المكدس (Stacks)

عند تنفيذ البرنامج يتم نقل البرنامج التنفيذي من وحدات التخزين إلى الذاكرة الرئيسية وتنقسم الذاكرة إلى عدة مقاطع ومنها مقطع البيانات Data segment والذي يتم تخزين البيانات فيه ومقطع المكدس Stack segment الذي يتم تخزين المتغيرات المحلية وعناوين الدوال والمقاطعات وغير ذلك.

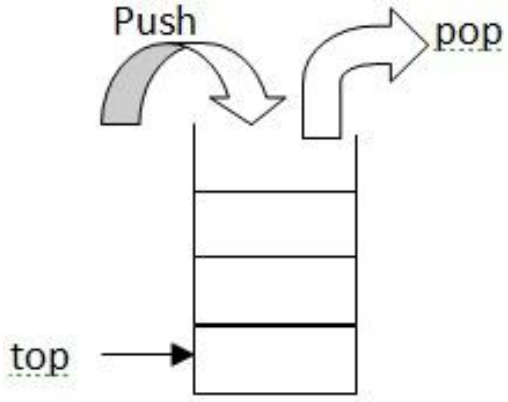
تعريف المكدس Stack

هي طريقة خاصة لخزن البيانات والتعامل معها بترتيب معاكس لترتيب استعادتها (سحبها)، أي أن آخر قيمة يتم ادخالها (دفعها) الى المكدس هي أول قيمة يتم سحبها من المكدس، وتدعى هذه الآلية LIFO (Last Input First Output) الداخل أخيراً الخارج أولاً وبهذا فإن الاضافة والحذف تكون من أعلى المكدس.

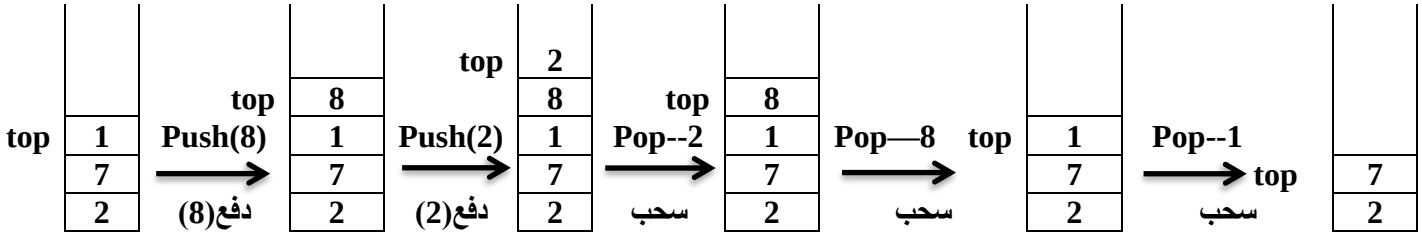
يشبه المكدس صندوق تخزن فيه البضائع فوق بعضها، ويمكن استخدامه في التطبيقات البرمجية لإجراء عمليات التراجع عن الخطوات المنفذة (Undo).

العمليات التي تتم على المكدس

يعتبر المكدس مساحة حقيقية في ذاكرة الحاسوب، لكن هذه المساحة تختلف من غيرها من حيث التعامل مع البيانات المخزنة فيها، والعمليات التي تتم على هذه البيانات محدودة ومقيدة بشروط، حيث لا يمكن القيام فقط بعمليتي الإضافة والحذف، لكن هاتين العمليتين لا تتم بشكل حر، بحيث يمكن الاضافة في أي مكان في أي مكان في المكدس، وكذا الحذف لا يمكن الحذف من أي مكان في المكدس، بل هاتين العمليتين مقيدتين بقيود هي، أنه يمكن عملية الإضافة وكذا الحذف من اتجاه واحد فقط من قمة المكدس كما هو موضح في الشكل (1)



الشكل (1) يوضح آلية عمل المكدس



الشكل (2) يوضح عملية الاضافة والحذف من المكدس

تطبيقات المكدس

المكدس له تطبيقات كثيرة أدناه بعض تطبيقات المكدس :

1. معالجة البرامج التي تحتوي على برامج فرعية

عند تنفيذ أي برنامج فإن المعالج سيكون له الدور الاساسي بعملية التنفيذ ، لذلك اذا حدث وان تم استدعاء دالة معينة في البرنامج فإن المعالج سيتوقف عن اكمال تنفيذ البرنامج حسب تسلسل الاوامر ويتفرع لإكمال تنفيذ الدالة ثم العودة الى ذات الموقع الذي تفرع منه لإكمال التنفيذ حسب تسلسل الاوامر ، في هذه الحالة فإن المعالج يجب أن تكون له الامكانية للعودة الى البرنامج في الموقع الذي تفرع منه لتنفيذ الدالة التي تم استدعائها ، لذلك فإن العنوان الحالي للبرنامج (أي الاوامر الذي كان من المفروض أن ينفذ قبل أن يتفرع المعالج الى تنفيذ الدالة سيدفع الى المكدس ليتم حفظه ، وعندما ينتهي المعالج من تنفيذ الدالة ، فإن العنوان الذي تم خزنة سيتم سحبه من المكدس ليوجهه المعالج اليه لإكمال التنفيذ .

2- المكدس يدعم طريقة الاستدعاء الذاتي.

المكدس يستخدم ايضاً بواسطة المترجمات بعملية معالجة أو تقييم التعابير وتوليد شفرة لغة الماكنة، كذلك يستخدم لخزن عناوين الاعادة بسلسلة من طرق الاستدعاء خلال تنفيذ البرنامج.

3- يستخدم المكدس بشكل كبير في أنظمة الحواسيب الحديثة، يستخدم على مستوى معمارية الحاسوب

حيث يستخدم في التصميم الاساسي لنظام التشغيل لأغراض تنفيذ المقاطعة واستدعاء دوال نظام التشغيل، ومن بين الاستخدامات الاخرى فإن المكدس يستخدم لتشغيل أو تنفيذ ماكنة جافا الافتراضية Java Virtual Machine ولغة جافا نفسها لديها صنف يسمى المكدس، والذي من الممكن استخدامه من المبرمج. المكدس في كل مكان، أنظمة الحاسوب التي تعتمد على المكدس هي واحدة من التي تخزن المعلومات بشكل مؤقت في المكدس بدلا من مسجلات وحدة المعالجة المركزية.

4- ايجاد نتيجة التعابير الحسابية:

المكدس مفيد لإيجاد نتيجة تعبير حسابي، افرض لديك التعبير التالي:

5*3+2+6*4

ممكن ايجاد نتيجة التعبير وذلك بإيجاد، اولا نتيجة حاصل ضرب 5 في 3 ، بعد ذلك تخزن النتيجة في متغير مثلا A ، ثم قم بإضافة الرقم 2 الى المتغير A واخزن النتيجة بالمتغير A . الان اجري عملية الضرب للرقمين 6 في 4 واخزن الناتج في متغير B عند ذلك ستنتهي العملية بإضافة A الى B واترك النتيجة النهائية بالمتغير A

A=15 (5 3 *)
A=15+2 (15 2 +)
=17
B=6*4 (6 4 *)
=24
A=17+24 (17 24 +)
=41
5 3 *2 + 6 4* +

ويعرف هذا التدوين (تعبير العلامة اللاحقة Postfix) ويتم تقيمها على النحو المبين أعلاه .

التعابير الجبرية Algebraic Expression

- التعبير الجبري :هو ترتيب رياضي يحتوي على معاملات وعوامل رياضية.
- المعاملات :هي كميات(وحدة بيانات) والتي تجرى عليها العمليات الرياضية.
- والمعاملات ممكن ان تكون متغيرات مثل x, y, z او ثوابت مثل.. 22 - 2 الخ,2, 34,
- العوامل الرياضية :هي رموز تدل على عمليات رياضية او منطقية بين المعاملات ,مثال على بعض العوامل الشائعة=> ,^ ,/ ,* ,- ,+

اذا ما اخذنا هذين التعريفين للعوامل والمعاملات , فإننا يمكننا الان ان نكتب تعبير مثل :

x + y * z

واقعا هناك ثلاث أنواع من التعابير الرياضية .وهي:

- تعبير العلامة القياسية : **infix**

وهذا هو النوع الذي اعتدنا على التعامل معه في الرياضيات منذ الايام الاولى للمدرسة، حيث يكون موضع العامل الرياضي(العلامة الرياضية)بين المعاملين الذين من المفروض ان تجرى عليهم العملية الرياضية وفقا لهذا العامل , مثل

A + B

- تعبير العلامة اللاحقة : **Postfix**

في هذا التعبير يكون العامل الرياضي تابع او لاحق الى المعاملين اي بعد المعاملين الذين من المفروض ان تجرى عليهم العملية الرياضية وفقا لهذا العامل، مثل

A B +

- تعبير العلامة السابقة : **Prefix**

هنا يكون مكان العامل الرياضي قبل المعاملين الذين من المفروض ان تجرى عليهم العملية الرياضية وفقا لهذا العامل، مثل

+ A B

<u>اسباب استخدام تعبير العلامة اللاحقة وتعبير العلامة السابقة</u>

بالتأكيد فان القارئ ربما يتساءل لماذا نستخدم هذا الشكل الغريب من تعابير العلامة السابقة والعلامة اللاحقة. عندما يكون لدينا تعبير علامة قياسية بسيط، فسيكون من المفاجأ ان تعرف ان تعبير العلامة القياسية ليس بسيط كما يبدو خصوصا عندما ترغب بإيجاد نتيجته .لغرض حساب قيمة تعبير العلامة القياسية فانك تحتاج الى الاعتماد على افضلية العوامل والصفات المشتركة معه، مثال: التعبير 4*5+3 يحسب على انه 32 وهو يعني , 4*(5+3) او يحسب 23 بمعنى(4*5)+3 . ولحل هذه المشكلة فان الاسبقيات او الافضليات للعوامل تم تحديدها ,حيث ان اسبقية العوامل تتحكم بترتيب حساب القيم للعمليات الحسابية المختلفة ضمن التعبير الحسابي الواحد .العامل ذو الاسبقية الاعلى يتم حسابة او تطبيقه قبل العامل ذو الاسبقية الاوطأ. فتعبير العلامة القياسي من الصعب تحليله وذلك:

- لأنه يحتاج الى تحديد افضلية العوامل ,المحددات ,وكسر التعادل(اي عندما يكون عاملين بنفس الافضلية ,) وهذا يجعل تقييم الحاسوب للتعبير صعب اكثر من اللازم.
- بينما تعبير العلامة السابقة وتعبير العلامة اللاحقة لا يعتمد على افضلية العوامل ,المحددات, وكسر التعادل .لذا فان تقييم التعابير بهذه الصيغ يكون اسهل.

تحويل تعبير العلامة القياسية الى تعبير العلامة اللاحقة

خوارزمية تحويل تعبير العلامة القياسي الى تعبير العلامة اللاحقة

1- يعامل تعبير العلامة القياسي على انه سلسلة من الرموز(معاملات وعوامل رياضية) ويتم فحصها او قراءة هذه الرموز واحد بعد الاخر من اليسار الى اليمين.
2- يتم استخدام المكدس كخزان وسطي يكون ابتداء فارغ ,وسلسلة للمخرجات تكون بداية خالية من أي رمز
3- افحص العنصر اللاحق بالمدخلات(اي ضمن تعبير العلامة القياسية) .
4- اذا كان هذا العنصر معامل(متغير او ثابت)يتم وضعة في سلسلة المخرجات مباشرة
5- اذا كان قوس مفتوح يتم دفعة الى المكدس.
6- اذا كان عامل رياضي قم بما يلي:
   a. اذا كان المكدس فارغ ادفع العامل الى المكدس .اذهب الى الخطوة 7
   b. اذا كان موجود بأعلى المكدس هو قوس مفتوح ادفع العامل الى المكدس .اذهب الى الخطوة7
   c. اذا كان هذا العامل له اسبقية اعلى من العامل في اعلى المكدس ادفع هذا العامل الى المكدس.اذهب الى الخطوة7
   d. اذا كان هذا العامل له اسبقية اوطأ من العامل الموجود في اعلى المكدس، في هذه الحالة اسحب العامل من اعلى المكدس واخرجة ليتم وضعة في سلسلة المخرجات، اذهب الى الخطوة a في بداية الفقرة 6
7- اذا كان قوس مغلق اسحب العوامل من المكدس ويتم وضعهم في سلسلة المخرجات واحد بعد الاخر، لحين تصل الى القوس المفتوح .عملية السحب ستهمل القوس المفتوح.
8- اذا كان هناك مزيدا من المدخلات اذهب الى الخطوة1
9- اذا لم يكن هناك مزيد من المدخلات، اسحب العوامل المتبقية في المكدس واضفهم الى سلسلة المخرجات.

أمثلة لتحويل تعبير العلامة القياسية الى تعبير العلامة اللاحقة (هنا باستخدام الاسبقيات وليس المكدس)

مثال 1 :- تعبير العلامة القياسية a+b*c

اسبقية عامل الضرب اكبر من اسبقية عامل الجمع

تحويل الضرب الى تعبير العلامة اللاحقة // a+(b*c)
تحويل الجمع الى تعبير العلامة اللاحقة // a+(bc*)
ازالة الاقواس // a+(bc*)+
شكل تعبير العلامة اللاحقة : abc*+ (لاحظ عدم الحاجة الى الاقواس في تعبير العلامة اللاحقة)

مثال 2 :- تعبير العلامة القياسية (A+B)*C

لاحظ هنا أن اسبقية الاقواس اكبر من اسبقية العمليات الاخرى (اكبر من اسبقية عامل الضرب)
تحويل الجمع الى تعبير العلامة اللاحقة // (AB+)*C
تحويل الضرب الى تعبير العلامة اللاحقة // (AB+)C*
ازالة الاقواس // AB+C*
شكل تعبير العلامة اللاحقة : AB+C* (لاحظ عدم الحاجة الى الاقواس في تعبير العلامة اللاحقة)

مثال 3 :- تعبير العلامة القياسية a+((b*c)/d)

اسبقية علامة الضرب وعلامة القسمة متساويتان لذلك تكون الاسبقية للموجودة على اليسار ، لكن في هذا المثال الاسبقية للاقواس

تحويل الضرب الى تعبير العلامة اللاحقة // a+((bc*)/d)
تحويل القسمة الى تعبير العلامة اللاحقة // a+(bc*d/)
تحويل الجمع الى تعبير العلامة اللاحقة // a (bc*d/)+
ازالة الاقواس // abc*d/+
شكل تعبير العلامة اللاحقة : abc*d/+ (لاحظ عدم الحاجة الى الاقواس في تعبير العلامة اللاحقة)

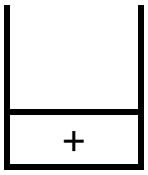
مثال 4 :- تعبير العلامة القياسية ((a+b)*c-d)*e

تحويل الجمع الى تعبير العلامة اللاحقة // (ab+*c-d)*e
تحويل الضرب الى تعبير العلامة اللاحقة //(ab+c*-d)*e
تحويل الطرح الى تعبير العلامة اللاحقة //ab+c*d-*e
تحويل الضرب الى تعبير العلامة اللاحقة // ab+c*d-e*

شكل تعبير العلامة اللاحقة : ab+c*d-e* (لاحظ عدم الحاجة الى الاقواس في تعبير العلامة اللاحقة)

مثال5: تحويل تعبير العلامة القياسية الى تعبير العلامة اللاحقة باستخدام المكدس : a+b*c-d

ابتداء فان المكدس يكون فارغ وسلسلة التعبير اللاحق ليس فيها اي رموز .الان ,الرمز الاول الذي يتم فحصه هو 'a' والذي ستتم إضافته الى سلسلة تعبير العلامة اللاحقة .الرمز الثاني الذي تم فحصة هو علامة الجمع '+' وهذا بالطبع عامل رياضي وسيتم إضافته الى المكدس.

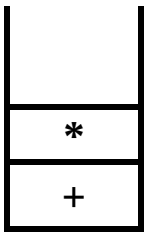


المكدس

a

سلسلة تعبير العلامة اللاحقة

الرمز التالي الذي سيتم قراءته هو الرمز 'b' والذي سيوضع في سلسلة تعبير العلامة اللاحقة .بعدها يكون الدور للرمز الذي سيتم قراءته هو علامة الضرب '*' والذي هو عامل رياضي .الان هذا يجب ان يودع في المكدس ويجب ان نلاحظ ان اعلى المكدس يحتوي العامل الرياضي '+' والذي هو ذو اسبقية اقل من عامل الضرب ,لذلك فان عامل الضرب سيدفع الى اعلى المكدس.

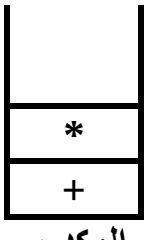


المكدس

ab

سلسلة تعبير العلامة اللاحقة

الرمز اللاحق هو 'c' والذي سيوضع في تعبير العلامة اللاحقة.



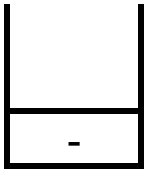
المكدس

abc

سلسلة تعبير العلامة اللاحقة

الرمز الاخر الذي سيتم قراءته هو علامة الطرح '-' وهذا عامل رياضي المفروض ان يوضع في المكدس ,لاحظ ان اعلى المكدس يحتوي علامة الضرب '*' والذي هو ذو اسبقية اعلى من عامل الطرح ونظرا لعدم امكانية ان يوضع عامل رياضي ذو اسبقية دنيا فوق عامل رياضي ذو اسبقية اعلى في المكدس لذا سيتم سحب العامل '*' من المكدس ويضاف الى سلسلة تعبير العلامة اللاحقة .حتى الان فان المكدس ليس فارغ .الان بعد سحب علامة الضرب فان علامة

الجمع '+' هي في اعلى المكدس وهذا العامل مساوي بالأسبقية لعامل الطرح ,ولذا فان عاملين متساويين بالأسبقية لا يمكن ان يكونا احدهما فوق الاخر مباشرة بالمكدس لذا فان العامل '+' يسحب ايضا من المكدس ويوضع في سلسلة تعبير العلامة اللاحقة ,عند ذاك يتم دفع العامل '-' الى المكدس.

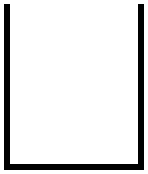


المكدس

abc*+

سلسلة تعبير العلامة اللاحقة

الرمز الذي سيقرأ الان هو الرمز 'd' والذي سيضاف الى تعبير العلامة اللاحقة , وبعد قراءة الرمزd نكون قد قرأنا جميع الرموز في تعبير العلامة القياسية وبهذا وصلنا الى حالة هي عدم وجود اي رمز اخر لم يتم قراءته وعلية فان جميع العناصر الموجودة في المكدس سيتم سحبها واحد بعد الاخر ,وتتم اضافتها الى سلسلة تعبير العلامة اللاحقة بالتعاقب .فيتم هنا سحب عامل الطرح ويضاف الى سلسلة تعبير العلامة اللاحقة .لذا بعد ان يتم قراءة جميع الرموز فان المكدس وسلسلة تعبير العلامة اللاحقة سيكونان كما في الشكل أدناه:



المكدس

abc*+d-

سلسلة تعبير العلامة اللاحقة

النتيجة النهائية ستكون :

abc*+d-

امثلة متنوعة لتحويل تعبير العلامة القياسي الى تعبير العلامة اللاحقة

مثال 1 : تحويل تعبير العلامة القياسية a + b * c الى تعبير العلامة اللاحقة ؟

a + b * c		
عناصر تعبير العلامة القياسية	المكدس	سلسلة المخرجات تعبير العلامة اللاحقة
a	فارغ	a
+	+	a
b	+	ab
*	+*	ab
c	فارغ	abc*+
تعبير العلامة اللاحقة	a b c * +	

مثال 2 : تحويل تعبير العلامة القياسية C * (A + B) الى تعبير العلامة اللاحقة ؟

(A + B) * C		
عناصر تعبير العلامة القياسية	المكدس	سلسلة المخرجات تعبير العلامة اللاحقة
(	(	
A	(	A
+	(+	A
B	(+	AB
)	فارغ	AB+
*	*	AB+
C	فارغ	AB+C*
تعبير العلامة اللاحقة	A B + C *	

مثال 3 : تحويل تعبير العلامة القياسية (d / (b * c)) + a الى تعبير العلامة اللاحقة ؟

a + ((b * c) / d)		
عناصر تعبير العلامة القياسية	المكدس	سلسلة المخرجات تعبير العلامة اللاحقة
a	فارغ	a
+	+	a
(	+(	a
(	+((	a
b	+((	ab
*	+((*	ab
c	+((*	abc
)	+(	abc*
/	+(/	abc*
d	+(/	abc*d
)	فارغ	abc*d/+
تعبير العلامة اللاحقة	abc*d/+	

مثال 4 :تحويل تعبير العلامة القياسي 3*5+(2-1)/3*2 الى تعبير العلامة اللاحقة باستخدام المكدس؟

3*5+(2-1)/3*2		
سلسلة المخرجات تعبير العلامة اللاحقة	المكدس	عناصر تعبير العلامة القياسية
2	فارغ	2
2	*	*
23	*	3
23*	/	/
23*	/(	(
23*2	/(	2
23*2	/(-	-
23*21	/(-	1
23*21-	/	)
23*21-/	+	+
23*21-/5	+	5
23*21-/5	+*	*
23*21-/53	+*	3
23*21-/53*+	فارغ	
23*21-/53*+	تعبير العلامة اللاحقة	

مثال 5 : تحويل تعبير العلامة القياسي 1+2*3^4/5-6 الى تعبير العلامة اللاحقة باستخدام المكدس ؟

1+2*3^4/5-6		
سلسلة المخرجات تعبير العلامة اللاحقة	المكدس	عناصر تعبير العلامة القياسية
1	فارغ	1
1	+	+
12	+	2
12	+*	*
123	+*	3
123	+*^	^
1234	+*^	4
1234^*	+/	/
1234^*5	+/	5
1234^*5/+	-	-
1234^*5/+6	-	6
1234^*5/+6-	فارغ	
1234^*5/+6-	تعبير العلامة اللاحقة	

مثال 6 : حول تعبير العلامة القياسي ((2+2)*(3-2))/(5-6)*(8+4) الى تعبير العلامة اللاحقة

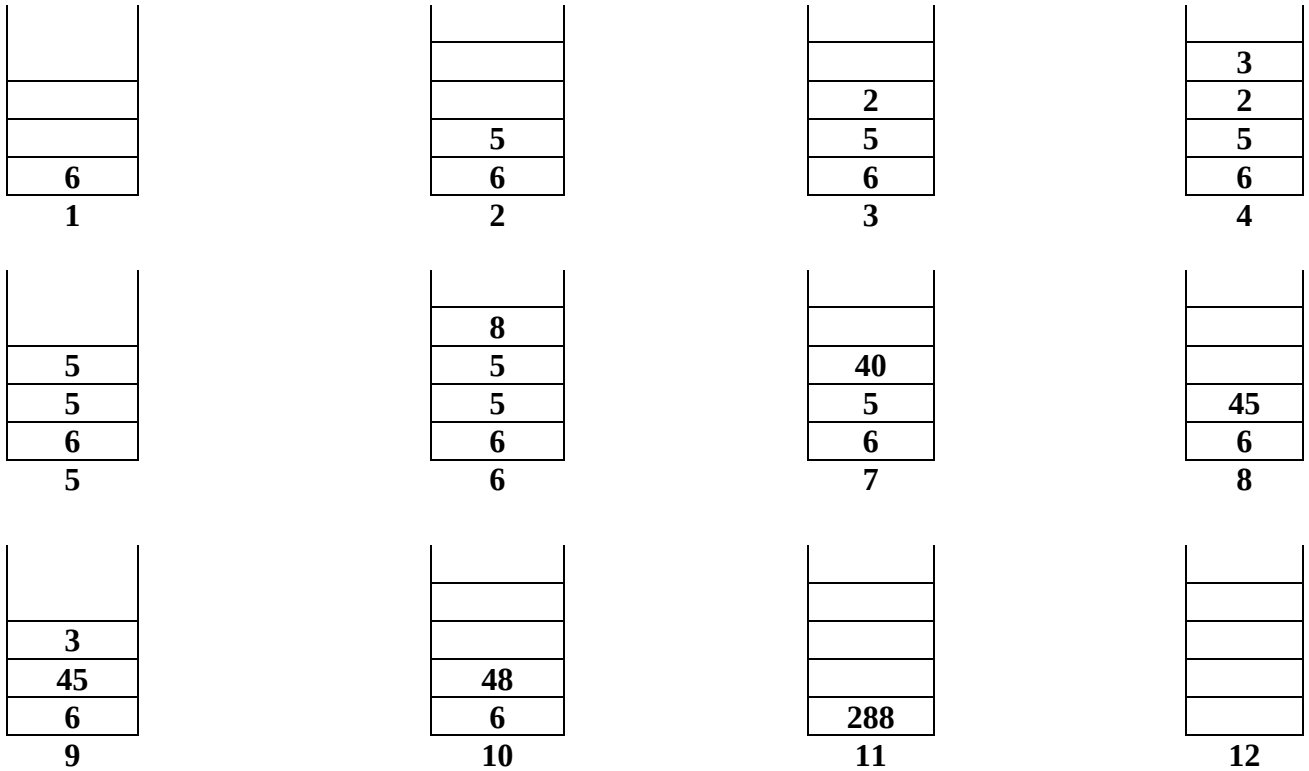
((2+2)*(3-2))/(5-6)*(8+4)		
سلسلة المخرجات تعبير العلامة اللاحقة	المكدس	عناصر تعبير العلامة القياسية
	(	(
4	(	4
4	(+	+
48	(+	8
48+	فارغ	)
48+	*	*
48+	*(	(
48+6	*(	6
48+6	*(-	-
48+65	*(-	5
48+65-	*	)
48+65-*	/	/
48+65-*	/(	(
48+65-*	/((	(
48+65-*3	/((	3
48+65-*3	/((-	-
48+65-*32	/((-	2
48+65-*32-	/(	)
48+65-*32-	/(*	*
48+65-*32-	/(*(	(
48+65-*32-2	/(*(	2
48+65-*32-2	/(*(+	+
48+65-*32-22	/(*(+	2
48+65-*32-22+	/(*	)
48+65-*32-22+*	/	)
48+65-*32-22+*/	فارغ	
48+65-*32-22+*/		تعبير العلامة اللاحقة

حساب (نتيجة) تعبير العلامة اللاحقة باستخدام المكدس

بالإمكان حساب نتيجة تعبير العلامة اللاحقة باستخدام المكدس

1- ابدأ بمكدس فارغ
2- تعبير العلامة اللاحقة يتكون عادة من معاملات وعوامل رياضية بالإمكان اعتبارها كسلسلة من الرموز يتم قراءتها أو فحصها من اليسار الى اليمين تباعا (رمز كل مرة) .
3- يتم قراءة رمز واحد من السلسلة في كل مرة .
4- اذا كان الرمز هو معامل يتم دفعة الى داخل المكدس ..
5- اذا كان الرمز عامل رياضي عندها يتم ما يلي :
   a. سحب اثنين من المعاملات من اعلى المكدس (أي المعامل الاعلى في المكدس والمعامل الذي يليه).
   b. اجراء العملية الحسابية على المعاملين الذين تم سحبهما ووفقا للعامل الرياضي الذي تم قراءته في الخطوة 5
   c. يتم دفع النتيجة المستحصلة من الخطوة b الى داخل المكدس (توضع عادة في اعلى المكدس) .
6- اذا لم تكن السلسلة فارغة الانتقال الى الخطوة 2

مثال 1: إيجاد نتيجة تعبير العلامة اللاحقة التالي : 6523+8*+3+*



يتم التعامل مع تعبير العلامة اللاحقة على أنه سلسلة من الرموز ، وفي كل مرة يقرأ أحد هذه الرموز وذلك من اليسار الى اليمين اولا يقرأ الرقم 6 ويدفع الى داخل المكدس الفارغ (الارقام (المعاملات) تدفع الى المكدس) بعدها يأتي الرقم 5 فيدفع الى المكدس ايضا ثم يأتي الرقم 2 ويدفع الى المكدس ايضا حسب القاعدة الرمز الاخر هو الرقم 3 ويدفع ايضا الى المكدس بعد دفع الرقم 3 الى المكدس نصل الى علامة الجمع وكما اسلفنا أن العوامل الرياضية لا تدفع الى المكدس وما يحدث هو ان يتم سحب رقمين من اعلى المكدس واجراء هذه العملية الرياضية عليهما هنا في اعلى المكدس الرقم 3 وبعده الرقم 2 يتم سحبهما من المكدس وتجرئ عليهم عملية الجمع ليكون الناتج 5 هذا الناتج يعاد دفعة الى المكدس وطبعا سيكون في اعلى المكدس (الخطوة 5) تستمر عملية قراءة رموز تعبير العلامة اللاحقة فتقرا الرقم 8 وهذا رقم يدفع فيدفع الى المكدس كما في الخطوة (6) وبعدها نقرا علامة الضرب وبهذا يتم سحب رقمين من اعلى المكدس وهما الرقم 8 والذي يليه الرقم 5 لتتم عليهم عملية الضرب ويكون الناتج 40 الذي يدفع الى اعلى المكدس (خطوة 7) تستمر عملية قراءة رموز تعبير العلامة اللاحقة حيث علامة الجمع عندها يسحب اعلى رقمين وهم 40

والرقم 5 وتجرى عملية الجمع والناتج هو 45 يدفع الى اعلى المكدس ,بعدها يتم قراءة الرقم 3 ويدفع الى اعلى المكدس ثم نقرأ علامة الجمع ونسحب الرقم 45 والرقم 3 من اعلى المكدس وتجرى عليهم عملية الجمع ليكون الناتج 48 يدفع الى اعلى المكدس الخطوة (10) واخيرا تتم عملية قراءة عامل الضرب ويتم سحب الرقم 48 والرقم 6 لإجراء عملية الضرب وينتج الرقم 288 الذي يدفع الى المكدس وبهذا تنتهي العملية الرياضية نظرا لانتهاء جميع رموز تعبير العلامة اللاحقة, ولم يبقى بالمكدس سوى الناتج الذي هو 288 .

العملية تتوقف عندما تنتهي جميع العوامل الرياضية والمعاملات في السلسلة (تعبير العلامة اللاحقة) نتيجة حساب التعبير المتحصلة يتم سحبها من المكدس وستكون هي الوحيدة في المكدس اي معامل واحد يبقى بالمكدس.

مثال 2 : إيجاد نتيجة تعبير العلامة اللاحقة باستخدام المكدس +3^2*+/382-+623

623+-382/+*2^3+		
الرمز	المكدس	الملاحظات
6	6	معامل يدفع الى المكدس
2	6 , 2	معامل يدفع الى المكدس
3	6 , 2 , 3	معامل يدفع الى المكدس
+	6 , 5	عامل رياضي, يتم سحب اعلى معاملين من المكدس , وتجرى عليهم العملية الرياضية وفقا للعامل الرياضي هنا عملية جمع (2+3) ثم تعاد النتيجة (5) الى اعلى المكدس
-	1	عامل, يسحب اعلى معاملين ويجرى عليهم عملية الطرح (6-5) وتدفع النتيجة (1) الى اعلى المكدس
3	1 , 3	معامل يدفع الى المكدس
8	1 , 3 , 8	معامل يدفع الى المكدس
2	1 , 3 , 8 , 2	معامل يدفع الى المكدس
/	1 , 3 , 4	القسمة (8/2) وتدفع النتيجة (4) الى اعلى المكدس ملاحظة: المعامل الاول الذي يسحب من المكدس يوضع بعد العامل الرياضي بينما المعامل الثاني يوضع قبل العامل الرياضي
+	1 , 7	جمع المعاملين (جمع المعاملين (4+3) ودفع النتيجة (7) الى اعلى المكدس
*	7	عملية ضرب لأعلى معاملين بالمكدس (1*7) ودفع النتيجة (7) الى اعلى المكدس
2	7 , 2	معامل يدفع الى المكدس
^	49	عامل الرفع (اس) لأعلى معاملين بالمكدس (2^7) وتدفع النتيجة (49) الى اعلى المكدس
3	49,3	معامل يدفع الى المكدس
+	52	اجراء عملية الجمع على اعلى معاملين بالمكدس (3+49) ودفع النتيجة (52) الى اعلى المكدس, وهي اخر عملية وتمثل النتيجة النهائية
	52	نتيجة حساب تعبير العلامة اللاحقة

تمثيل عمل المكدس بواسطة الهياكل الاستاتيكية (المصفوفات)

يعد المكدس جزء من الذاكرة الرئيسية للحاسوب، ولتوضيح طريقة عمل هذا الجزء الهام من الذاكرة يمكننا عمل برنامج محاكاة يوضح آلية عمل المكدس. ويمكن تمثيل عمل المكدس باستخدام المصفوفات الأحادية، ولكن المصفوفات تعد من الهياكل الاستاتيكية لابد من معرفة حجم المصفوفة مسبقاً، ووفقاً لذلك فعملية الاضافة للمكدس تتم بشرط أن المكدس غير ممتلئ، كما تتم عملية الحذف من المكدس بشرط وجود قيم بداخل هذا المكدس. وبما أن العمليات التي يمكننا عملها على المكدس هي فقط الاضافة والحذف، ومن اتجاه واحد فقط يمكننا عمل برنامج سنحتاج الدوال التالية لكي نمثل آلية عمل المكدس

- Push تعني وضع عنصر داخل المكدس (اضافة)
- Pop تعني أخذ عنصر من المكدس (حذف)

كذلك سنحتاج الى مؤشر Top الذي يزيد بمقدار واحد كلما قمنا بعملية الاضافة ويقل بمقدار واحد كلما قمنا بعملية الحذف .

ألية عمل المؤشر (Top)

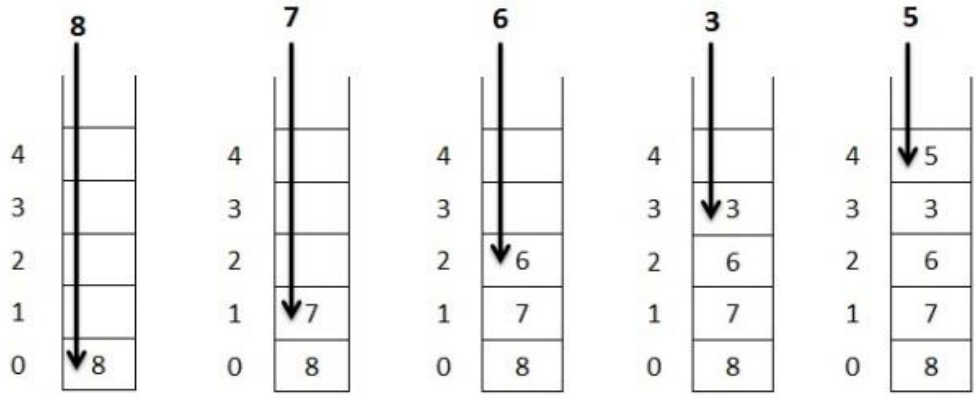
وعندما يكون المكدس فارغا تكون قيمة المؤشر top=-1 وعند ادخال أول عنصر في المكدس تكون قيمة المؤشر top=0 وعند ادخال القيمة الثانية تكون قيمة top=1 بمعنى أن تتم زيادة قيمة الـ top بمقدار واحد في كل مرة تتم فيها اضافة عنصر الى المكدس الى أن يمتلأ المكدس ويكون المكدس ممتلئ عندما تكون top==size

عندما تكون العملية حذف فانه يتم انقاص مؤشر top بمقدار واحد في كل مرة يتم فيها أخذ قيم من المكدس الى أن يتم افراغ المكدس ويكون المكدس فارغا عندما top=-1

عملية أدخال القيم للمكدس

حتى نتمكن من اضافة قيم للمكدس نتبع الخوارزمية التالية :-

1- نمثل عمل المكدس باستخدام مصفوفة تتسع لخمس قيم صحيحة مثلاً ، وتعرف على النحو : int stack[5];
2- نفترض أن المطلوب ادخال القيم التالية للمكدس {8,7,6,3,5}



الشكل (2) يوضح عملية ادخال القيم للمكدس

3- نحدد قيمة مؤشر المكدس في البداية بالقيمة (Top=-1) للدلالة بأن المكدس خالي من القيم (لا يوجد قيم).
4- قبل الشروع بإدخال قيم جديدة للمكدس، يجب السؤال عن الوضع الحالي للمكدس هل المكدس ممتلئ أم لا؟ ويمتلئ المكدس عند تخزين آخر قيمة مدخلة في عنوان آخر خلية، أي أن المكدس يصبح ممتلئاً عند تحقق الشرط التالي:-

Top=size-1

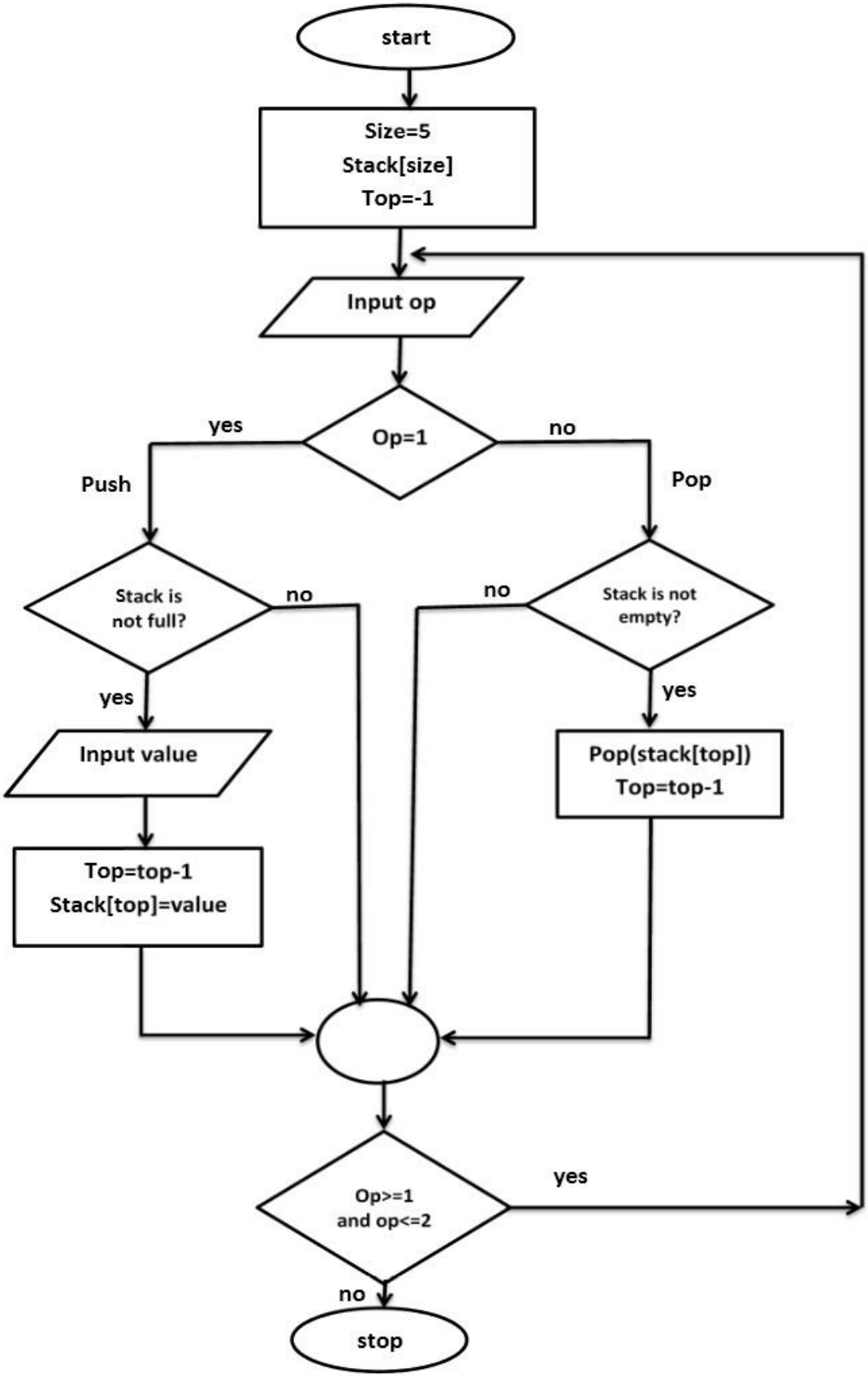
5- حيث أن size يمثل حجم المصفوفة ولأن عناوين المصفوفة تبدأ من 0 كما تم ذكره ، فإن آخر عنصر يتم حجزه في المكدس يكون عند الخلية ذات العنوان size-1
6- إدخل القيمة 8 للمكدس ، تخزن القيمة كأول عنصر في المكدس ، ويزداد قيمة المتغير Top بمقدار واحد وتصبح قيمة مؤشر المكدس Top تساوي 0
7- يتم تكرار إدخال القيم للمكدس بتكرار الخطوات 4,5 ، يزداد قيمة المتغير Top بمقدار واحد بعد كل إضافة للمكدس فعند إدخال القيمة 5 للمكدس الممثل في الشكل (2) ، تزداد قيمة المتغير Top بمقدار واحد وتصبح قيمة مؤشر المكدس تساوي 4 حينها لا يمكن إضافة قيم جديدة حيث شرط امتلاء المكدس (Top=size-1) .

عملية حذف القيم من المكدس

من العمليات التي تتم على البيانات المخزنة في المكدس، عملية الحذف لكن الحقيقة أن الهياكل الثابتة لا تحذف البيانات بشكل حقيقي، بل يتم تحريك مؤشر المكدس نحو الأسفل، ولحذف قيم المكدس يجب اتباع الخوارزمية التالية:

1- يجب التأكد من وجود قيم في المكدس (هل المكدس فارغ أم لا ؟) يكون المكدس خالي من القيم عند تحقق الشرط Top=-1 .
2- يتم حذف قيم المكدس من قمته فأول قيمة يمكن حذفها هي القيمة 5 ، وبحذف أول عنصر نتجه بمؤشر المكدس (Top) إلى الأسفل (Top=Top-1) .
3- نكرر الخطوات 1,2 بهدف حذف القيم من المكدس، عند إخراج القيمة 8 من المكدس يصبح المكدس فارغ، أي Top=-1 .

ويمكن تمثيل خوارزمية العمل للمكدس بواسطة المخطط الانسيابي الموضح بالشكل (3) .



الشكل (3) المخطط الانسيابي للعمليات التي تتم على المكدس.

شرح آلية عمل المكدس باستخدام المصفوفات

```
#include<iostream>
using namespace std;
 const int size=5;
   int mystack[size],top=-1;
   int is_full()
   {
      if(top==size-1)
         return 0;
      else
         return 1;

   }
    int is_empty2()
   {
       if(top==-1)
         return 0;
      else
         return 1;
   }

void push(int val)
{
   if(is_full()==1)
   mystack[++top]=val;
   else
   cout<<"stack is Full\n";
}

void pop()
{
   if(is_empty2()==1)
   cout<<mystack[top--]<<endl;
   else
     cout<<"stack is empty\n";
}
```

```
main()
{
int op,x;
cin>>op;
while(op>=1 && op<=2)
{
   switch(op)
   {
   case 1:
     cin>>x;
     push(x);
      break;
   case 2:
      pop();
      break;
   }
   cin>>op;
}
}
```

هيكلة البيانات الثابتة:-

نعني بهيكلة البيانات، عملية تنظيم البيانات بحيث يتسنى العمل عليها بيسر وتتم عملية الهيكلة من خلال فرز البيانات ذات الصفات المشتركة التي ترتبط فيما بينها بعلاقة ما، وتجمع هذه البيانات في قوالب (هياكل، سجلات) حتى يصبح من السهل التعامل معها والتسريع من العمليات التي تتم عليها بهدف سرعة الحصول على النتائج.

إن السجل أو الهيكل (struct) يعد واحدا من أنواع البيانات المركبة المعرف في جميع لغات البرمجة ويعرف بأنه قالب يحتوي على مجموعة من البيانات تشترك فيما بينها بعلاقة.

هيكلة بيانات المكدس

بالعودة إلى آلية عمل المكدس والذي تم تمثيلة بواسطة الهياكل الإستاتيكية، نلاحظ بأن المكدس قد تم تمثيله بواسطة مصفوفة معرفة الحجم، كما أن العمليات التي تتم على المكدس محدودة وتقتصر على عمليتي الاضافة والحذف من قمة المكدس (push , pop) بما أن المكدس تم تمثيلة بواسطة مصفوفة لذا فإنه عند إجراء أي من العمليات السابقة لا بد من التأكد قبل ادخال قيم جديدة في المكدس من أن المكدس ممتلئ أم لا ؟ وعلى إثر ذلك يمكن الاضافة أو لا، كذلك في حالة حذف أي قيمة من المكدس لابد من التأكد من وجود قيم بداخل المكدس أم لا ؟ ويمكن لنا معرفة هل المكدس ممتلئ أم لا ؟ وهل يوجد قيم في المكدس أم لا ؟ من خلال استخدام مؤشر للمكدس (top) تتغير قيمة المؤشر بالزيادة عند ادخال قيم جديدة للمكدس والنقصان عند حذف القيم من المكدس يمكننا هذا المؤشر معرفة ما اذا كان المكدس ممتلئا أو فارغا . وبذا يمكن القول بأن كل قيمة في المكدس مرتبطة بمؤشر يوضح عنوان هذه القيمة في المكدس أي أن كل عنصر من عناصر المكدس يمثل بقيمة وعنوان لهذه القيمة وبحسب هذه العلاقة بين عناصر المكدس يمكن تمثيل هذه العناصر باستخدام الهيكل(السجل)

```
struct stack2
{
int ar[size];
int top;
};
```

وبهذه الطريقة يتم حجز مساحة واحدة في الذاكرة لمكونات المكدس مما يمكننا من التعامل مع مكونات المكدس، ككتلة بيانات واحدة من السهل التعامل معها وعمل مختلف العمليات على هذه البيانات. ولكي نمثل طريقة عمل المكدس نحن بحاجة إلى مصفوفة ومؤشر يوضح عدد العناصر الموجودة في المكدس، وبدلاً من تعريف متغير خاص بالمصفوفة في مكان ما في الذاكرة ومتغير خاص كمؤشر للمكدس في مكان آخر سنقوم بحجز موقعين متتاليين للمتغيرات الخاصة بالمكدس على النحو التالي :-

هيكلة بيانات المكدس باستخدام الهيكل Struct

```
#include<iostream>
using namespace std;
const int size=5;
struct stack2
{
int ar[size];
int top;
};
stack2 mystack;

int is_full()
  {
if(mystack.top==size-1)
 return 0;
 else
 return 1;
  }
int is_empty2()
  {
if(mystack.top==-1)
return 0;
else
return 1;
  }

void push(int x)
{
mystack.ar[++mystack.top]=x;
}

int pop()
{
return mystack.ar[mystack.top--];
}
```

```
main()
{
  mystack.top=-1;
int type,x;
cin>>type;
while(type>=1&&type<=2){
switch(type)
{
case 1:
  if(is_full()==1)
  {
cin>>x;
push(x);
break;
}
else{
cout<<"stack is full\n";
break;
}
case 2:
if(is_empty2())
{
  cout<<pop()<<endl;
break;
}
else
{
cout<<"stack is empty\n";
break;
}
}
cin>>type;
}
}
```

تمثيل عمل المكدس باستخدام القوائم الأحادية المتصلة

يعد تمثيل عمل المكدس بواسطة القوائم المتصلة أسهل بكثير من تمثيلة باستخدام المصفوفات، وكون المكدس عبارة عن مخزن للبيانات أما العمليات التي تتم على المكدس محدودة وذلك يعود الى طبيعة عمل المكدس، فعملية الإضافة التي تتم على المكدس مشروطة أن تتم الاضافة من قمة المكدس، وعملية حذف البيانات من المكدس يجب أن تتم أيضاً من قمة المكدس ولتنفيذ هذه العمليات على المكدس يجب وكخطوة أولى هيكلة بيانات المكدس بواسطة السجلات :

```
Struct stack2
{
int data;
stack2 *last;
};
```

يتكون المكدس من مجموعة من العقد المترابطة، كل عقدة تتكون من جزئيين جزء خاص بالبيانات وجزء آخر يتم بواسطته ربط العقدة الجديدة بالمكدس، وكون المكدس يعمل وفق قانون الداخل أولاً الخارج أخيراً (FILO) لذا فكل عقدة في المكدس يجب أن تشير للعقدة السابقة.

عملية الاضافة في المكدس

عملية الإضافة للمكدس تتم من القمة، وبدون أي شروط بعكس عملية الإضافة للمكدس الممثل بواسطة المصفوفات، ولإضافة قيم جديدة للمكدس نتبع الخطوات التالية:

1- تعريف مؤشر يشير لقمة المكدس على النحو التالي:

```
stack2 *top=NULL;
```

2- تكوين أول عقدة في المكدس

```
stack2*np=new stack2;
```

3- تدخل البيانات للعقدة، وكون هذه هي أول عقدة فإن مؤشر العقدة لا يشير الى شيء

```
cin>>np->data;
np->last=top; // ربط العقدة الجديدة بالسابقة
```

4- نجعل المؤشر top يشير للعقدة الأولى، والتي تعد قمة المكدس في الوقت الحالي:

```
top=np; // نقل مؤشر المكدس ليشير لآخر عقدة
```

5- لإضافة قيمة جديدة للمكدس نقوم بإنشاء عقد جديدة بتكرار الخطوة 2,3,4

عملية الحذف من المكدس

عملية الحذف تتم من قمة المكدس، تتم عملية الحذف طالما يوجد بيانات في المكدس (تشبه عملية الحذف من نهاية القائمة المتصلة) ، ولحذف قيم من المكدس نتبع الخوارزمية التالية :

1- يتم تعريف مؤشر جديد يشير للعقدة التي يشير اليها top `stack2*p=top;`

2- يتم نقل المؤشر top لكي يشير للعقدة التالية في المكدس `top=top->last`

3- تحذف العقدة المشار اليها بالمؤشر p `delete p`

شرح آلية عمل المكدس باستخدام القوائم المتصلة

```
#include<iostream>
using namespace std;
struct stack2
{
   int data,sum=0;
   stack2 *last, *tail=NULL;
int top()
{
 return tail->data;
}

int empty2()
{
  if(tail !=NULL)
   return 0;
   else
   return 1;
}
void Push(int c)
{
   sum++;
   stack2 *np=new stack2 ;
   np->data=c;
   np->last=tail;
   tail=np;
}
void pop()
{
    stack2 *p=tail;
   if(!empty2())
   {
      tail=tail->last;
      delete p;
      sum--;
   }
   else
   cout<<"stack is empty\n";
}
};
```

```
int size2()
{
 return sum;
}
main()
{
stack2 mystack;

   int x;
  for(int i=1;i<=5;i++)
  {
     cin>>x;
    mystack.Push(x);
  }
cout<<"********************\n";

  while(!mystack.empty2())
{

   cout<<mystack.top()<<endl;
    mystack.pop();
}

}
```

برامج على المكدس

برنامج لإيجاد اكبر قيمة داخل المكدس

```
#include<iostream>
#include <stack>
using namespace std;
main()
{
  stack<int>mystack;
 int x;
  for(int i=0;i<5;i++)
  {
     cin>>x;
    mystack.push(x);
  }
  int max=mystack.top();
while(!mystack.empty())
{
   if(mystack.top()>max)
      max=mystack.top();
       mystack.pop();
}
cout<<max;
}
```

برنامج لطباعة الاعداد الزوجية من المكدس

```
#include<iostream>
#include <stack>
using namespace std;
main()
{
  stack<int>mystack;
 int x;
  for(int i=0;i<5;i++)
  {
     cin>>x;
    mystack.push(x);
  }
 while(!mystack.empty())
{
   if(mystack.top()%2==0)
      cout<<mystack.top();
   mystack.pop();
}
}
```

برنامج للتبديل بين عناصر مكدسين .

```
#include<iostream>
#include <stack>
using namespace std;
 main()
{
  stack<int>foo,bar;
  foo.push(10);foo.push(20);foo.push(30);
  bar.push(11);bar.push(22);
  swap(bar,foo);
  cout<<foo.size()<<endl;
  cout<<bar.size()<<endl;
}
```

برنامج لجمع عناصر المكدس .

```
#include<iostream>
#include <stack>
using namespace std;
main()
{
  stack<int>mystack;
  int sum=0;
  for(int i=1;i<=10;i++)
   mystack.push(i);
  while(!mystack.empty())
 {
     sum+=mystack.top();
     mystack.pop();
  }
  cout<<"total:"<<sum<<"\n";
}
```

برنامج لعكس عناصر المكدس في نفس المكدس.

عن طريق المصفوفات

```
#include<iostream>
#include<stack>
using namespace std;
 main()
 {
   stack<int>mystack;
   int x;
   for(int i=1;i<=5;i++)
   {
      cin>>x;
      mystack.push(x);
   }

int arr[mystack.size()];


for(int i=0;i<5;i++)
{
   arr[i]=mystack.top();
   mystack.pop();
}


for(int i=0;i<5;i++)
{
   mystack.push(arr[i]);
}


cout<<"**************************\n";


 while(!mystack.empty())
 {
    cout<<mystack.top()<<endl;
    mystack.pop();
 }

}
```

عن طريق المكدسات

```
#include<iostream>
#include <stack>
using namespace std;
main()
{
  stack<int>mystack,mystack2,mystack3;
  int x;
   for(int i=0;i<5;i++)
   {
      cin>>x;
     mystack.push(x);
   }


while(!mystack.empty())
{
  mystack2.push(mystack.top());
  mystack.pop();
}


while(!mystack2.empty())
{
  mystack3.push(mystack2.top());
  mystack2.pop();
}


while(!mystack3.empty())
{
  mystack.push(mystack3.top());
  mystack3.pop();
}


while(!mystack.empty())
{
cout<<mystack.top()<<endl;
  mystack.pop();
}
}
```

برنامج لحذف القيمة التي يريدها المستخدم من المكدس.

```
#include<iostream>
#include <stack>
using namespace std;
main()
{
  stack<int>mystack,mystack2;
  int data,value;
   for(int i=0;i<5;i++)
   {
      cin>>data;
     mystack.push(data);
   }
cout<<"enter the value delete\n";
cin>>value;

while(!mystack.empty())
{
   if(value==mystack.top())
   {
     mystack.pop();
     break;
   }
   else
   {
      mystack2.push(mystack.top());
      mystack.pop();
   }

}

while(!mystack2.empty())
{
mystack.push(mystack2.top());
  mystack2.pop();
}

while(!mystack.empty())
{
cout<<mystack.top()<<endl;
  mystack.pop();
}
}
```

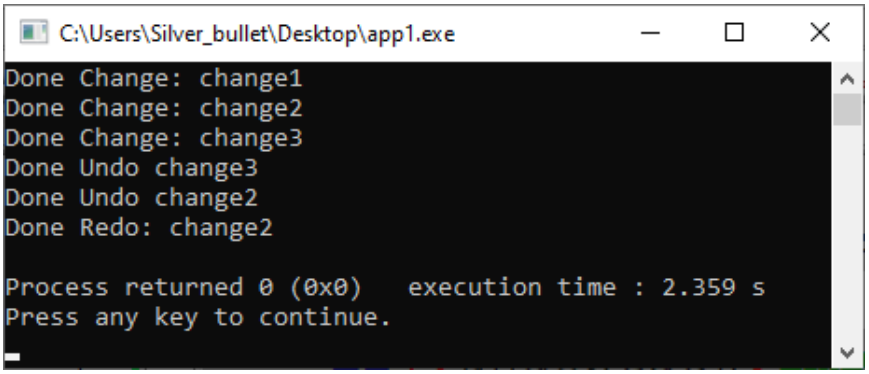
برنامج يستقبل تعبير رياضي بصيغة infix ثم يقوم البرنامج بتحويله الى postfix ؟

```
#include<iostream>
#include<stack>
#include<string>
using namespace std;
string infix_to_postfix(string exp)
{
 stack<char> stack1;
 string output="" ;
 for(int i=0;i<exp.length();i++)
 {
    if(exp[i]==' ')
      continue;
    if(isdigit(exp[i]) || isalpha(exp[i]) )
     output +=exp[i] ;
    else if(exp[i]=='(')
     stack1.push('(');
    else if(exp[i]==')'){
      while(stack1.top() !='(')
      {
         output+= stack1.top();
         stack1.pop();
      }
      stack1.pop();
    }
    else
    {
       while(!stack1.empty() &&
priority(exp[i]) <=priority(stack1.top()))
       {
         output+=  stack1.top();
         stack1.pop();
       }
       stack1.push(exp[i]);
    }
 }
 while(!stack1.empty()){
   output+= stack1.top();
   stack1.pop();
 }
 return output;
}
```

```
int priority(char c)
{
   if(c=='-' || c=='+')
      return 1;
   else if(c=='*' || c=='/')
      return 2;
   else if(c=='^')
      return 3;
   else
      return 0;
}

main()
{
string Infixexpressions;
  cout<<"input Infixexpressions :";
  cin>>Infixexpressions;
  cout<<"infix_to_postfix:";
  cout<<infix_to_postfix(Infixexpressions)<<endl;
}
```

برنامج يوضح عملية التراجع عن الخطوة والعودة للخطوة؟

```
#include<iostream>
#include<stack>
using namespace std;
stack<string>undostack,redostack;
void undo(){
   if(!undostack.empty()){
      string change=undostack.top();
      undostack.pop();
      redostack.push(change);
      cout<<"Done Undo "<<change <<endl;
         }
   else{
      cout<<"No Changes to Return from it."<<endl;
   }
}

void redo(){
if(!redostack.empty()){
   string change=redostack.top();
   redostack.pop();
   undostack.push(change);
   cout<<"Done Redo: "<<change<<endl;
}
else{
   cout<<"No changes to Return it."<<endl;
}
}

void makeChange(string change){
   undostack.push(change);
   cout<<"Done Change: "<<change<<endl;
}
main(){
makeChange("change1");
makeChange("change2");
makeChange("change3");
undo();
undo();
redo();
}
```



تمارين على المكدس

س1) أكتب برنامج يقوم بحفظ الاعداد الزوجية في أسفل المكدس والفردية في قمة المكدس؟

س2) أكتب برنامج يقوم بحذف الأعداد المتكررة من المكدس؟

س3) أكتب برنامج يقوم بترتيب عناصر المكدس تصاعدياً؟

س4) أكتب برنامج يقوم بدمج مكدسين في مكدس واحد؟

س5) أكتب برنامج يقوم بحذف الأعداد الأولية من المكدس؟

س6) أكتب برنامج يقوم بإيجاد أصغر وأكبر عنصر عنصر في المكدس؟

س7) أكتب برنامج يقوم بالبحث عن عنصر معين في المكدس؟

س8) أكتب برنامج لقراءة جملة string وطبعها بصورة معكوسة باستخدام المكدس؟

س9) أكتب برنامج لإدخال رقم واسم طالب ومعدله في مكدس وطباعة البيانات؟

س10) حول كل تعبير من تعابير العلامة القياسية أدناه الى تعبير العلامة اللاحقة باستخدام المكدس

1. **A + B * C – D * E / F**
2. **A + B ^ C / D * E + F – A**
3. **A ^ B ^ C / 2 * B + 4**
4. **A / B + C / D – E * F ^ 2 / B**
5. **((A+B)*C-D)*E**
6. **(A + B) * (C – B) / 2 ^ 4**
7. **(A + B) * (C – D)**
8. **A – B / (C + D * E)**
9. **((A + B) * C – (D – E))/(F + G)**
10. **(A + B) * (C – D)**
11. **A – B / (C + D * E)**
12. **((A + B) * C – (D – E))/(F + G)**
14. **(A-B)/C*(D+E)-F**