PDF Eraser Free

جامعة ديالى
كلية التربية الاساسية
قسم الحاسبات

بسم الله الرحمن الرحيم

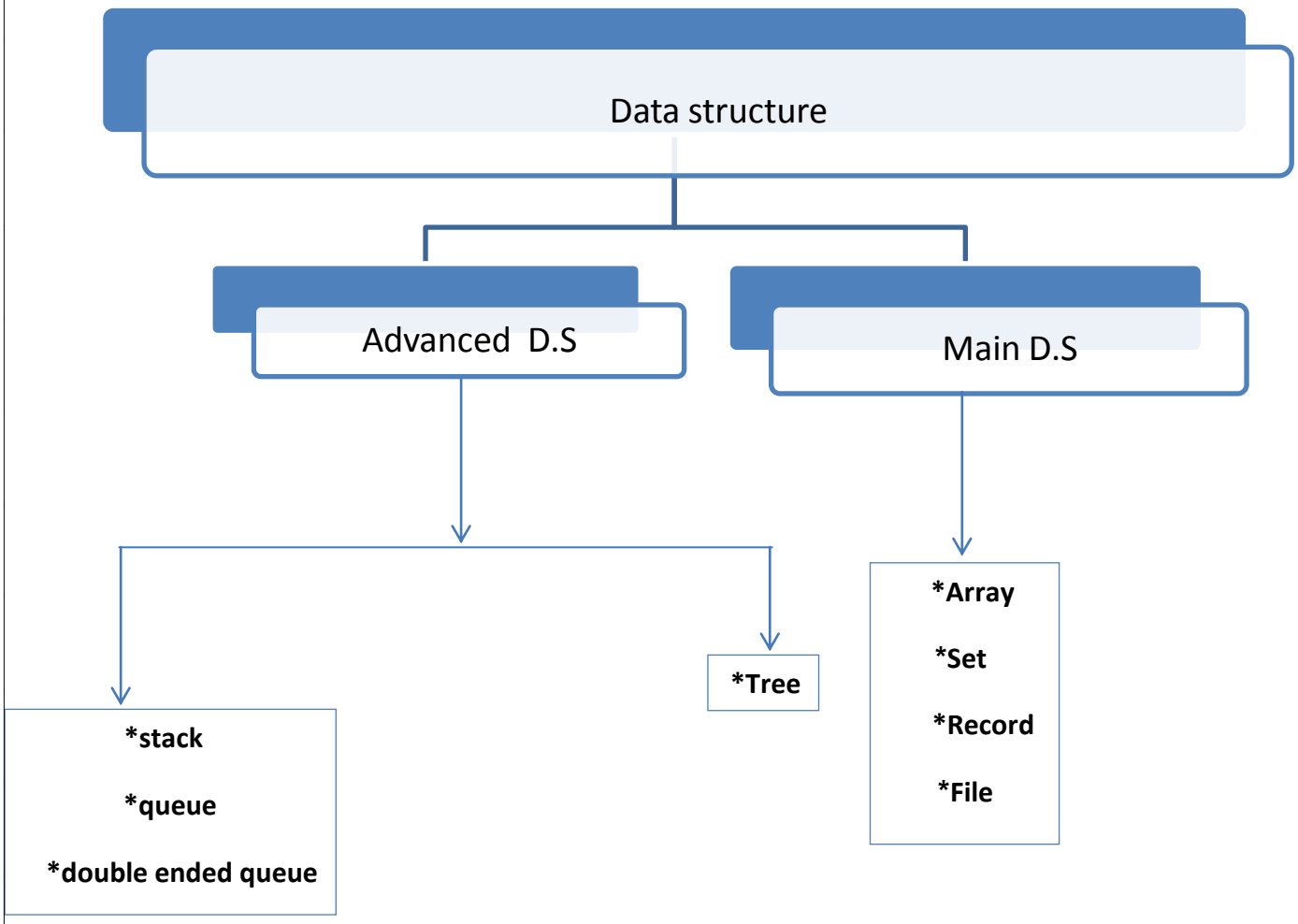
DATA STRUCTUER

(هياكل بيانات)

للاستاذ:

ياسر العنبكي

PDF Eraser Free



data:بيانات خام

Information:البيانات بعد معالجتها

Knowledge:مجموعه من الـInformation

Experience:مجموعه من الـKnowledge

Expert system:decision support system & data mining

Set:مجموعه من الـ files

File:مجموعه من الـ records

Record:وهي عباره عن سجل يحتوي مجموعه من البيانات

PDF Eraser Free

Data mining:

***ماهي هيكلة البيانات؟؟**هي طريقه تم إنشائها لكثرة البيانات (Data)المستخدمه في كل المجالات فكان لزاما علينا هيكلتها وتنظيمها لمعرفة كيفية الوصول للبيانات بالطرق الاسهل والادق، والواضح لدينا ان كل تخصص يتعلق بالحاسوب يبنى عليه _Data structure_

***ماهو الفرق بين البرنامج والخوارزميه؟؟**

Algorithm	Program
*Semantic Human language *منتهي	*Syntax *Machine language *غير منتهي

****(Syntax)**

وهي الصيغ او اكواد كتابة العمل او التعليمات في لغات البرمجه وتختلف من لغة برمجه الى اخرى

****(semantic)**

وهي معنى الصيغه البرمجيه بشكل عام في اللوغاريتمات او لغة البشر

***ماهي الخوارزميات؟؟**

مفهوم الخوارزميه قديم فقد وضع من قبل البابليون (١٨٠٠ سنه ق.م)في عهد حام ورابي وهو اول توصيف لقواعد بعض انواع المعادلات وجاءت كلمة خوارزميه نسبة للعالم المسلم(محمد بن موسى الخوارزمي)،

والخوارزميات هي مجموعه من الخطوات المنطقيه التي يتم تنفيذها حسب ترتيب محدد والتي تُصف بدقه ووضوح وتكون طريقه عامه لحل مسأله معينه.

PDF Eraser Free

ولحل مسائل عن طريق الحاسوب يجب إتباع الخطوات ذات الترتيب التالي:

١. تعريف المسأله، اي فهمها ثم تحديد المدخلات والمخرجات (I\O) لهذه المسأله.
٢. التحليل، اي تحديد العمليات التي تؤدي الى حل المسأله.
٣. التنفيذ، اي ادخال البرنامج وبياناته الى الحاسوب لحل المسأله وايجاد الناتج.

وكل مسأله نريد حلها عن طريق خوارزميه يجب ان تتوفر فيها المواصفات التاليه:

١. المدخلات(In put)واقل قيمه للمدخلات هي صفر(i\p>=0) اي اننا نستطيع عمل برنامج بدون ان نقوم بإدخال اي بيانات.
٢. المخرجات (out put)واقل قيمه للمخرجات هي ١ اي ان النتائج الخارجه يجب ان تكون معلومه واحده(o\p>=1) على الاقل
٣. المحدد، فكل تعليمه (instruction)يجب ان تكون واضحه ليس فيها غموض .
٤. يجب ان يكون لها حد اي (نهايه) فإذا قمنا بتتبع الإيعازات او الأوامر لأي لوغاريتميه يجب ان نصل الى عدد محدد من الخطوات في النهايه.
٥. الفعاليه (effectiveness)،اي ان كل ايعاز يجب ان يكون اساسي وبسيط وفي حال وصول العمليه اليه يجب ان يكون قابل للتحقيق.

*ما علاقة الخوارزميه بال data structure؟؟
هيكلة البيانات هي نتيجة لدمج الكائنات مع الخوارزميات،
والكائن(0bject) وهو بيان او رمز تحفظ البيانات بداخله وقد نعتبره x ونمرر من خلاله قيمه (بيان) وقد نحتفظ بالقيمه بداخله.

*ماهي خصائص الخوارزميات:
١) خطوات الخوارزميه مرتبه.
٢) خطواتها منتهيه.
٣) خطوات الخوارزميه تنفذ اعمال بسيطه.
٤) معرفه جدا.
٥) طريقة الحل يجب ان تكون عامه.

PDF Eraser Free

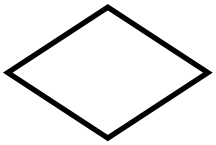
٦) فعاله وعند تنفيذ الخوارزميه في الحاسوب يجب ان لا تكون التكلفه عاليه سواءً من ناحية زمن المعالجه (processing time)او المساحه المشغوله في الذاكرة الرئيسيه.
٧) استخدام المخططات الانسيابية.
٨) استخدام جمل قصيره وخاليه من الكلمات الغير مفيده.

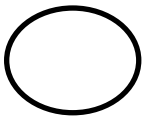
أشكال المخططات الانسيابية:


يستخدم للدلالة على بداية ونهاية الخوارزميه.


يستخدم للدلالة على عمليات الادخال والاخراج.


يستخدم للدلالة على عمليات المعالجات الحسابيه.


يستخدم لاختبار شرط معين وعمليات المقارنه وللعمليات التي تحتاج الى اتخاذ قرار.

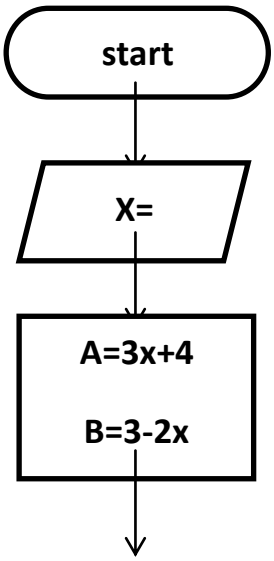

يستخدم لنقطة عودة البيان وتصل جز ء معين المخطط الانسيابي بأخر وغالبا يكتب داخلها رقم او حرف.


يدل على ترتيب خطوات الخوارزميه.

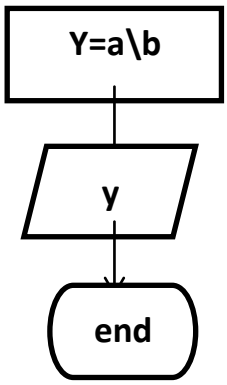
**مثال (١) ضع المخطط الانسيابي لخوارزميه لحل المسأله الرياضيه التاليه:

Y=3x+4\3-2x

الحل:



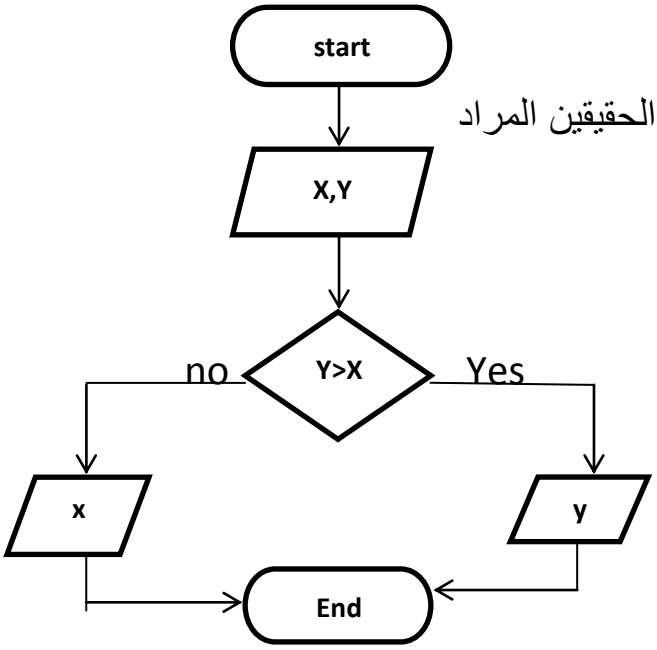
PDF Eraser Free



مثال(٢)صمم المخطط الانسيابي لخوارزميه للمقارنه بين عددين حقيقين موجبين وطباعة الاكبر بينهما:

الحل:

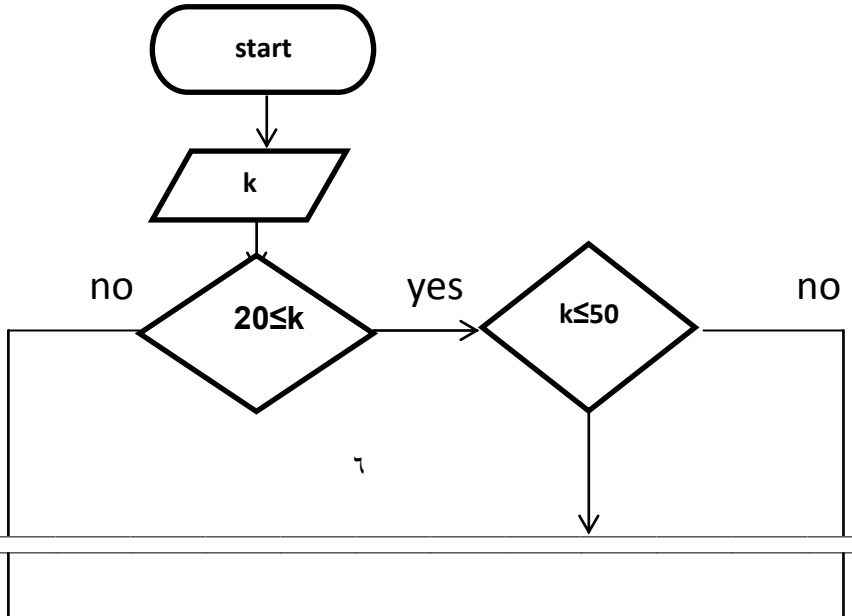
حيث ان X,Yتمثلان العددين الحقيقين المراد المقارنه بينهما.



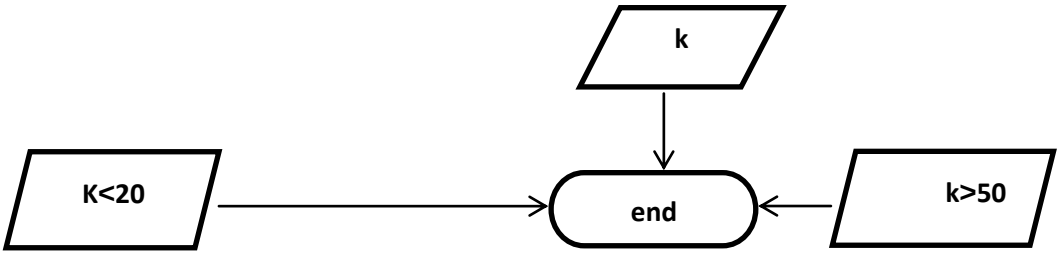
مثال(٣)ارسم المخطط الانسيابي للخوارزميه المناسبه للمتراجحه التاليه:

$$20 \leq k \leq 50$$

الحل:



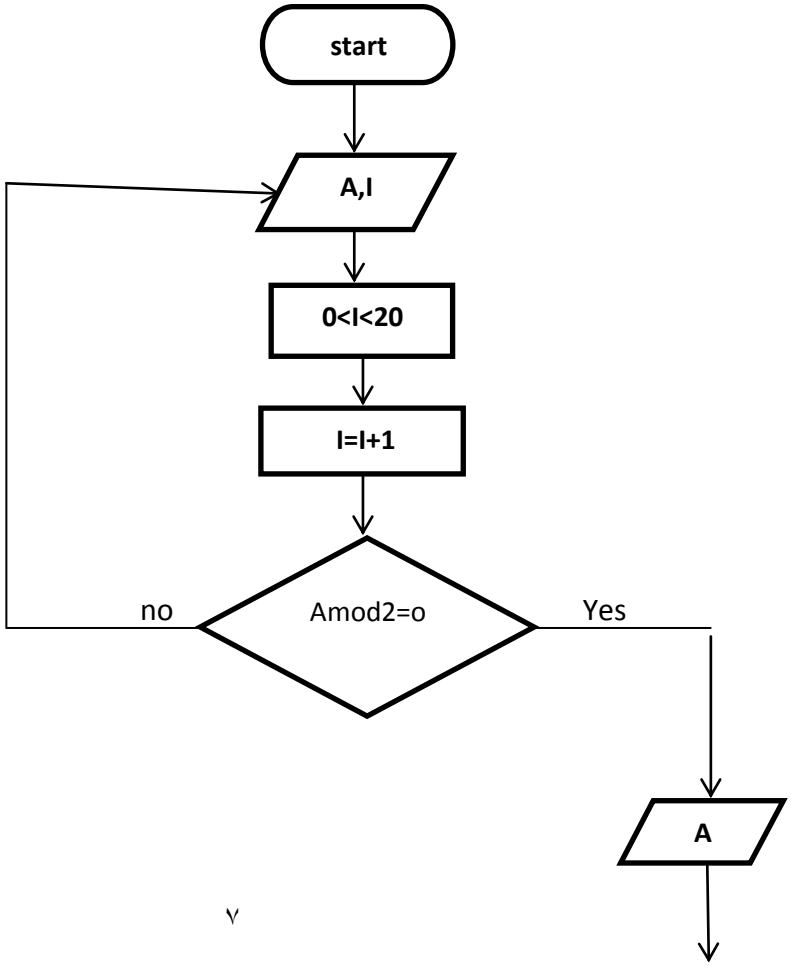
PDF Eraser Free



المكونات التكراريه:

يتفوق الحاسوب على الانسان بالقدره على تكرار العمل مهما كان حجم التكرار كبير ولأي فتره زمنيه دون تعب او كلل او ارتكاب اخطاء ويتم في حلقة دوران بطريقة (counter)،فبعض المسائل تتضمن تكرارلمعالجة بعض أجزائها عدداً معين من المرات ويطلق على هذه العمليه (loop) وهي تكرار معالجة مجموعه من الخطوات الى ان يتحقق الشرط ثم انهاء البرنامج

*مثال(٤)لدينا ٢٠ عدد صحيح ونريد فرز الزوجي منها فقط:



PDF Eraser Free

END

*الهيكل العام للخوارزميه:

يتكون الهيكل العام للخوارزميه من رأس وجسم فيتكون الرأس من اسم الخوارزميه ومجموعة (parameters) ويجب ان تنتهي بقوسين ()، اما جسم الخوارزميه فيتكون من مجموعة الخطوات او الاوامر التي تطلبها الخوارزميه.

*العناصر الأساسية للخوارزميه:

- ❖ اسم الخوارزميه يكون متبوع بقوسين.
- ❖ يتم التواصل عن طريق (parameters) بين الخوارزميات وتبادل البيانات.
- ❖ نقطة البدايه.
- ❖ جسم الخوارزميه وداخلها الخطوات والاوامر.
- ❖ النهايه (end)

مثلا:

```
Algorithm add1()          اسم الخوارزميه
Begin                     جسم الخوارزميه
Sum=0;
For I = 1 to 10 do;
Sum=sum+I
end                       النهايه
```

PDF Eraser Free

Main D.S.:

المصفوفات (Arrays):وهي عباره عن مجموعه من البيانات التي تتشابه في النوع ولها اسم مشترك وتخزن في الذاكره بشكل تسلسلي.

اولاً المصفوفه في بعد واحد بمعنى اما مصفوفة صف واحد اوعمود واحد تحتوي على (index) يبداء من ٠في لغة C ومن ١ في لغة باسكال.

ويتم تمثيله ب١)العموديه ، بداية ال (index)+ عدد الصفوف

٢)الصفيه ، بداية ال (index)+ عدد الاعمده

اما اكثر من بُعدين فيتم تمثيلها بالطريقه الاتيه:

١) العمود بداية ال(index +رقم الصف ×عدد الاعمده)
٢) الصف بداية ال(index +رقم العمود ×عدد الصفوف)

تمثيـــــــــــل المصفوفه في الحاسوب:

اي مصفوفه مهما كانت ابعادها سواءً بُعد واحد اوعدة ابعاد فإن الحاسوب يحولها الى بعد واحد دائما ويعتمد على مواقع العناصر ال(index) كما في الشكل التالي

(index):هي مواقع العناصر في المصفوفه

PDF Eraser Free

A{0}	A{1}	A{2}	A{3}	A{4}	....	....	....	A{I}

مواقع العناصر في مصفوفه ذات بعد واحد في الحاسوب

A{0,0}	A{0,1}	A{0,2}	A{0,3}	....	....	...	....	A{i,j}

تمثيل مواقع العناصر في مصفوفه ذات عدة ابعاد في الحاسوب

وفي المصفوفه ذات البعد الواحد لمعرفة موقع اي عنصر نعتمد على موقع العنصر الاول في المصفوفه وذلك لان موقع اول عنصر يتولد عشوائيا (لأنه في الramاعتماداً على ما قبله من قيم والذي يليه يكون تراكمي، وتوزع العناوين اوتوماتيكياً وحسب نوع البيانات.

فمثلا بفرض ان نوع البيانات integer والموقع الاول هو 005 فإن الموقع الذي يليه هو 007 وذلك لان هذا النوع من البيانات يحجز تلقائيا ٢بايت في الذاكره لكل عنصر.

A{1}	A{2}	A{3}	.....	....	....
005	007	009			

وذلك بفرض ان البيانات من نوعinteger ⟵

*ولإيجاد موقع اي عنصر نأخذ الخوارزميه التاليه وهي بلغة باسكال:

Loc (a{i}) = add (a{1} + size of type * i-1)

حيث ان a{i} هو موقع العنصر المراد ايجاده

A{1} عنوان اول عنصر في المصفوفه

size of type حجم نوع البيانات

i-1 وهو موقع العنصر السابق للعنصر المطلوب

*مثال : اوجد قيمة الموقع a{3} في الشكل السابق:

Loc (a{3}) = add (a{1} + 2*2)

=(005 + 2*2)

=009

ولإيجاد موقع العنصر نفسه بلغة الC++

Loc(a{i}) = add (a{0} + size of type *I)

PDF Eraser Free

Loc (a{2}) = add (a{0} + 2*2)

A{0}	A{1}	A{2}	....	....	....
005	007	009			

=(005+ 2*2)

=009

*اما في المصفوفه ذات عدة بعدين نجد مواقع المصفوفه التاليه بلغة الباسكال كالتالي:

A{1,1}	A{1,2}	A{1,3}	....	....	....
005	007	009	011		

Loc (a{i},{j}) = add (a{1} {1} + size of type *[M*(i-1)+(j-1)]

حيث ان m عدد الاعمده الكلي

** وفي لغة ال C++:

Loc (a{i},{j}) = add (a{0},{0}) + size of type*(M*i+j)

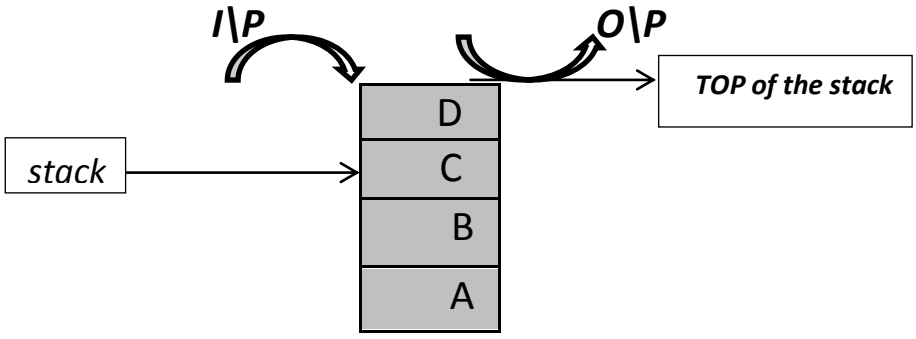
Advanced D.S:

Stack:

هو عباره عن مصفوفه تُعبأ البيانات فيها بطريقه تعكس البيانات ، وهو نمط من الـ D.S.

وعملية عكس البيانات تحدث بسبب وضعية الـStack وترتب البيانات على حسب FILO(Frist in last out)

ودخول البيانات وخروجها يتم من نهايه طرفيه واحده تسمى (top of the stack) المؤشر



PDF Eraser Free

ماهي العمليات(primitive operation) التي يجريها الـ stack:

١)التهيئه (initialization) وفيه يتم تحديد حجم الـstack وموقع المؤشر ونوعه

٢)الادخال (insertion) or (push)

٣)الاخراج(deletion) or (pop)

ولدينا top وهو المؤشر الخاص بـstack .

ففي البدايه يكون المؤشر (top) في الاسفل وكلما تم ادخال بيان على الـ stack يصعد المؤشر الى الاعلى فيعتمد حجم الـstack على موقع المؤشر، فعندما يكون الـstack فارغاً تكون قيمة المؤشر (-١ في c) و (٠ في باسكال)

ومن صفات المؤشر (top):

١)هو مؤشر يصعد ويهبط على حسب العمليات (ادخال، اخراج)

٢)وهو موقع في الذاكره يحمل قيمه واحده

٣)يتناقص عند الحذف ويزداد عند الاضافه او الادخال

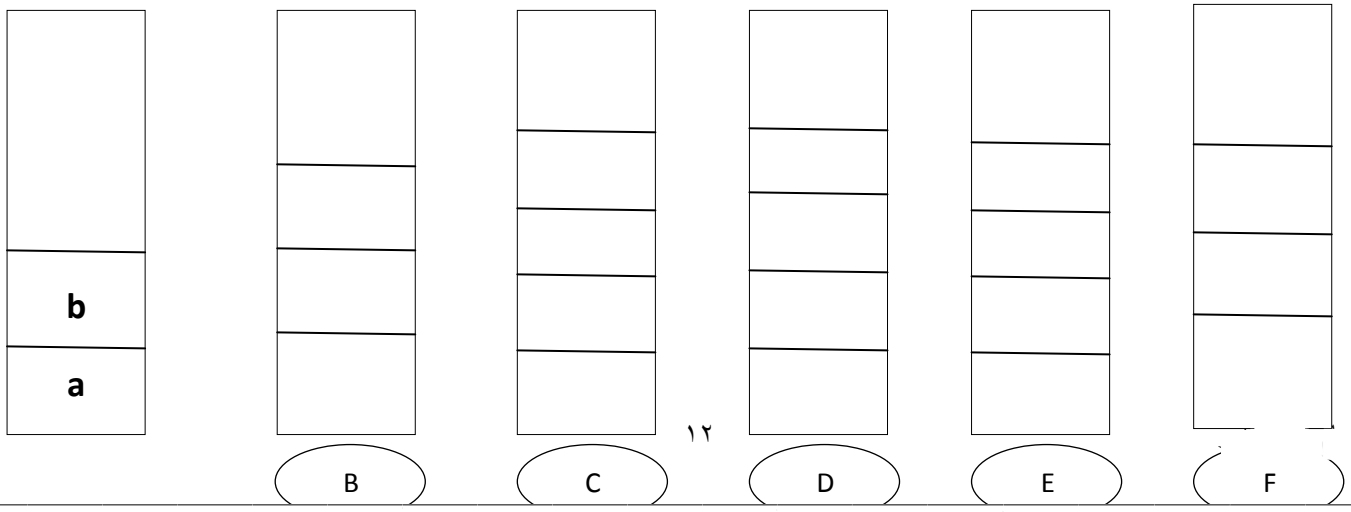
٤)يدل على حجم الـstack

٥)عند الادخال (push) يجب التأكد ان المؤشر لم يصل الى القيمه العظمى

٦)عند الاخراج (pop) يجب التأكد ان المؤشر ليس فارغا

وعند تهيئة الـstack علينا ان نعطي المؤشر قيمه تدل على انه فارغ ونحدد الـstack بـ s وitem ب i

فيقدموا لنا عملية الادخال push(I , s) و الاخراج pop(s)



PDF Eraser Free

A

Push (s , c) ; (frame (b))

Push (s , d) ; (farm (c))

Push (s , e) ; (farm (d))

Push (s ,f) ; (farm (e))

Pop (s) ; (farm (e))

Pop (s) ; (farm (f))

وسميت بالـ push down list بسبب العمليات المتكرره

اذن ما نلاحظه يشرح لنا عمليات pop & push

و ينتج لنا الاطار ويتم داخله جميع العمليات وهو ايضاً مايُعرف بالـarray ومعه المؤشر.

*الاطار (frame) في الخوارزميات ولغات البرمجه هو الكود الخاص المختص بإنشاء (stack) والمؤشر والـ items

**تمثيل الـ stack بلغة ال C :

مثال: انشئ بلغة C (stack) حجمه ١٠٠:

```
Include<isotream.h>
# dfine st_size 100
Stract stack
{
Int top;
Int item [s+ size];
};
```

frame

PDF Eraser Free

الـframe يحتوي على عنصرين رئيسين:

- ❖ مصفوفه لتحمل جميع العناصر لـstack
- ❖ Integer ليشير الى موقع مؤشر الـstack

*صمم إستماره تحتوي على ٢٠ عنصر (اسماء طلاب)

```
# dfine name_size 20
Strict     stack
{
Int top;
Char  item [name_size]
}s;
```

S عباره عن كائن نضع فيه البيانات وتوجد عمليات تطبيقيه ستؤخذ لاحقاً تعتمد على الادخال والاخراج.

لو افترضنا ان عناصر الـ stack تحتوي العناصر item[0] to item[99] وايضاً لا يوجد لدينا سبب بان نجعلها محدده فقط على نوع واحد من البيانات ولكن نفرض اننا نحتاج عدة انواع من البيانات مثلاً Float , char , int او أيا كانت نوع البيانات التي سنعطيها للـstack

اذن يمكن للـstack ان تحتمل مختلف انواع البيانات

```
# dfine STACK SIZE 100
# dfine INT 1
# dfine FLOAT 2
# dfine STRING 3
Stract stack element
{ int e type
```

PDF Eraser Free

```
Union
{ int I val;
  Float f val;
  String c val;
}element;
}
Strict stack
}
Int top ;
Strict stack element item[STACKSIZE] };
```

ولطباعة العنصر الاعلى للـstack :

```
Strict stack element  sp;
Sp = s.item {s.top}
Switch (sp-etype)
{
case INT :print f (sp.i val);
Case floaT :print f (sp.f val);
Case string :print f (sp.c val);
}
```

PDF Eraser Free

وعندما تكون الstackفارغه لا تحتوي على عناصر نجعل الtop=-١ أي (s. top=-١)كود برمجي

```
If(s.top==-1);
Return(nuII/Empty);
Else
Return (item);
```

ااو كخوارزميه عامه

```
If(s.top == -1)
stack is empty;
Else
Stack is not empty;
```

تطبيقات الـ pop:

إحتماليات (under flow) "وهي الوصول الى النهايه" توجب علينا دراسة عملية الـ pop للعناصر داخل الـstack الذي قد يكون فارغا وتجنب ذلك.

عند عمل مثل هذا الطلب عن طريق المستخدم يجب إظهار رساله للمستخدم تعلمه انه لا يوجد اي قيم وتخبره عن حالة (under flow) هذا يحتم على الـpop ان يؤدي الخيارات الثلاثه التاليه:

١)اذا كان الـstack فارغا إطبع رسالة تحذير وأوقف التنفيذ (halt execution)

٢)أبعد المؤشر (top) من الـstack

٣)ارجع هذا العنصر element للبرنامج الذي يقوم بالاستدعاء.

PDF Eraser Free

*مثــال:

```
Int pop (struct  stack .ps)
{
If (empty (sp) );
{ print ("stack under flow") ;
Exit();
 }
Return (ps_item s [ps_top--]);
}
```

ولو افترضنا ان عملية pop تستدعى

ps;

ps_top[88];

ps_top[87]; فيأشر الى

تطبيقات لعمليات push:

وهذه تكون أسهل في التطبيق باستخدام مصفوفة pest array وأول محاولة للـpush الإجراءات تكون كالتالي:

```
Void push(struct  stack PS, int x )
{
PS ⟶items[ ++(PS⟶ Top)]=x;
Return;
}
```

PDF Eraser Free

وهنا الإجراء يجعل item x push إلى stack بزيادة S.top بقيمة واحدة.

ومن ثم يدخل قيمة x إلى عناصر array s لكن تدخل معنا مشكلة overflow وأن تطفح الحاوية stack بـitems عن المقدر له هنا يجب إظهار رسالة بذلك.

```
Void push (struct* PS; int x)

{

If(PS      top== stack size -1)

{print f("stack is full");

Exit(1);

}else

PS      items[(PS      top) ];

Return;

};
```

ملاحظـــة:

١/أي شيء يستخدم مصفوفة يواجه مشاكل تحتاج لحل.

٢/لا تستطيع تغيير حجم المصفوفة أثناء التنفيذ بالبرنامج.

Infix, postfix, prefix:

إذا كانت العمليه ضمن المعادله فتعتبر ***Infix***

واذا كانت العمليه تلحق بالمعادله فتعتبر ***postfix***

اما اذا كانت العمليه سابقه للمعادله فتعتبر ***prefix***

هذا القسم يعتبر التطبيق الرئيسي الذي يوضح الانواع المختلفه بالstack والعمليات المتنوعه وايضا الدوال المطبقه عليهم.

PDF Eraser Free

*مثــال للتوضـــــيح:

Infix = A+B

Postfix= AB+

Prefix= +AB

حيث ان operands=A,B

$,%,*,-,+ وغيرها تكون operator (عمليات)

اي ان هذه التسميات تطلق على حسب موقع العمليات .

**مثــال:

حول العمليه التاليه الى ***Postfix*** و***Prefix*** :

١) ***A+B*C***

نقوم بتحريك العمليات فقط وليس العمليات،

Prefix= +A*BC

Postfix=A+BC*

=ABC*+

** مثال:

اذا كان لدينا المعادله الاولى ولكن مع وجود اولويه للجمع حولها لل Prefix , Postfix :

(A+B)*C

Prefix 1)=+AB*C

2)=*+ABC

Postfix= AB+C*

PDF Eraser Free

ولو اعتبرنا ان لدينا 5 operators التاليه اضافه الى العمليات المنطقيه:

a) Add
b) Multiplication
c) Subtraction
d) Division
e) Exponentiation
f) Not
g) % باقي القسمه

فيكون ترتيب أولوية العمليات كما يلي :

() العمليات داخل الاقواس

$ ، not

%

/ ، *

+ ، -

≥،≤،>،<

|| ، &&

** مثــال: حول المعادله التاليه الى postfix & prefix:

=(A+B) * (C-D)

Prefix =1) (+AB) * (-CD)

2)*+AB -CD

Postfix =1) (AB+) * (CD-)

PDF Eraser Free

2)AB+ CD-*

** مثال:

A$B*C-D+E/F/(G+H)

prefix= 1) $AB*C –D+E/F/ (G+H)

2)*$ABC- D+E/F/ (G+H)

3) -*$ABCD +E/F/ (G+H)

4)+-*$ABCD//EF +GH

تمرين: اوجد postfix لنفس المعادله؟؟

** مثال:

=((A+B)*C-(D-E)) $ (F+G)

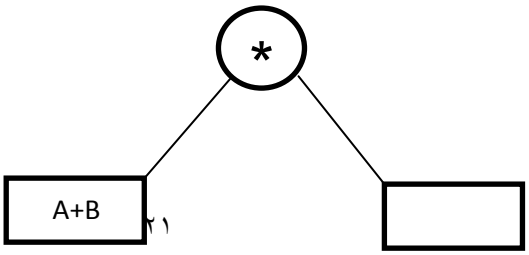
Prefix=1) ((+AB)* C- (-DE) $ (F+G)

2) -*+ABC-DE$(+FG)

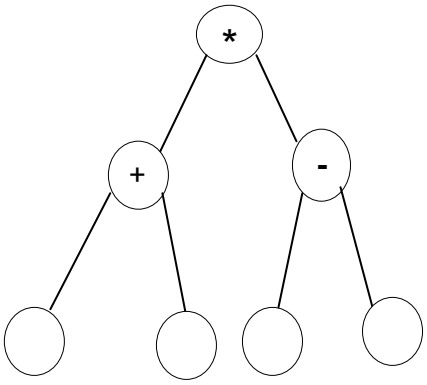
تمرين: اوجد postfix لنفس المعادله؟

طريقة TREE:

1: (A+B) * (C-D)



PDF Eraser Free

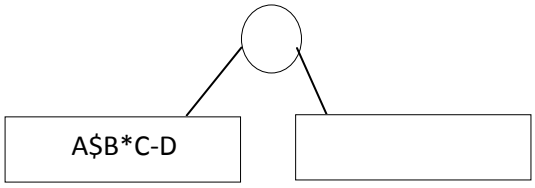


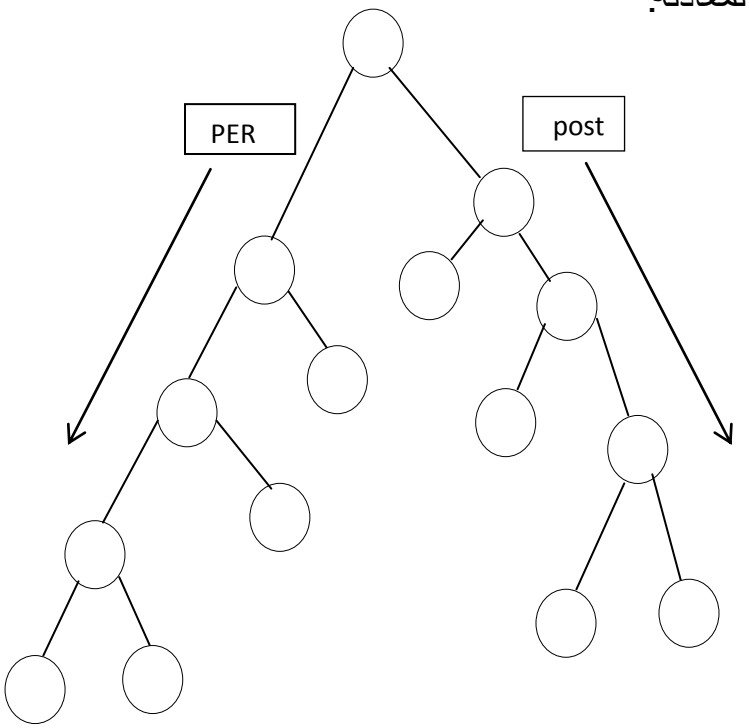
Per = *+AB-CD

Post = AB+CD-*

ونختار العمليه الاولى بحيث يحصل توازن في المعادله.

2:A$B*C-D+E/F/(G+H)

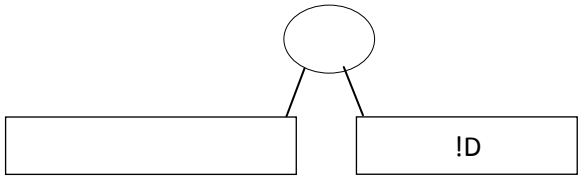


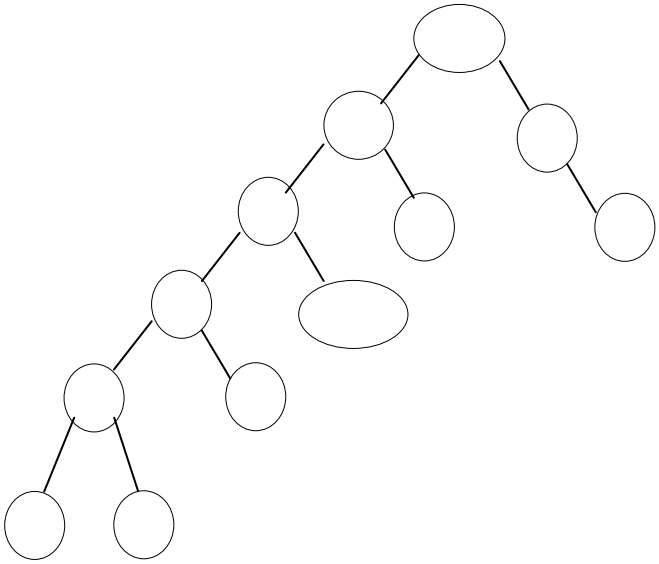


PER = +-*$ABCD//EF+GH

3:(X+Y*Z> 100)&&Cll ! D

PDF Eraser Free





pre = ll&&>+X*YZ100C!D+

:Queue

PDF Eraser Free

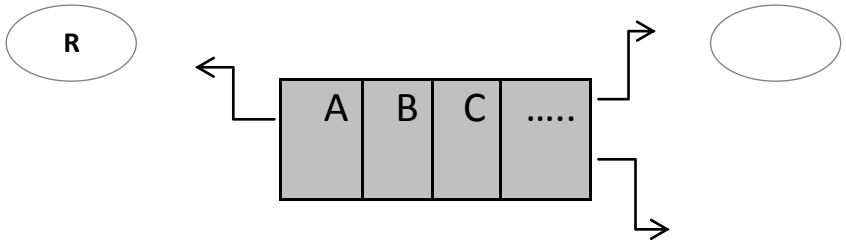
ماهو الفرق بين الـstack و الـ queue؟؟

*الـstack: كما ذكرنا إنها مصفوفه يتم التعامل معها بطريقه مميزه بحيث يكون الـ I/O من طرف واحد بواسطة الـ Top

* الـ queue:هي ايضاً مصفوفه لكنها مفتوحة الطرفين ، اذن فهي تحتاج الى مؤشرين يقومان بعمليات الحذف والاضافه .

1) insert ⟶ Rear (مؤشر عملية الادخال)

2) delete ⟶ Front (مؤشر عمليات الاخراج)



(queue)

كيف تتم التهيئـــــه؟

في الـstack كانت تتم التهيئه بجعل (Top = Null) ،

أما في الـ queue يكون من خلال Rear الذي يتحكم بالتهيئه عندما تكون (F= 0 , R= -1) وبذلك يتولد لدينا قانون خاص بالـ queue وهو (R < F) عند التهيئه

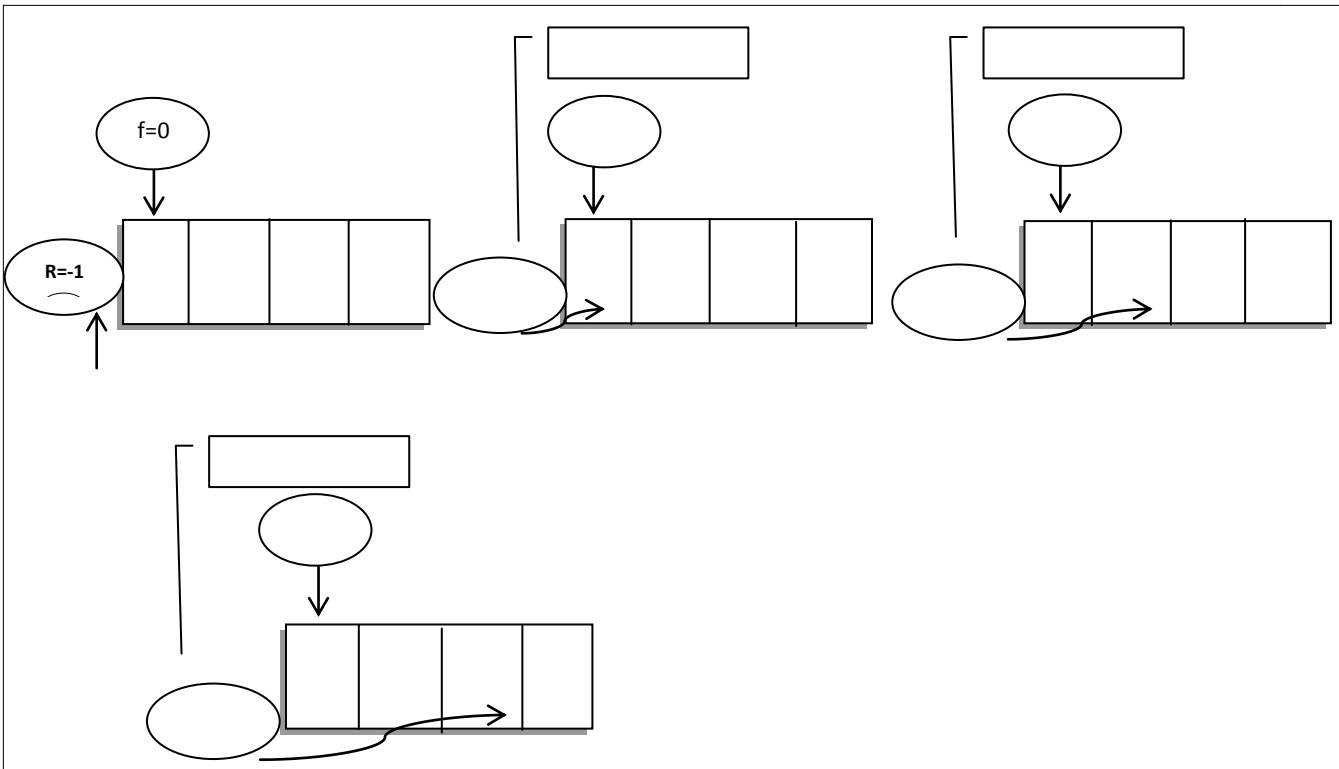
ولإيجاد عدد عناصر الـ queue نستخدم القانون:

$$(Q = R - F + 1) *$$

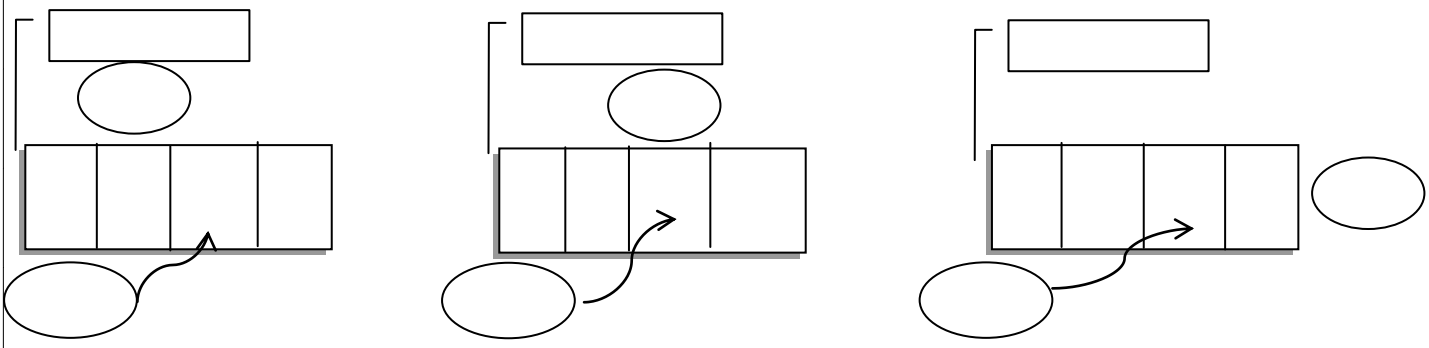
** تمثيل عملية الإضافه Insert :

-1-0+1=0

PDF Eraser Free



ولتمثيل عملية الحذف: مثلا(Delete A)



الفرق بين عمليتي الحذف والاضافه في ال stack وال queue

في stack اذا اردنا الحذف نجعل الTop عند القيمه المراد حذفها ثم نحذف فتنتقل قيمة الTop وعند الاضافه تزيد قيمة ال Top

Queue: عند الحذف تتزايد قيمة frontوتبقى قيمة rear ثابته وعند الاضافه تتزايد قيمة rear وتبقى front ثابته.

وعندما R < F يكون الQueue مهيئه.

*برنامج من اجل إنشاء Queueيحمل مؤشرين:

PDF Eraser Free

```
Struct Queue
{
 Char element [max];
        Int r, f ;
 }
```

** برنامج لتهيئة ال Queue:

```
Intial  (struct Queue * pq)
 {
    Pq ⟶  rear= -1;
    Pq⟶ front= 0;
  }
```

*** خوارزمية برنامج إضـــــــــافه:

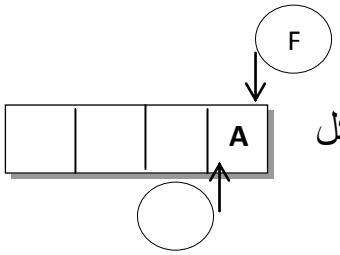
```
Insert (struct queue * pq , char)
   {
     If (pq ⟶ rear < max – 1)
          { (pq ⟶  rear)++
Pq ⟶ element [pq ⟶rear]=e;
          }
        Else print f ("full");
    }
```

PDF Eraser Free

**** خوارزمية برنامج حذف :

```
Char delete  (struct queue * pq)
{
    If (pq → rear < pq → front);
    {
    Print f ("empty")
    Return ;
    }
Else
Return (pq →element [pq →element [pq →front++];
}
```

اما اذا كان لدينا هذا الشكل

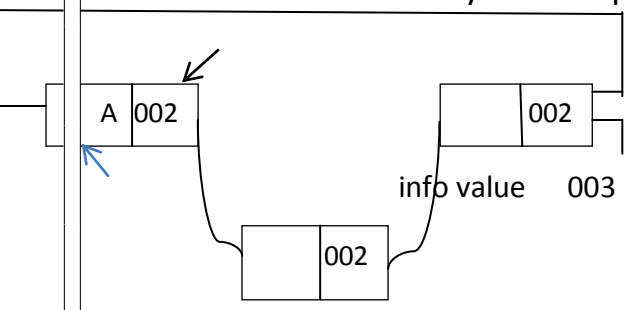


فلا نستطيع الاضافه لأن المؤشر يشير الى الامتلاء بحسب الترتيب ولكنه في الحقيقه فا رغ. ولذلك نجد أن الحل فيه عملية shift لنقل العناصر وهذا يأخذ منا وقت كبير وأيضا جهد في عمل code وأيضا يؤثر على مشكلة الأسبقية ولذلك ظهرت لدينا Queue circular .

PDF Eraser Free

Linked list :

كما ذكرنا أن stack و Queue عبارة عن مصفوفات لها خصائصها واحتياجاتها من الناحية الخوارزمية ومن الناحية البرمجية وقواعدها في الإدخال والإخراج وعملية الحفظ للعناصر والتي تتم في عناوين ثابتة في الذاكرة Index .وهنا يأتي تمثيل المصفوفات في الذاكرة بطريقة ديناميكية تسمى Dynamic representation



memory address

تمكنها من الحفظ والإضافة وتمثل على شكل عُقد

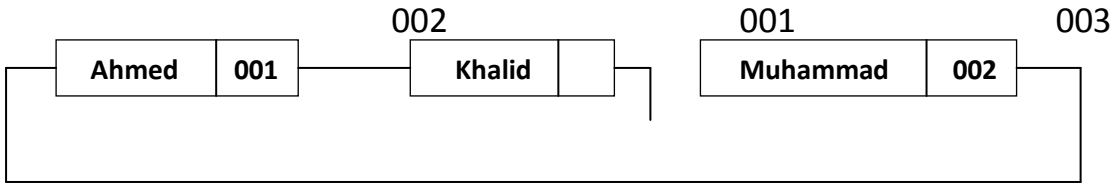
(node) وكنا إذا أردنا تحديد حجم الـstack وQueue

فلا بد أن نحدده قبلا ويصبح هنا الحجم ثابت.

ولكن هنا في Linked list لا نحدد الحجم مسبقا

فكنا إذا أردنا إضافة عنصر ينشأ عقدة جديدة وهكذا.

وللربط بين العُقد يكون بحسب العناوين. مثال:-



ولإنشاء سجل عن طريق node بلغة الباسكال:

```
Ptr node=^Node

Node=Record

Name=string [20]

ID=Int

Link: Ptr, Node;

End
```

^ :عندما يجد الـ compilerهذه العلامة فإنه يقوم بتعريفها لاحقا على حسب لغة البرمجة في C++language سيتم تعريفها في جميع اللغات على حسب اللغة إلا في لغة الباسكال فهو يعرف هذه العلامة محجوزة.

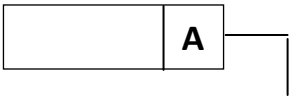
PDF Eraser Free

Link:(الرابط) يؤشر على node من نفس النوع .

وفي المثال السابق نعرف الرابط link بنفس Ptr Node (بنفس المؤشر)فهو يحمل عنوان Nodeآخر.

ومن أجل إنشاء أول عُقدة نستخدم مُعرف جديد(New) ومن أجل تعبئة البيانات سنكتب عبارة التعبئة (first ^.Name='A';).

```
Begin
    First (New)
    First ^.Link =null;
    First ^.Name='A';
                End;
```



عملية الحذف نضيف عبارة despose first;.

من مميزات Linked list:

١)عند الحذف تَحذف العنصر فعليا من الذاكرة ولا نستطيع استرجعها .

٢)لكل سلسلة لا بد أن يكون لها مؤشر لأول السلسلة وإذا فُقد الرابط (link) نفقد البيانات وحينها تظهر لنا رسالة وهي الذاكرة ممتلئة .

٣)إذا خرجنا من البرنامج تبقى العناصر في الذاكرة حاجزةً مساحة إلى أن يتم حذفها على عكس المصفوفات عند إغلاق البرنامج فإن البيانات تُحذف.

ولو قمنا بلغة C++بتعريف رابط وعقدة جديدة يجب أن يحتوي هذا التعريف على ثلاث نقاط أساسية:

أ - نقطة البداية .
ب - المعلومة التي بداخل هذه النقطة.
ت - الرابط (Link)

```
Struct prt Node  ⟵  جملة التصريح (١)
{
 Int info;
Struct ptr Node *link;
*First, *last;
```

PDF Eraser Free

}

هذا البرنامج يحدد أول عُقدة وآخر عُقدة للبرنامج .

Type def struct ptr Node Node ⟵ جملة تصريح (٢).

وعند هذا التصريح يجب أن أنسب جميع المتغيرات إليه مثل:

Type def int x.

إذا أردنا الحجز في الذاكرة هذا يعني أننا نريد أن نحجز في مؤشر لو افترضنا .....

Int *p; لو عرفنا P=malloC(2);

Int *P;

P=malloC(2);

مثـــال: للحصول على Node يجب أن ننشأ have بدالة الإنشاء وهي get Node.

إنشاء Node جديد ضمنNode السابقة:

```
Node *get Node ( )
{
   Node *P;
   P= (Node*) malloC (size of (int));
return;
}
```

أنشأ عقدة جديدة f تكون بالشكل:

```
Node *f
F=get Node ( )
```

PDF Eraser Free

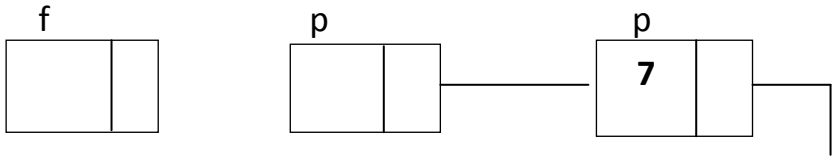
أدخل قيمة لهذه العُقدة ثم احذفها؟

Linked lest لها شروط:

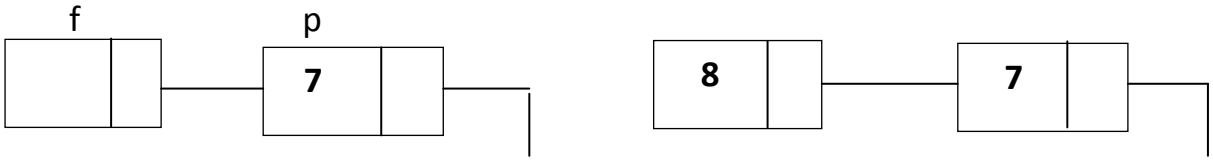
- إذا كانت وحيدة لها برنامج وهو:

```
Intial (Node *f)
{
   F=get Node;
    f ->Link=null;
    f-> Info=7;
 }
```

- في حالة أننا نريد إضافة عقدة جديدة للسابقة بحيث أن تكونf هي الأولى حيث أن تكون العقدf هي الأولى وتأخذ دائما قيم الجديدة والقيم السابقة إدخالها تنزاح إلى العقد الجديدة.



(a) (b)



(c) (d)

```
Insertion (Node *P)
{
  P=get Node;
  P->link=f;
```

PDF Eraser Free

```
  P=f;

  f->info=8;


}
```