طاهر قاسم

الفرع ب : المخرجات المطلوبة (4. 2.3.1. 5)

لاخراج العنصر (4) يجب اولا ادخال العناصر 4.3.2.1 وفق سلسلة العمليات SSSSU وتصبح محتويات المكدس :

ولاخراج العنصر (2) من المكدس بحالته الحالية يجب اخراج العنصر (3) قبله لذا فان هذا التسلسل من المخرجات (4.2.3.1. 5) لا يمكن تنفيذه .

الفرع ج: المخرجات المطلوبة (4.5.1.2.3)

يمكن اخراج العنصرين 4.5 بعد تنفيذ سلسلة العمليات الاتية :

المدخلات	→	1	2	3	4		5	
سلسلة العمليات	→	S	S	S	S	U	S	U
المخرجات	→					4		5

وستصبح محتويات المكدس :

وهنا سيتعذر اخراج العنصر (1) قبل العنصرين (2.3) لذا فان تسلسل المخرجات (4.5.1.2.3) غير صحيح .

الفرع د : المخرجات المطلوبة (4.3.5.2.1)

يمكن الحصول على هذه المخرجات عند تنفيذ عمليات الادخال والاخراج بالتسلسل الآتي :

المدخلات	→	1	2	3	4			5		
سلسلة العمليات	→	S	S	S	S	U	U	S	U	U	U
المخرجات	→					4	3		5	2	1

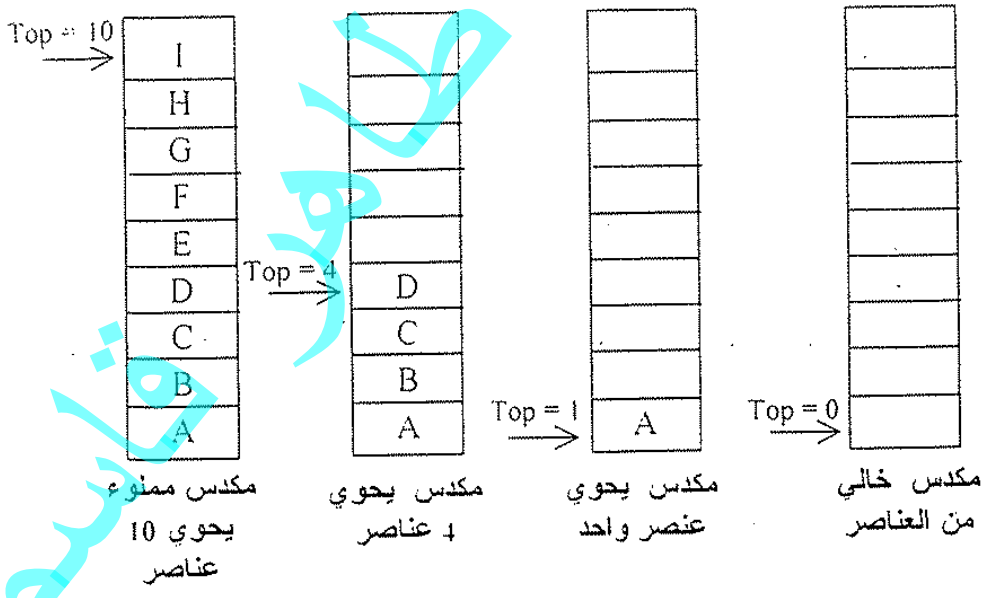
غادة محمد طاهر قاسم

3 – 2 – 1 تمثيل المكدس باستخدام المصفوفة

Array Representation of Stack

يمكن تطبيق المكدس باستخدام مصفوفة احادية بالسعة المطلوبة (size) وبالنوع المناسب للبيانات (Data Type) التي ستخزن فيه (integer , Real ... الخ) مع استخدام متغير مستقل يدعى (Top) يستعمل كموشر يشير الى موقع اعلى عنصر في المكدس (موقع اقرب عنصر الى النهاية المفتوحة) وابتداء تكون قيمة المؤشر (Top = 0) عندما يكون المكدس خاليا من العناصر ، ويعرف المكدس برمجيا كالأتي:-

```
Const   Size = 10 ;  { or  any other value }
Type    Stackelement = integer  ; {  or any other type }
        ST = array [ 1 .. size ]  of  stackelement :
Var     stack : ST ;
        Top : integer ;
```



الشكل (3 – 3)

غادة محمد طاهر قاسم

عملية الاضافة للمكدس (Push)

لتنفيذ عملية الاضافة بشكل صحيح نتبع الخطوات الآتية :

1- التحقق من كون المكدس غير مملوء (not full) أي ان المؤشر (Top <> Size) لتجنب حالة الفيض (over flow) وتعذر تنفيذ عملية الاضافة .

2- تحديث قيمة المؤشر Top = Top + 1 ليشير الى الموقع التالي .

3- اضافة العنصر الجديد في الموقع الجديد Stack [Top]

عملية الحذف من المكدس (Pop)

ان تنفيذ عملية حذف أي عنصر من المكدس يجب ان تكون وفق الخطوات الآتية:

1- التحقق بأن المكدس غير خال (not Empty) أي ان المؤشر Top<>0 لتجنب حالة الفيض (under flow) وتعذر تنفيذ عملية الحذف .

2- اخذ العنصر من الموقع الذي يشير اليه (Top) وخزنه وقتيا في متغير مستقل Item = Stack [Top]

3- تحديث قيمة المؤشر Top = Top – 1 ليشير الى موقع العنصر التالي للعنصر الذي حذف .

ملاحظة :

يتضح اعلاه ان الخطوتين 2 ، 3 في عملية الحذف معكوسة الترتيب عنها في عملية الاضافة .

3 – 2 – 2 خوارزميات المكدس Stack's Algorithms

يمكن تصميم مجموعة من الخوارزميات لتغطية فعاليات المكدس ومن ثم برمجتها لتمثيلها عمليا .

1- خوارزمية الاضافة Push Algorithm

```
If      Stack is full
Then    Overflow ← True
Else
        Overflow ← false
        Top ← Top + 1
        Stack [ Top ] ← New element
```

2- خوارزمية الحذف POP Algorithm

If Stack is Empty
Then under flow ⟵ True
Else
under flow ⟵ false
element ⟵ stack [Top]
Top ⟵ Top - 1

3- خوارزمية ملء المكدس Stack full

هذه الخوارزمية للتحقق من هل المكدس مملوء ام لا اعتمادا على قيمة المؤشر (Top) قبل عمليات الاضافة

If Top ⟵ size
Then stack full ⟵ True

4- خوارزمية خلو المكدس Stack Empty

هذه الخوارزمية للتحقق من هل ان المكدس خال ام لا اعتمادا على قيمة المؤشر (Top) قبل عمليات الحذف

If Top = 0
Then stack empty ⟵ True

5- خوارزمية اخلاء المكدس ClearStack

هذه الخوارزمية تستخدم لغرض تهيئة المكدس واخلائه من العناصر بجعل قيمة المؤشر (Top = 0)

Top = 0

3 – 2 – 3 البرامج الفرعية لتنفيذ عمليات المكدس Stack's Procedures and Functions

ان تصميم برامج فرعية (procedures . functions) لكل فعالية او عملية من عمليات المكدس تساعد على تبسيط وتوضيح كيفية برمجة تلك العمليات ومن ثم تجميعها في برنامج واحد تتوفر فيه صفات البرمجة المهيكلة ويكون واضحا للقراءة وسهل الفهم والمتابعة والتحديث والتطوير .

غادة محمد طاهر وفيق

ونفترض وجود التعريف التالي في مقدمة البرنامج لتكون البرامج الفرعية اللاحقة صحيحة

```
Const    Size  = 20 : { or any other integer value }
Type     Stackelement = integer ;  { or any other type }
         St : array [ 1 .. Size ] of  stackelement  ;
Var      Stack : ST  ;
         Top : integer ;
         Item : stackelement  :
```

1- برنامج فرعي لاخلاء المكدس

```
Procedure clearstack ( VAR  Top : integer )
Begin
      Top : = 0
End
```

لاحظ عدم الحاجة للمرور على جميع مواقع المصفوفة وجعلها مساوية للصفر والاكتفاء فقط بجعل المؤشر (Top = 0) و هذا البرنامج الفرعي يستدعى في بداية العمل لجعل المكدس خاليا .

2- برنامج فرعي للتحقق من امتلاء المكدس

```
Function FullStack ( Top : integer ) : Boolean
Begin
      If      Top = Size
      Then    FullStack : = True
      Else    FullStack : = False
End
```

هذه الدالة (Function) مدخلها المؤشر (Top) وبموجب قيمته فأن المخرج هو المتغير المنطقي (FullStack) اذ تكون قيمته اما (True) عندما يكون المكدس مملوء وتكون قيمته (False) عندما يكون المكدس غير مملوء .
والبرنامج الفرعي (FullStack) يستدعى داخل البرنامج الفرعي (procedure push) لينفذ عملية الاضافة .

غادة محمد طاهر وليد

3- برنامج فرعي للتحقق من خلو المكدس

```
Function EmptyStack ( Top : integer ) : Boolean
Begin
      If      Top = 0
      Then    EmptyStack : = True
      Else    EmptyStack : = False
End
```

هذه الدالة مدخلها المؤشر (Top) وبموجب قيمته فأن المخرج هو المتغير المنطقي (EmptyStack) وقيمته (True) عندما يكون المكدس خاليا و(False) عندما يكون المكدس غير خالي وهذا والبرنامج الفرعي (EmptyStack) يستدعى داخل البرنامج الفرعي (procedure pop) الذي ينفذ عملية الحذف .

4- برنامج فرعي لأضافة عنصر واحد الى المكدس

```
Procedure Push ( Var Stack : ST : Var Top : integer :
                                  Var Item : stackelement) :
Begin
      If Fullstack ( Top)
      Then writeln ( ' Error ... the Stack is Full ' )
      Else
          Begin
              Top : = Top + 1 :
              Stack [ Top] : = Item
          End
End
```

هذا البرنامج الفرعي يضيف عنصر واحد (Item) للمكدس ويمكن استدعائه في البرنامج الرئيسي (main Program) باي عدد من المرات باستخدام احد ايعازات التكرار مثل (For ... Do) الذي يتضمن قراءة العنصر (Item) ثم استدعاء البرنامج الفرعي (push) لاضافته الى المكدس .

ان هذه الصيغة تسمح باستدعائه في البرنامج الرئيسي في اكثر من موقع وعدة مرات .

غادة محمد طاهر قاسم

5- برنامج فرعي لحذف عنصر واحد من المكدس

```
Procedure POP (Stack : ST ; Var Top :integer ,
                                Var  Item: Stackelement ) ;
Begin
      If     Emptystack ( Top)
      Then writeln ( ' Error … the Stack is Empty ' )
      Else
          Begin
              Item : = Stack [ Top ] ;
              Top : = Top - 1
          End
End
```

هذا البرنامج الفرعي يأخذ العنصر الذي يشير اليه (Top) وينسخه في المتغير (Item) لاستخدامه لاحقا بمعالجة معينة لتحقيق الغرض الذي من اجله سحب هذا العنصر من المكدس .

ولغرض حذف او سحب اكثر من عنصر من المكدس بصورة متتابعة فان هذا البرنامج يستدعى باي عدد من المرات وفي أي موقع من البرنامج الرئيسي .

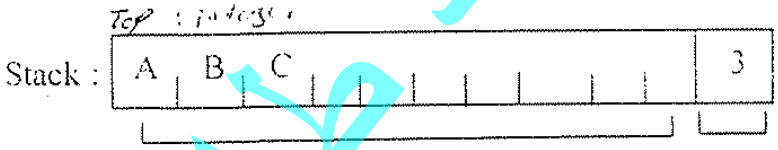
غادة محمد طاهر قاسم

4 – 2 – 3 تطبيق المكدس باستخدام القيد

Record Implementation of Stack

في التطبيق السابق باستخدام المصفوفة ورد تعريف المؤشر (Top) كمتغير مستقل عن المصفوفة التي تمثل المكدس ، الا اننا هنا نستخدم القيد (Record) في تمثيلها معا كهيكل بياني واحد حيث يتكون القيد من جزأين الاول يمثل المكدس وهو على شكل مصفوفة والجزء الثاني هو حقل يمثل المؤشر (Top) ويعرف في لغة باسكال بالطريقة التالية :

```
Const    Size = 20 ;    { or any other value }
Type     Stelement = integer ; { or any other type }
         St = Record
         elements : array [ 1 .. size ] of Stelement ;
         Top : 0 ... size
         End ;
Var      Stack : ST ;
         Item : Stelement ;
```



elements هذا الجزء هو Top

stack . elements ويستخدم باسم يستخدم باسم stack .Top

لاضافة عنصر جديد لهذا المكدس نتبع الخطوات التالية :

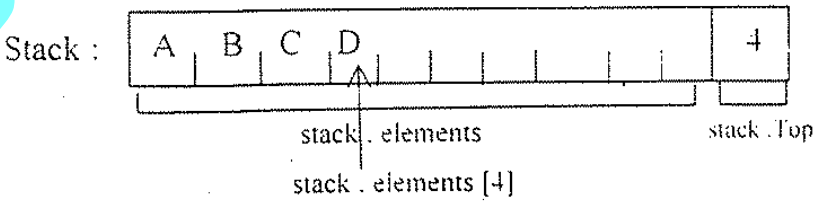
1- نحدث قيمة المؤشر (Top) الذي هو حقل في القيد stack ليصبح (4)

```
stack . Top = stack . Top + 1
```

2- نضيف العنصر الجديد (D) في الموقع الجديد (4)

```
stack . element [ stack . Top ] = D
```

وبهذا يصبح المكدس بالصورة التالية :



خالدة محمد طاهر قواسمة

تمرين : اعد كتابة البرامج الفرعية لعمليات المكدس باستخدام القيد .

تمرين : اكتب برنامجا فرعيا لاضافة ثلاثة عناصر من الاعداد الصحيحة الى المكدس (SET) الذي سعته (20).

الحل : ان المكدس المطلوب يمثل بالمصفوفة (SET) وسعتها (20) ونوع البيانات (integer)

```
Type  St : array [ 1 .. 20 ] of integer ;
Procedure  Push3(Var  SET : St ; Var Top : integer ) ;
Var     I : integer ;
Begin
For  I := 1   TO  3   Do
        Begin
        If  Top : = 20
        Then writeln ( ' the tack SET  is full ' )
        Else
                Begin
                Top : = Top + 1;
                Write ( ' Enter the element ' ) ;
                Readln ( Set [ Top ] )
                End
        End
End;
```

Read Item
set[Top] = item

تضمن هذا البرنامج الفرعي (procedure) خطوة التحقق من امتلاء المكدس داخل ايعاز التكرار لينفذ عند كل عملية اضافة مع ان سعة المصفوفة (20) والسبب اننا لا نعرف عدد عناصر المكدس قبل الاضافة .

غادة محمد طاهر قاسم

مثال : المكدس (TABLE) بسعة (30) عنصر يحتوي على اربعة عناصر A ، B ، C ، D ، اكتب برنامجاً فرعياً (procedure) لاضافة (8) عناصر اخرى .

الحل : ان المكدس المطلوب يمثل بالمصفوفة (TABLE) وسعتها (30) ونوع بياناته هو (char)

```
Type     st = array [ 1 .. 30 ] of char :
Procedure  Push8(Var TABLE : st : Var  Top : integer ) :
Var   I : integer :
Begin
        Top : = 4 :
        For   I : = 1   To   8    Do
        Begin
            Top : = Top + 1 :
            Write ( ' Enter the new element ' ) :
            Readln ( TABLE [ Top] )
        End
    End:
```

في هذا البرنامج الفرعي (procedure) لم نضع خطوة التحقق من امتلاء المكدس لكونها غير ضرورية لأن سعة المكدس هي (30) ويحتوي على اربعة عناصر فقط والاضافة المطلوبة هي (8) فقط لذا فإن المكدس لن يصل الى حالة الامتلاء .

عادة محمد طاهر قاسم

مثال : اكتب برنامج فرعي لحذف (4) اعداد حقيقة من المكدس (BOB) الذي سعته (15) عنصر

الحل : ان المكدس المطلوب يمثل المصفوفة (BOB) بسعة (15) عنصر ونوع البيانات (Real)

```
Type   St = array [ 1 .. 15 ] of real ;
 Procedure   POP4( BOB : St ; Var  Top : integer ) ;
Begin
     For   I : = 1   to   4   Do
     Begin
          If   Top = 0
          Then  writeln ( ' Error ... the tack is Empty ' )
          Else
                Begin
                     Item : = BOB [ Top ] :
                     Top : = Top – 1
                End
     End
End :
```

عادة محمد طاهر قاسم