DATA STRUCTURES

C#

هياكل البيانات

المعلومة هي المهم

هاني الأسعد


كل القلوب الى الحبيب تميل *** ومعي بهذا شاهـــد ودليل
أما الدليل اذا ذكـــرت محمداً *** صارت دموع العارفين تسيـــل
هذا رسول الله نبراس الهـدى *** هذا لكـــل العالمين رســـول
يا حبيبي
يا محمد
يا رسول الله
محمد صلى الله عليه وسلم


إلا رسول الله
بأبي انت وامي يارسول الله
حتى الدنمارك تسخر بنبينا والناس تأكل والشراب يدار
لو قاطعوها ما تجرأ مجرم ولماتت الأغنام والأبقار
لا تكن عونا لهم
شارك في مقاطعة المنتجات الدنماركية


السيرة الذاتية
aldopaee@hotmail.com...a717@maktoob.com
الاسم: عمار محمد عيسى الدبعي
الجنسية: يمني العمر: 21 سنة محل الإقامة: الجمهورية اليمنية
المؤهلات / طالب جامعي كلية علوم وهندسة حاسوب
www.31.1hh.com

الإهداء

إلى من أعطتني الرعاية دائماً

إلى من كانت بجواري دائماً

إلى ألبذره التي طالما روتني لأصبح ثمرة

أمي الحبيبة

إلى من هواهم وأحبهم قلبيي صديقي العمر

إلى كل من يعشق نبينا محمد(صلى الله عليه وعلى اله)

إلى كل من علمني وتعب من اجلي

إلى ارض اليمن الحبيبة

تقبلوا هديتي

الفهرس

المقدمة

الحمد لله رب العالمين حمداً يوافي نعمة ويكافئ مزيدة الحمد لله الذي خلق الظلمات والنور وصلى الله على سيد الخلق معلم الناس الخير حبيبنا وشفيعنا محمد بن عبد الله النبي الأمي الذي بذا برسالته ب(اقرأ باسم ربك الذي خلق خلق الإنسان من علق إقراء وربك الأكرم الذي علم بالقلم) وختمها ب(اليوم أكملت لكم دينكم وأتممت عليكم نعمتي ورضيت لكم الإسلام دينا) صدق الله العظيم وبلغ رسوله الكريم ونحن على ذلك شاهدين من اليوم إلى يوم الدين أما بعد

أخواني وأحباب قلبي يا من حييتهم من غير أن تراهم أعيني ابدآ بالسلام والتحية المباركة آلا وهي السلام عليكم ورحمة الله وبركاته .

الحمد لله الذي من علي بهذا الفضل العظيم أن علمني بالقلم وجعلني مسلما فمنحني الإرادة لانجاز هذا الكتاب فلقد أنشأته بسبب فقدان كتب هذه المادة على الانترنت وبالغة العربية واستحالة وجودها وأنا من عانيت من هذه المشكلة والسبب الأخر تيسيراً لإخواني الطلاب الجامعيين الذين صعبت عليهم فهم هذه المادة وتبسيط لهم معنى هذه المادة وأنا متأكد أن كل من سيحصل على هذا الكتاب أنة سيفرح ويشكرني كثيراً, املآ أن ينتفع به كل من يقراه أو كانت له حاجة في علوم الحاسوب .

فيتناول هذا الكتاب موضوعات متعددة لوصف هياكل البيانات الخطية وغير خطية ولقد وثقت هذه المواضيع ببرامج علمية طبقت جميعها على لغة السي بلاس بلاس للتأكد من صحتها وأيضاً وتقت بالرسوم البيانية لترسيخ الفكر في ذهن القارئ .

وأخيراً نسأل الله أن يحقق هذا الكتاب الهدف الذي كتب لأجله ويعلم الله أن غايتي في هذا أن يعم الفائدة في ارض المسلمين وكل مسلم ومسلمة طالباً منكم دعوه صالحة في ظهر الغيب وان تصلوا وتسلموا على من علمنا وأنبانا نبينا محمد حبيب قلبنا ألف ألف مليون صلاة وسلام من رب العباد عدد تحرير السطور وعدد المخلوقات والمخلوق صلاه دائمة من اليوم إلى يوم الدين .

واسأل الله أن يبارك لنا ولكم في كل ما كتبته و تعلمته وتعلمتموه .

والحمد لله .....

مقدمة إلى هياكل البيانات

من المألوف قبل الشروع إلى فهم شيء يجب معرفة توابعه وتفاعل هذه العناصر مع بيئتها المتوافرة فيها والعمليات التي يمكن أن تحدث على هذه العناصر التي تؤلف فيما بينها وحدة متناغمة مترابطة لكي توصل الفكرة إلى العقل بشكل جيد وسريع.

ومن الأشياء التي يجب تحديدها كمتخصصين في علم الحاسب الآلي هي البيانات DATA و المعلومات INFORMATION لأنها هي أساس تعاملنا مع الحاسب.

فما هي البيانات وما هي المعلومات وكيف يمكن لنا أن نحددها ؟

فالبيانات /هي مجموعة من الحقائق أو الأفكار قد تكون حروف أو أرقام أو صوراً أو خليطاً مما سبق.

والمعلومات /مجموعة من الحقائق والأفكار عن شيء ما تمت معالجتها .

إذن فالبيانات هي المادة الخام للمعلومات أو أن المعلومات هي البيانات بعد معالجتها ومن الواضح آن الفرق الأساسي بينهم هي المعالجة .

- أنواع المعالجة على البيانات

1. الإضافة.
2. الحذف أو الإلغاء .
3. الدمج .
4. الفرز .
5. التحليل والتركيب باستخدام العمليات الحسابية (+ ,*,- ,/)والعمليات المنطقية (= ,=! ,=> ,=< ,> ,<) .
6. النسخ الالكتروني (SAVE).
7. الحماية والفك .
8. الاسترجاع والتعديل والتخزين .

إذن لكي نحصل على معلومات لابد من الحصول على بيانات أولاً ثم القيام بمعالجة هذه البيانات معالجة صحيحة.

- التركيب الفيزيائي والتركيب المنطقي للبيانات

نجد أن سرعة معالجة البيانات تعتمد كثيراً على عدة عوامل أضافه إلى العامل الزمني اللازم للمعالجة ومن أهميتها:-

- ✓ عوامل تحدد من الذاكرة الرئيسية.
- ✓ عوامل تحدد من وحدات الإدخال والإخراج.
- ✓ عوامل تحدد من تفاعل الإدخال والإخراج مع الذاكرة الرئيسية(تبادل المعلومات بين هذه الوحدات أي عملي مقايضة.

نجد أن العامل الأول يتطلب وجود برنامج للمعالجة والنسبة للعامل التأني فأنة يتطلب سرعة للوصول إلى المعلومات لإحضارها من الوحدات الإدخال , والذي يهمنا هو الإقلال من عملية المقايضة بين الذاكرة الرئيسية ووحدات الإدخال والإخراج ولهذا لابد من الإشارة إلى التركيب الفيزيائي والمنطقي للبيانات حيت نجد تعريف الاثنين باختصار

التركيب المنطقي / هو وجه نظر المبرمج في سير البرنامج أي آن هذا ترتيب معلومات البرنامج بشكل معين حتى يتم تنفيذ هذا البرنامج بطريقة صحيحة .

التركيب الفيزيائي / وهو يعني كيفية ترتيب البيانات على أوساط التخزين مثل الشريط المغناطيسي والقرص المغناطيسي حيت تخزن البيانات على القرص المغناطيسي بطريقة مباشرة أو تتابعيه أي تسلسلية مفهرسة.

هياكل البيانات وتراكيب البيانات وبني المعطيات لهما نفس المعنى /عبارة عن آليات وخوارزميات معينة توضع لبرامج بحيث تطبق بشكل جيد فهي مفيد جدا في برمجه قواعد البيانات و تساعد على تنفيذ

مهام وتسهيل مهام من مهام الكمبيوتر ومن استغلال مواقع الذاكرة بشكل جيد ومنظم ويجب على المبرمج تطيق هذه الآليات بشكل جيد وإلا خلت من معنى الخوارزمية .
وبتعريف منطقي له /هي طريقه ترابط و ترص البيانات مع بعضها البعض في الذاكرة بحيث هذه البيانات تتخذ شكلا وهيكلاً معيناً في الذاكرة فتعتبر بنية عضوية لمجموعة من عناصر البيانات المتطابقة نوعاً وشكلاً والتي تنظم في نسق واحد لتؤدي غرضاً محدداً.

- فوائد هياكل البيانات

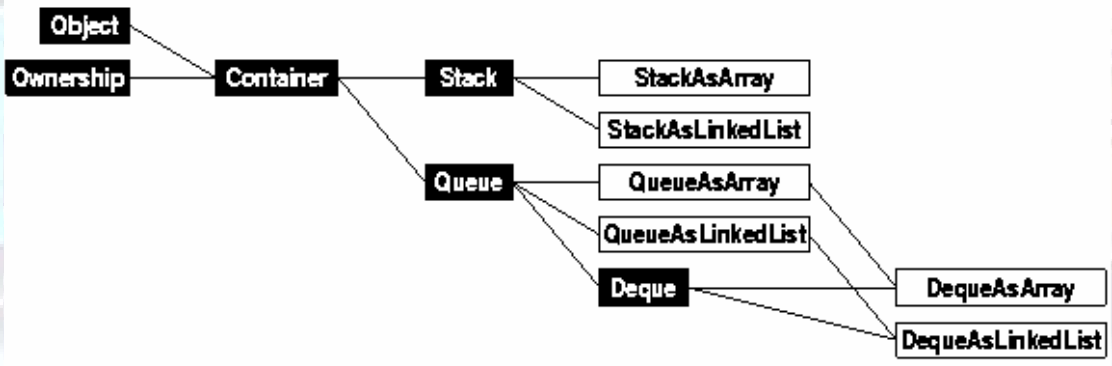
✓ التحكم في توزيع البيانات و التعرف إلى طبيعتها وبنائها الأساسي بنسق معين في الذاكرة.
✓ بناء برامج قوية ومتماسكة من حيت البناء والمنطق.
✓ تمكين المبرمج من أبداع طرق مبتكرة في كتابة البرامج المختلفة.
✓ اختصار زمن التخزين واسترجاع البيانات من الذاكرة .

- أنواع هياكل البيانات

هياكل بيانات ثابتة ساكنة (STATIC INFORMATION).
كالمتجهات والجداول والسجلات وعند التصريح عنها فيجب تحدي حجم هذه البيانات فلا تقبل الإضافة فوق حجمها المحدد .....
هياكل بيانات ديناميكية إي متحركة متغيرة
وينقسم هذا النوع إلى نوعين
1)هياكل بيانات خطية متغيرة /وهي التي تنظم في خط متتالي

❖ الملفات .
❖ القوائم .
❖ الطوابير .
❖ المكدسات .
❖ الأبجديات .
❖ المجموعات .

2)هياكل بيانات متشعبة إي بشكل عشوائي مخزنة في الذاكرة / مثل الأشجار , الخرائط

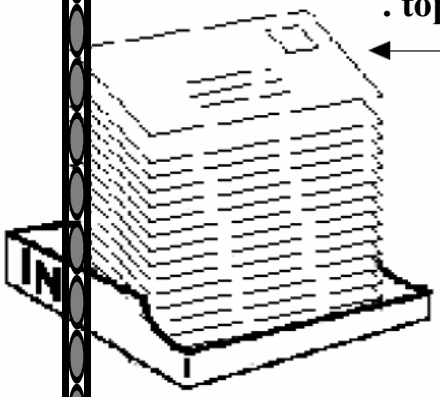


وسنتكلم عن هذه المواضيع مبدئياً بالهياكل الاستاتيكية بواسطة المتجهات .....

☒ أولاً المكدس (Stack)

المكدس / عبارة عن نموذج خاص لتخزين البيانات بالية ثابتة وإخراجها باليه ثابتة بشكل مؤقت وهو عبارة عن صندوق توضع به البيانات بالية الداخل أولا الخارج أخرا والداخل أخرا الخارج أولا LIFO (LAST INPUT FIRST OUTPUT)

وكمثال بسيط أيضا نشبه عملة بقشطه المسدس الرصاصة الأولى تخرج أخر شيء والرصاصة الأخيرة تخرج أولاً ولهذا فان الإضافة تتم من الأعلى والحذف و يوجد مؤشر واحد يسمى top .

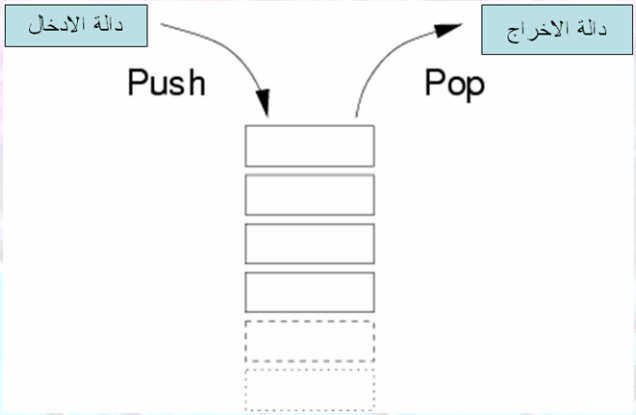


والقراءة أيضا يتم من الأعلى إي من طرف واحد عن طريق top .

- **ومن فوائد المكدس**
  - ✓ **إيجاد قيم التعابير الحسابية**
  - ✓ **يستخدم لغايات الاستدعاء الذاتي**
  - ✓ **في عمليات الاعتراض والمقاطعة المستخدمة بالويندوز**
  - ✓ **استدعاء البرامج الفرعية**

ومن الإشكال السابقة نجد أن المكدس لا يحتوي إلا على مؤشر واحد فقط TOP

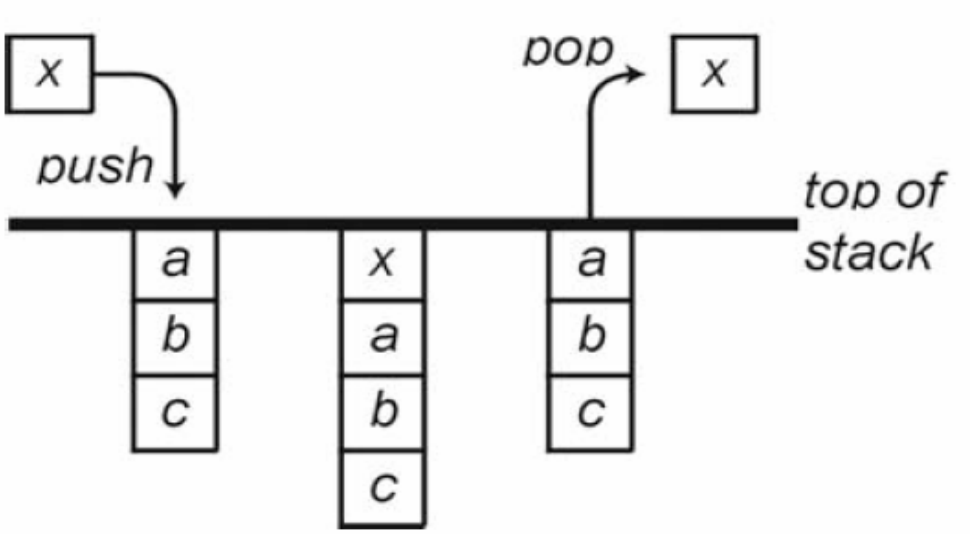
فعندما يكون المكدس فارغاً فأن TOP=-1 وعند إدخال أول قيمة فإننا نزيد من قيمة TOP ++ وكل ما أدخلنا قيمة فان المؤشر يزيد بمقدار واحد إلا أن يمتلئ المكدس .



وعملية أخراج القيم من المكدس فإننا ننقص المؤشر بمقدار واحد أيضا إلى أن يصل قيمة المؤشر = -1 أو NULLفبهاكدا يكون المكدس فارغاً.

- **طرق تمثيل المكدس وتخزين عناصره في الذاكرة وتامين عملية بلوغها:**
  - ✓ **التمثيل المترابط الحلقي لعناصر المكدس على شكل لائحة إي على شكل قائمة .**
  - ✓ **التمثيل المتراص (COMPACT) للعناصر في الذاكرة إي على شكل مصفوفة أحادية .**

وكمثال على تمثيل المكدس بالمتجهات لننظر إلى الشكل الأتي



ولنفترض أن لدينا مصفوفة حجم (4) وادخلنا أخر قيمة (X) وإذا أردنا إخراج قيمة فان المكدس سيعطي لنا أخر قيمة دخلت وهي (x) كما في الشكل السابق

وهذا أول برنامج له يعمل على إدخال قيم داخل المكدس ثم يقوم بطباعتها

1)

```
#include<iostream.h>
#include<conio.h>
int size=10;
int a[10],top=-1;
int pop();
void push(int[],int);
main(){int i,k;clrscr();
for(i=0;i<size;i++){cin>>k;push(a,k);}
for(i=0;i<size;i++)cout<<pop()<<" ";
getch();}
```

Top ويسمى ذيل المكدس وهو متغير عام تستطيع الدوال الوصول إلية وتغير قيمته

دالة أخراج البيانات من المكدس

دالة إدخال البيانات من المكدس وهي تأخذ بارا متر من نوع مصفوفة ومن نوع أنتجر

```
void push(int a[],int k){
if(top==sise-1)cout<<" FULL STACK";
else
a[++top]=k;
}
int pop(){
if(top<0)cout<<"EMPTE STACK";
else
return a[top--];
}
```

نعلم أن المكدس ممتلئ عندما يكون مؤشر الذيل أي top = حجم المصفوفة -1

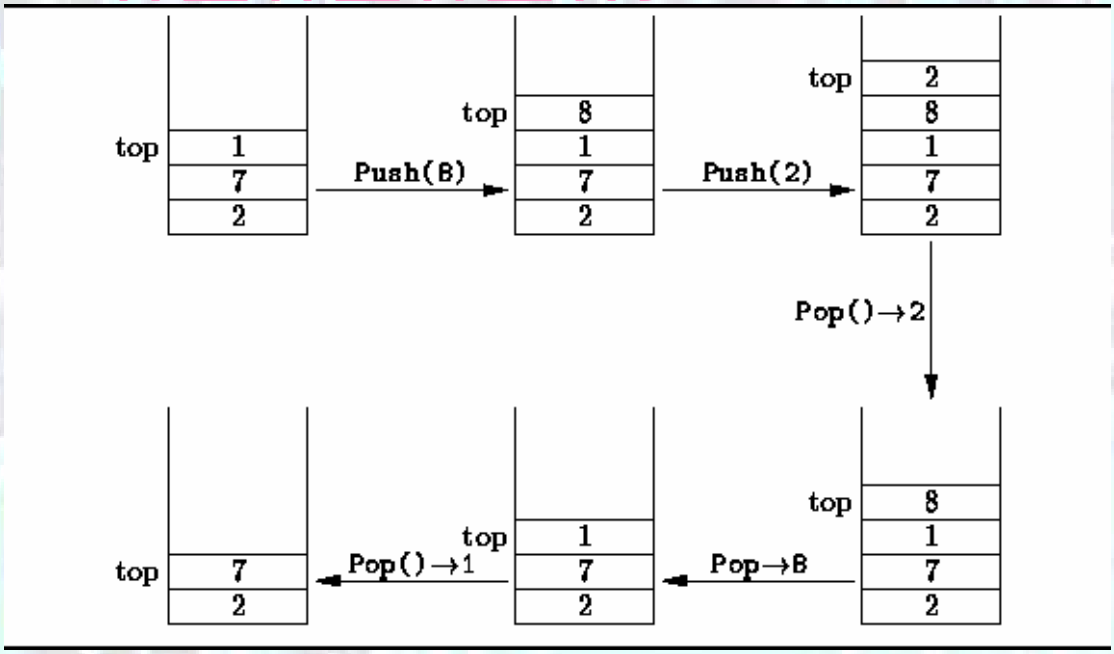
وإلا سنزيد من المؤشر بواحد وسنضع القيمة بداخل المصفوفة بداخل الموقع الذى تكون قيمته top

نعلم أن المكدس أصبح فارغاً عندما تكون قيمة المؤشر top =-1 أي اقل من الصفر

وإلا سنخرج القيمة من داخل المصفوفة التي تحمل عنوان قيمة top وسنطرح قيمة المؤشر بواحد .

STACK TOP ,PUSH,POP

كل هذه التعابير عبارة عن أسماء متغيرات وليس من الضروري التقيد بهذه الأسماء فهي ليست دوال .



ومن المثال السابق والإشكال التوضيحية يتبين لنا إليه عمل المكدس فيا أحبابي لا يخيفكم هذا المصطلح الغريب STACK فهو مثل المصفوفة ولا يختلف عنها إلى بشيء واحد ألا وهي إليه عملة التي ذكرناها سابقاً.

وكمثال أخر سنقوم باستخراج اكبر قيمة بالمكدس فقط سيكون التغير في دالة ألإخراج pop

```
2) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],top=-1;
int pop();
void push(int[],int);
main(){clrscr();int i,k;
for(i=0;i<size;i++){cin>>k;push(a,k);}
cout<<"THE MAX  VAL = "<<pop();
getch();}
void push(int a[],int k){
if(top==size-1)cout<<" FULL STACK";
else

a[++top]=k;
}
int pop(){int i,max=a[top--];
for (;;)
{if(top<0)break;
 else
if(max<a[top])max=a[top];top--;}
return max;
}
```

سيتوقف عمل اللوب عندما يكون top <0 وإلا سنقارن قيمة max بالقيم وهكذا إلى أن يصبح Max حاصل على اكبر قيمة

اعتقد أن المثال واضح وبسيط ولا يوجد بة إي تعقيد فقط الاختلاف هو إن في البرنامج السابق كان اللوب في الدالة الأساسية وفي هذا البرنامج عملنا اللوب بداخل دالة الجلب وعندما تنتهي القيم من المكدس يعاد للدالة اكبر قيمة حصلنا عليها ز

معنى for(;;) أنها حلقة لا نهائية فاجبر على عمل شرط التوقف BREAK .

ومن تطبيقات المكدس أيضا فحص الأقواس بتعابير Prefix and Postfix وسنورد مثالنا بدون استخدام المكدس .

```
3) #include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void check(char[]);
main(){clrscr();char s[100];
cin>>s;check(s);
getch();}
void check(char s[]){char c;int i,x,y;x=y=0;
for (i=0;(c=s[i])!='\0';i++){
if(c=='('||c=='[')x++;
 else
```

فكرة البرنامج
هو جعل متغيرين متغير لقوس الفتح) ومتغير لقوس الإغلاق (
وكل ما يمر على قوس يزيد من قيمة كل متغير المخصص للحالة
ويقارن في اللوب عن القيمتين فإذا كان قوس الإغلاق اكبر من قوس الفتح فأنة يطبع رسالة للمستخدم ويتوقف عمل البرنامج .....
وفي الأخير يقارن أيضا القيمتين فإذا كان إحدى المتغيرين اكبر من الأخر بعمل نفس الشيء وإلا يطبع قبول التعبير

```
if(c==')'||c==']')y++;
if(y>x){cout<<"ERROR\n";exit(1);}
}
if(y>x||x>y){cout<<"ERROR\n";exit(1);}
cout<<"ACCEPT \n";
}
```

وهذا البرنامج نفس السابق إلا أنة بواسطة المكدس وبواسطة الكيانات إي السجلات وسيتم شرح السجلات لاحقاً.

```
4) #include<iostream.h>
#include<conio.h>
#include<string.h>
#define size 10
struct stack
{int top;
char a[size];}s;

void push(struct stack*);
void pop(struct stack*);
int full(struct stack*);
int empty(struct stack*);

void main()
{clrscr();
s.top=-1;int i;
char x[size];
cout<<"enter your string"<<endl;
cin>>x;
for (i=0;x[i]!='\0';i++)
{  if (x[i]=='(')
   {
   if (!full(&s))
   push(&s);
   }

  else if(x[i]==')')
   {
   if (!empty(&s))
   pop(&s);
   else s.top--;
   }
}

if (empty(&s))
cout<<"true";
else
```

```
cout<<"false";
getch();}

void push(struct stack *s)
{s->top++;
s->a[s->top]='(';}

void pop(struct stack *s)
{s->a[s->top--]=')';}

int full(struct stack *s)
{if (s->top==size-1)
return 1;
else
return 0;}

int empty(struct stack *s)
{if (s->top==-1)
return 1;
else
return 0;}
```

إلا ألان اعتقد قد تبثث فكرة المكدس والية عملة في ذهنك , ألان إذا طلب منك إن تدخل بيانات إلى المكدس وتعكس المكدس فكيف ذلك سيكون فكر قليلاً وتذكر إلية عمل المكدس ولا تقول نقوم بطباعة المصفوفة من البداية فهذا ليس صحيحاً فقد خليت من عملة .....

ها هل أتت الفكرة بعقلك حاول ولا تستعجل.....

يبدو لي أن الفكرة لم تأتي إليك إذن صلي على نبيك وتتبع البرنامج التالي بهدوء

```
5) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],top=-1;
int pop();
void r(int[]);
void push(int[],int);
main(){clrscr();int i,k;
for(i=0;i<size;i++){cin>>k;push(a,k);}
r(a);
for(i=0;i<size;i++)cout<<pop()<<" ";
getch();}
void push(int a[],int k){
if(top==size-1)cout<<" FULL STACK";
else
a[++top]=k;
}
int pop(){return a[top--];}
```

```
void r(int a[]){int x[10],y[10],top2=-1,top3=-1;
while(top>=0)x[++top2]=a[top--];
while(top2>=0)y[++top3]=x[top2--];
while(top3>=0)a[++top]=y[top3--];}
```

في برنامجنا السابق كانت فكرته هو استخدام مكدسين آخرين لعملية نقل البيانات فعندما نقلنا البيانات من المكدس الأول إلى المكدس الثاني فان البيانات قد انعكست ولكننا استخدمنا مكدس ثالث لكي نعيد البيانات إلى المكدس الأصلي بالطريقة التي طلبت منا وبالية عمل المكدس .

واليكم هذا المثال لعملية طباعة اكبر قيمة بالمكدس بواسطة الاستدعاء الذاتي

```
6) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],top=-1;
int pop();
void push(int[],int);
main(){clrscr();int i,k;
for(i=0;i<size;i++){cin>>k;push(a,k);}
cout<<pop();
getch();}
void push(int a[],int k){
if(top==size-1)cout<<" FULL STACK";

else
a[++top]=k;
}
int pop(){static int max=a[top--];
if(top<0)return max;
else
{if(max<a[top])max=a[top];top--;pop();}
}
```

وضيفة هذا الأمر هو حفظ قيمة المتغير في الذاكرة لأنة من المعلوم أن عندما نخرج من الذاكرة فان المتغيرات تموت وتنتهى القيم

وهذا الكود يعمل على حذف إي قيمة من المكدس ؟

```
7) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],top=-1;
int pop();
void del_pop(int[],int);
void push(int[],int);
main(){clrscr();int i,k;
for(i=0;i<size;i++){cin>>k;push(a,k);}
cout<<"ENTER NUMBER DELETING\n";
cin>>i;del_pop(a,i);
getch();}
```

```
void push(int a[],int k){
if(top==size-1)cout<<" FULL STACK";
else
a[++top]=k;
}
void del_pop(int a[],int i){int top2=-1,m[10];
while(top>=0){if(i!=a[top])m[++top2]=a[top];top--;}
while(top2>=0)cout<<m[top2--]<<" ";}
```

.....

إلى هنا نكون قد انتهينا من موضوعنا الشيق ويفترض بعد اطلاع القارئ إلى هذا الشرح والاك واد الذي شرحناها سابقاً أن يحل إي تمرين قد ياتية والآن حان وقت التمارين وعلى القارئ التدرب عليها وحلها .

*) اكتب برنامج يقوم باستخراج الأعداد المتكررة من المكدس إي بحذف الأعداد المتكررة ؟
*) اكتب برنامج يقوم بعملية ترتيب مكدس ؟
*)اكتب برنامج يقوم بعملية إزاحة يمين وشمال للأعداد حسب الإزاحة المطلوبة من المستخدم؟
*) اكتب برنامج يقوم بدمج مكدسين ويعمل جميع العمليات المنطقية (تقاطع, اتحاد, طرح) ؟
*) اكتب برنامج يقوم بتحويل الأعداد الزوجية بمكدس والفردية بمكدس أخر ؟
*) اكتب برنامج يقوم بحذف الأعداد الأولية من المكدس ؟

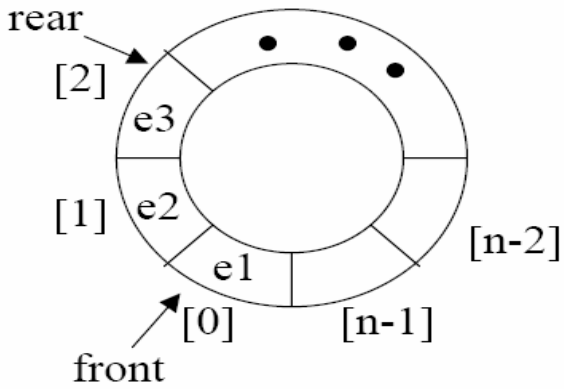
☒ ثانياً الطوابير (Queues) أو سجلات الانتظار

وهي عبارة نوع من هياكل البيانات الخطية ويشبه المكدس لتخزين المعلومات بشكل مؤقت مع فارق يكمن في أن التنظيم المتبع لإدخال المعلومات وإخراجها هو FIFO (First Input First Output) إي الداخل أولا الخارج أولا إي تكون عملية الإضافة من النهاية والحذف من الأمام إي يوجد للطابور مؤشرين مؤشر الرأس ويسمى head or front ومؤشر الذيل ويسمى tail or rear وعند الإضافة فإننا نزيد من قيمة الذيل بواحد وعند الحذف فإننا نزيد قيمة الرأس بواحد أيضا فتكون البيانات مرتصة بشكل نتتالى ومتقاربة على شكل خط وليست على مواقع متفرقة بالذاكرة إي أشبة بالطابور المدرسي فأول طالب حاضر هو أول طالب داخل للفصل .
فهو يشبه طابور الانتظار للإفراد عند المؤسسة أو المستشفى.

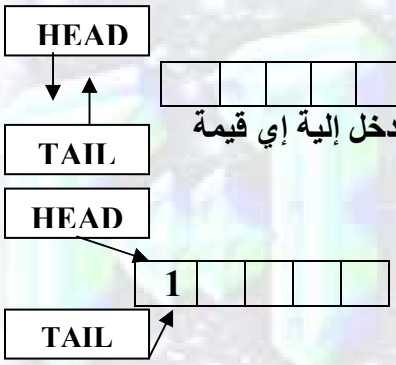
✓ أنواع الطوابير

1) طابور خطي / وهو له حجم محدود وشرط امتلائه أن تكون قيمة الذيل تساوي حجم المصفوفة .

2) طابور دائري / نفس تعريف السابق إلى أن شرط الامتلاء يختلف عن السابق الرأس =1 و الذيل =حجم المتجه أو الرأس=الذيل+1 .

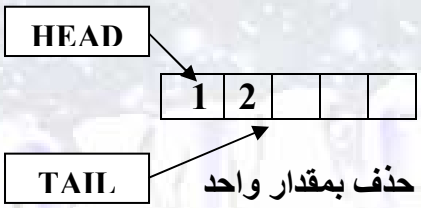


وسنبدأ بالتحدث إلى الطابور الخطي :-



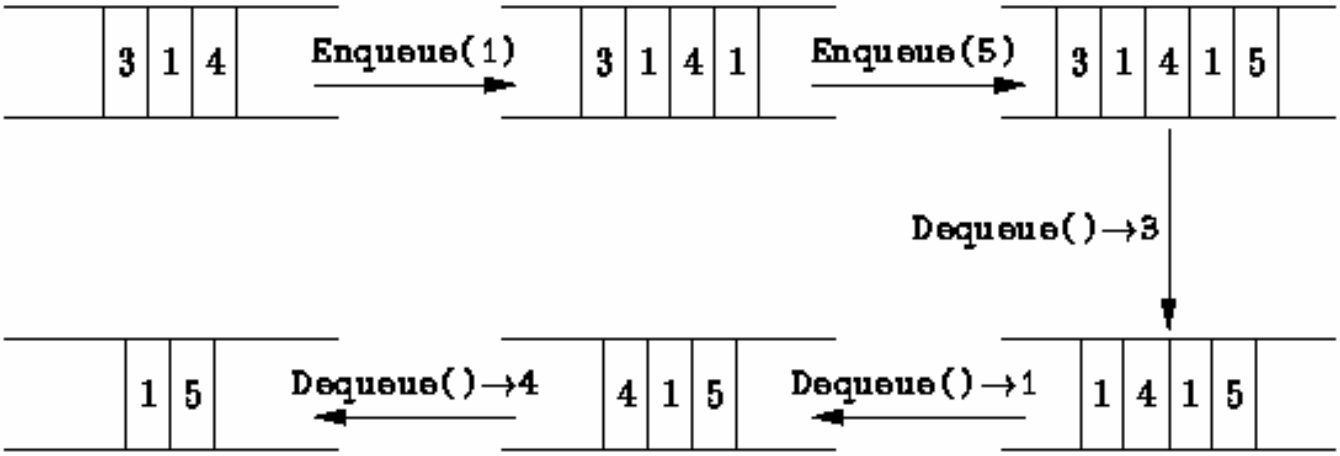
نجد في البداية يكون الرأس والذيل لا يؤشران لأي موقع ولمعرفة أن الطابور لم تدخل إلية إي قيمة عندما يكون if(tail==-1||head==-1)

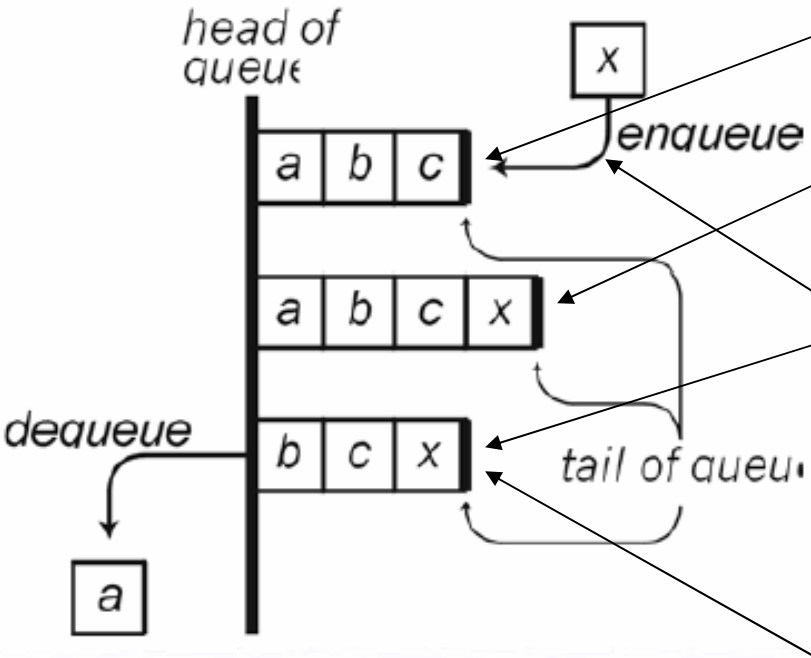
وعند إدخال أول قيمة يصبح قيمة الرأس والذيل = 0



وعند إدخال تأني قيمة نزيد من قيمة الذيل بواحد فقط أم الرأس يبقى كما هو

وعملية الحذف عكس السابق إي يكون الذيل تابت والرأس يزيد في كل عملية حذف بمقدار واحد مع عمل إزاحة للمتجه لليسار في كل عملية حذف إن أردت .





يبين الشكل أن الطابور يسمح بالإضافة
If(tail<size-1&&head==0)

يبين الشكل أن الطابور قد امتلئ ولا يسمح بالإضافة
If(tail==size-1&&head==0)

نلاحظ من الشكل الذي على اليسار أن عندما أضفنا x فان الذيل زاد بمقدار واحد وأصبح يؤشر علية .
وعندما أخرجنا قيمة من الطابور فان أول قيمة دخلت فإنها ستخرج الأولى وفي هذا الشكل قد خرجت . a وزدنا من قيمة الرأس بواحد وأصبح يؤشر على الذي بعدة .

يبين الشكل آن الطابور بة بيانات ويسمح بالحذف والإخراج
If(tail>0&&head<=tail)

وهذا اول برنامج بالطابور

```
8) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],tail=-1,head=-1;
```

```
int p_q();
void add_q(int[],int);
main(){clrscr();int i;
for(i=0;i<size;i++){if(tail==size-1){cout<<" FULL Queue";break;}
add_q(a,i+1);}
while(tail>=head)cout<<p_q()<<" ";
getch();}
void add_q(int a[],int k){
if(tail==-1){head=tail=0;a[tail]=k;}
else
a[++tail]=k;
}
int p_q(){return a[head++];}
```

وسنبدأ بسرد العمليات المطبقة على الطابور

*)الإضافة ADD

والكود التابع لهذه العملية هو نفس الكود السابق

*)الحذف DEL

وهذا الكود لهذه العملية

```
9) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],tail=-1,head=-1;
int p_q();
void add_q(int[],int);
void del(int[],int);
main(){clrscr();int i;
for(i=0;i<size;i++){if(tail==size-1){cout<<" FULL Queue";break;}
add_q(a,i+1);}
cout<<"ENTER VAL DELETE\n";cin>>i;
del(a,i);
while(tail>=head)cout<<p_q()<<" ";
getch();}
void add_q(int a[],int k){
if(tail==-1){head=tail=0;a[tail]=k;}
else
a[++tail]=k;
}
int p_q(){return a[head++];}

void del(int a[],int k){int x[10],t2,h2,m=head;t2=h2=-1;
if(tail==-1||head>tail)cout<<"THE QUEUE EMPTE\n";
else
while(m<=tail){
  if(a[m]!=k){
```

```
   if(t2==-1){h2=t2=0;x[t2]=a[m];}

else
  x[++t2]=a[m];
          }m++;}head=tail=-1;
while(t2>=h2){
if(tail==-1){head=tail=0;a[tail]=x[h2];}
else
a[++tail]=x[h2];
h2++;    }
                }
```

الفكرة المستخدمة بالبرنامج هو خلق طابور جديد وإدخال جميع القيم ماعدا القيمة التي تساوي القيمة المراد حذفها تم نقل الطابور الجديد للقديم

*) البحث

نفس البرنامج السابق إلى إننا لا نقوم بخلق طابور ونقل وو
بل إننا نبحث علية إن وجد نطبعه وإلا نطبع إننا لم نحصل علية .

```
10) #include<iostream.h>
#include<conio.h>
int size=10;
int a[10],tail=-1,head=-1;
int p_q();
void add_q(int[],int);
void ser(int[],int);
main(){clrscr();int i;
for(i=0;i<size;i++){if(tail==size-1){cout<<" FULL Queue";break;}
add_q(a,i+1);}
cout<<"\nENTER VAL SERCH\n";cin>>i;
ser(a,i);
while(tail>=head)cout<<p_q()<<" ";
getch();}
void add_q(int a[],int k){
if(tail==-1){head=tail=0;a[tail]=k;}
else
a[++tail]=k;
}
int p_q(){return a[head++];}

void ser(int a[],int k){int y=0,t=tail,h=head;
if(t==-1||h>t)cout<<"THE QUEUE EMPTE\n";
else
while(h<=t){
  if(a[h]==k){y=1;cout<<"FOUND\n"<<a[h];break;}h++;}
if(y==0)cout<<"NOT FOUND\n"<<k;cout<<endl;}
```

*) دمج طابورين

سأذكر فكرة البرنامج وعلى القارئ أن يحل الكود
أولا يجب التأكد من أن الطابور الثاني بوجد فيه مساحة كافية لاستيعاب قيم الطابور الأول

ومن العلاقة الآتية
(عدد العناصر الموجودة في الأساسي-عدد القيم)=<(عدد العناصر بداخلة - حجم الطابور المستضيف) ثم بعد ذلك نضيف الطابور الثاني في الأول .

*)ترتيب الطوابير

واليكم كود الترتيب باليه عمل الطابور وقد وجدت في أكثر الكتب أن عملية الترتيب يرتب الطابور مثل أليه ترتيب المصفوفة وهذا اعتقد انه خطاء.

```
11) #include<iostream.h>
#include<conio.h>
main(){clrscr();
int x[5],y[5],i,j,k,t=0,tail,taily,head,heady;
tail=taily=head=heady=-1;
cin>>j;tail=head=0;x[tail]=j;
for(i=1;i<5;i++){cin>>j;t=0;
while(head<=tail){heady=0;
if(t==0&&j>=x[head]){y[++taily]=j;t=1;}
y[++taily]=x[head++];}
head=tail=-1;if(t==0)y[++taily]=j;
while(heady<=taily){x[++tail]=y[heady++];head=0;}
heady=taily=-1;
}
for(k=0;k<=tail;k++)cout<<x[k]<<" ";cout<<endl;
getch();}
```

ثانيا الطابور الدائري:-

نفس البرامج التي ذكرناها سابقاً والاختلاف سيكون في أوامر الشرط
يكون الطابور فارغاً if(head==tail+1) .
يكون الطابور غير ممتلئ if(tail!=size&&head=1) .
يكون الطابور ممتلئا .if(head==1&&tail==size)
ونفس البرامج التي ذكرتها بالمكدس تطبق على الطوابير بالية الطابور.
وإلى هنا يجب على القارئ أن يكون قد اتضحت فكرة الطابور وان يكون قادراً على تنفيذ هذه التمارين.

تمارين على الطوابير

*) اكتب برنامج يقوم بعكس طابور ؟
*) اكتب برنامج يقوم بفصل طابور من مكان محدد من قبل المستخدم ؟
*) اكتب برنامج يقوم بحذف المواقع الزوجية بالطابور ؟
*) اكتب برنامج يقوم بتدوير طابور بالاتجاهين يمن وشمال ؟
*) اكتب برنامج يقوم بنسخ مكدس إلى طابور ؟// ملاحظة يوجد فرق بين النسخ والقص //
*) لديك طابور بة أشخاص وهم في المستشفى ومرقمين من 1 إلى 5 وهم في صف الانتظار فجأة الشخص رقم 3 أصيب بوجع قوي ولا يستحمل الانتظار فأمر الطبيب بإدخاله لأنة حالة طارئة فكيف تعالج هذه المشكلة بتطبيق إلية وخوارزمية الطابور الأتي أولاً للطبيب الداخل أولاً ؟

- **الهياكل الديناميكية**

وقبل أن ندخل في تفاصيل القوائم الأحادية والمذبلة والأشجار سنقوم بسرد مختصر عن كيفية التعامل مع السجلات, الكيانات والمؤشرات.

السجلات (Structures):

السجل عبارة عن مجموعة مترابطة من البيانات كما في المصفوفات ولكن السجل يحتوي بيانات مختلفة الأنواع وليست من نوع واحد كما في المصفوفة.

والسجل يتكون من عدة حقول (fields) تحوي البيانات المختلفة ويستخدم السجل لتخزين بيانات مترابطة متكررة، كما في قاعدة البيانات حيث تتكون قاعدة البيانات من سجلات بكل سجل منها نفس الحقول، ولكن قيم تلك الحقول تختلف من سجل لآخر.

كيفية تعريف السجل في لغة سي:

أولاً لابد ان نعلم ان كلمة **struct** كلمة محجوزه في لغة سي و سي++ ، و نستطيع تعريف السجل كالتالي:

```
struct (إسم السجل
{
أعضاء السجل
};
```

– طبعاً هذه الطريقة هي أحدا الطرق التي تستطيع تعريف السجل بها.

فلو اردنا ان نعرف سجل إسمه **data** و يحتوي على إسم من نوع **char*** و العمر من نوع **int**

إذا سيكون التعريف كالتالي:

```
struct data
{
   char namr[30];
   int age;
```

```
};
```

حيث (stucture_name) هو اسم السجل وبداخل السجل تتوالى الحقول المختلفة(الأنواع field1, field2،..... ولكل حقل نوعه الخاص.

وبتعريفنا للسجل يمكننا بعد ذلك تعريف متغيرات من نوع هذا السجل لاستخدامها في البرنامج حسب الحاجة

ويتم تعريف المتغيرات من السجل كما هو موضح بالشكل التالي الذي يوضح تعريف متغير (var1) من نوع السجل (structure1)

```
struct structure1
      {
            type   field1;
            type   field2;
            …
      }  var1;
```

ويمكننا تعريف أي عدد من المتغيرات من نوع هذا السجل كما يتطلب البرنامج.

والآن كيف نتعامل مع السجلات؟؟

إننا نحتاج مثلا لتخزين قيمة معينة في أحد الحقول، وفي هذه الحالة نستخدم المؤثر (.)

والمثال التالي يوضح عمل سجل باسم (Student) وتخصيص اسم (Mohammed) لحقل الاسم (name)

```
#include<iostream.h>
struct Student
      {
            char* name;
            int number;
      };
main()
{
      Student Sdt1;
      Std1.name=”Mohammed”;
```

```
    Cout << Std1.name;
}
```

وعند تنفيذ البرنامج تقوم العبارة الأخيرة بطباعة الاسم " **Mohammed**" وهو الذي قمنا بتخزينه في الحقل (**name**) من المتغير (**Std1**).

مصفوفة السجلات :

لقد علمنا ان السجل نوع كأي نوع من انواع البيانات، لذلك من الممكن ان يكون السجل مصفوفة ايضاً و الطريقة سهله جداً كالتالي :

```
structure_name var[NUM] ;
```

فلو اخذنا السجل :

```
typedef struct
{
    char name[30];
    int age;
}data;
```

و اردنا ان نعرف مصفوفة من نوع data يسكون كالتالي :

```
data student[100] ;
```

طبعاً العدد ١٠٠ إختياري .

و نحن في السابق أخذنا نوع student من السجل data و سيكون سجل واحد و لكن هنا سيتضح اهمية السجلات فعندما عرفنا student كمصفوفة من نوع data أصبح كأنه لدينا

١٠٠ طالب و كل عنصر في المصفوفة عباره عن سجل بحد ذاته.

و للوصول إلى محتويات السجل نتبع الطريقة التاليه :

student[indix].name & student[indix].age ...

و غالباً تستخدم مصفوفة السجلات إذا كان العدد محدداً أما إذا كان العدد غير محدد نستخدم طريقة من طريق الـ Data Structure منها اللنك لست درسنا القادم.

السجلات و المؤشرات :

و نعيد و نكرر انه بعد تعريف السجل يصبح نوع كأي نوع آخر من انواع البيانات، إذا يمكن للسجل ان يكون مؤشر (Pointer) و العمليه كالتالي:

```
typedef struct
{
	char name[30];
	int age;
}data;
```

و سنعرف مؤشر للسجل كالتاالي :

```
data *s ;
```

فالنأخذ البرنامج التالي للتوضيح :

```
#include<stdio.h>
#include <conio.h>
```

```
typedef struct
{
	char name[30];
	int age;
}data;

int main()
{
	data *s, std;

	s = &std;	// Assign std to s

	strcpy(std.name,"Talal");
	std.age = 20;

	printf("std.name = %s, std.age =
%d\n\n",std.name, std.age);
	printf("s->name = %s, s->age = %d\n\n",s-
>name, s->age);

return 0;
}
```

طبعاً نلاحظ الآن ظهور العلامة '->' بدل من النقطة عند إستخدام المتغير **s** ؟! لماذا ؟

الجواب : لأنه مؤشر لسجل و مؤشر السجل يستعمل في لغة السي و السي++ هذه العلامة بدلاً من العلامة '.' ، و هذا من الاختلافات التي تميز لغة السي و السي++ عن باقي اللغات مثل الجافا و الدلفي فهي لا تفرق إذا كان مؤشر أو لا .

إذا قاعدة في لغة سي و سي++ هي إنه عند إستخدام مؤشر لسجل نستخدم -> بدلاً من '.'

طبعاً هناك طريقة أخرى و هي هكذا:

s->name بدل (*s).name

طبعاً العلامة '->' أسهل :).

●السجلات و الدوال :

عند إستخدام السجلات مع الدوال إما أن يكون السجل مرسل للدالة أو إما ان يكون معاد من الدالة و إما ان يكون مستخدم في ضمن الدالة .

الحالة الأخيره معروفة و عملنا عليها في السابق داخل الدالة **main** و الـ **main** دالة اصلاً.

أما الحالتين الأولى و الثانيه فسنتطرق لها الآن.

- أولاً السجل معامل من معاملات الدالة :

أي أن نرسل السجل للدالة و الدالة تقوم بالعمليات على هذا السجل مثلاً: طباعة، معالجة، ... إلخ

و لنأخذ هذا المثال و نشرحة بعد قرائة المثال جيداً:

```
//-----------------------------------------------------

#include<stdio.h>
#include <conio.h>
//-----------------------------------------------------
typedef struct
{
        char name[30];
        int age;
```

```
}data;
//-------------------------------------------------------
void display(data r);
//-------------------------------------------------------
main()
{
	data std;

	strcpy(std.name,"Talal");
	std.age = 20;

	display(std);
}

//-------------------------------------------------------
void display(data r)
{
	printf("(r.name) = %s,\n(r.age) =
%d\n\n",r.name, r.age);
}

//-------------------------------------------------------
```

و في هذا المثال لقد كتبنا رأس الدالة كالتالي:

```
void display(data r) ;
```

أي أنه يوجد دالة إسمها display تستقبل السجل r من نوع data ولا تقوم بإرجاع شئ.

و عند إستدعاينا الدالة و بعد إعطائها القيم كالتالى:

```
display( std ) ;
```

ارسلنا لها السجل كاملاً لتستقبله و تطبعه في جسم الدالة display .

و لنأخذ مثالاً آخر لإعطاء قيم السجل في الدالة و طبعاتها في الـ main :

```
#include<stdio.h>
#include <conio.h>

typedef struct
{
	char name[30];
	int age;
}data;

void assign(data *r);

main()
{
	data std;

	assign(&std);
	printf("std.name = %s,\nstd.age =
%d\n\n",std.name, std.age);
}

void assign(data *r)
{
	strcpy(r->name,"Talal");
```

```
        r->age = 20;
}
```

و في هذا المثال كتبنا رأس الدالة (التعريف) هكذا :

```
void assign(data *r) ;
```

و جعلنا r كمؤشر لأن قيمة r ستتغير (نحن نريد ذلك) لإعطائها القيم.

و قمنا بإرسال السجل كالتالي :

```
assign( &std ) ;
```

لأن الدالة assign تستقبل مؤشر للسجل لذلك نرسل لها عنوان السجل و ليس السجل نفسه.

و داخل الدالة assign إستخدمنا r->name و r->age لأن r في الدالة مؤشر (و مع المؤشرات نستخدم -> بدلاً من '.') .

<u>- ثانياً إرجاع سجل من الدالة :</u>

أي أن الدالة تقوم بإرجاع السجل عند الانتهاء من عملها و نستطيع تغيير البرنامج السابق ليرجع السجل بدلاً من إرسال السجل كعنوان و إستقباله كمؤشر.

سيتغير البرنامج ليصبح هكذا :

```
include<stdio.h> #
#include <conio.h>

typedef struct
```

```
{
        char name[30];
        int age;
}data;

data assign(void);

main()
{
        data std;

        std = assign();
        printf("std.name = %s,\nstd.age =
%d\n\n",std.name, std.age);
}

data assign(void)
{
        data r;
        strcpy(r.name,"Talal");
        r.age = 20;
    return r;
}
```

و هنا عرفنا الدالة كالتالي :

```
data assign(void) ;
```

أي أن الدالة assign لا تستقبل شئ و القيمة المرجعة من الدالة هي عباره عن سجل من نوع data .

و قمنا بإستدعا الدالة هكذا :

```
std = assign() ;
```

أي أن القيمة المرجعة من الدالة ستوضع قيمتها في السجل std .

و في جسم الدالة assign عرفنا المتغير r من نوع سجل data و أعطينا لها قيم و قمنا بإرجاع هذا السجل من الدالة عن طريق الامر

```
return r ;
```

- إسناد السجلات :

نستطيع ان نسند سجلين لبعضهما البعض لكن شريطة أن يكونا من نفس النوع .

فلو أنشئنا السجل التالي :

```
typedef struct
{
char name[30];
int age;
}data;
```

و عرفنا منه متغيرين هكذا :

```
data a, b ;
```

و أعطينا المتغير a هذه القيم :

```
strcpy( a.name, "talal" ) ;
a.age = 20 ;
```

فبإمكاني ان اسند للمتغير b نفس محتويات المتغير a عن طريق هذه الجملة :

```
b = a ;
```

- إعطاء السجل أكثر من إسم أو إعطائه المتغيرات لحظة بناء السجل :

فلو كان لدينا السجل التالي :

```
typedef struct
{
char name[30];
int age;
}data, MyData ;
```

أستطيع أن اعرف المتغيرات سواء كان بـ data أو بـ MyData و كلها صحيحه.

فلو قلت :

```
MyData student ;
```

أو

```
data student ;
```

كانا سواء .

و هذا هو إعطاء السجل اكثر من إسم ، أما إعطاء السجل أكثر من متغير لحظة بناء السجل و بدون تحديد إسم للسجل يكون كالتالي :

```
struct
{
الاعضاء
}إسم المتغير ;
```

فلو اردنا ان نعمل على ١٠٠ طالب فقط و متأكيدن أن العدد لن يزيد عن ١٠٠ طالب فالأفضل بناء السجل هكذا :

```
struct
{
    char name[30] ;
    int age ;
} student;
```

و هكذا يصبح student متغير و نقول :

```
student. name & student. age
```

طبعاً إلى الآن تعلمنا كيف ننشئ السجل بثلاثة طرق بقي الطريقة الرابعة و الاخيره و هي كالتالي :

```
struct (إسم السجل
{
الاعضاء
(المتغيرات) ؛}
```

أي نستطيع أن ننشئ سجل الطالب الذي تكرر علينا كثيراً بالطريقة الرابعه هكذا :

```
struct data
{
    char name[30] ;
    int age ;
} student;
```

هنا student سيكون متغير و data هو إسم السجل و هنا نستطيع في كل مرة نحتاج فيها لإنشاء سجل أن نشئ سجل بالطريقة :

```
struct data VAR ;
```

و إستعمال student كمتغير جاهر غير محتاج للتعيرف .

** نقطة أخيره :

في كل جزئ من أجزاء البرامج التي كتبتها و التعريفات و إنشاء المتغيرات في الدرس إستخدمت غالباً التعريف التالي :

```
typedef struct
{
```

```
    char name[30];
    int age;
}data;
```

و أنشئت المتغيرات كالتالي :

```
data VAR ;
```

ممكن تغييره إلى

```
struct data
{
char name[30] ;
int age ;
};
```

و لكن تعريف المتغير سيكون :

```
struct data VAR ;
```

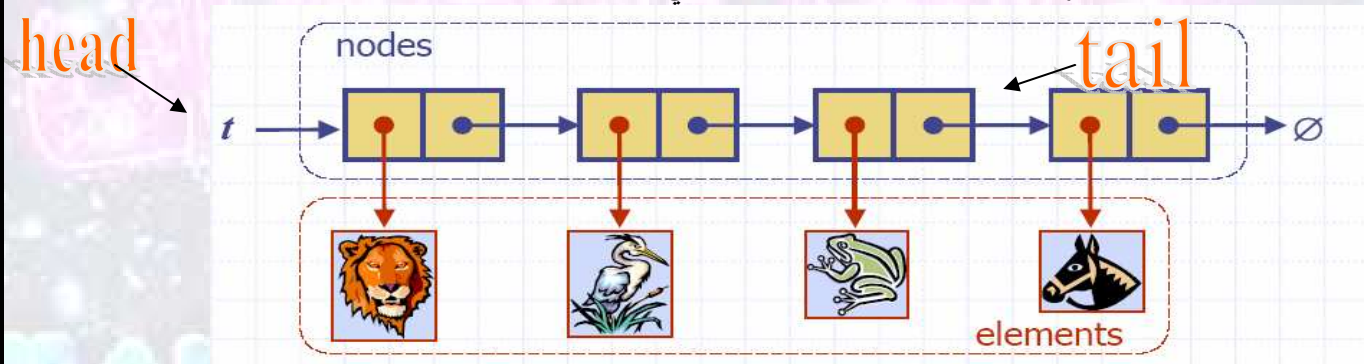
• ثالثا القوائم (List)

وهي نوع من هياكل البيانات الخطية تتألف من مجموعة من الخلايا المرابطة فيما بينها وكل عنصر فيها يسمى عقدة وهذه العقدة فيها حقلين حقل للقيم وحقل يؤشر لعنوان للعقدة الذي بعدها أو قبلها أو NULL وتستعمل هذه الكلمة للدلالة إلى نهاية اللائحة , ومن الممكن أن تتألف العقدة على أكثر من مؤشر ومعلومات إي قيم , فتكون ضمن مجموعة (block) أو كتلة , ولا بد من مؤشر يؤشر إلى أول عقدة ومؤشر يؤشر إلى أخر عقدة إي مثل الطابور .

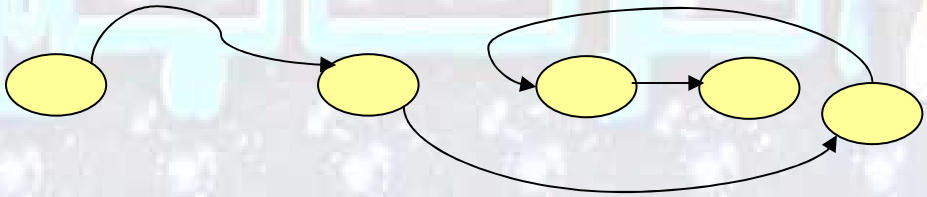
➢ أنواع القوائم

- ✓ القوائم الأحادية.
- ✓ القوائم المذبلة أي الثنائية.
- ✓ القوائم الدائرية.

وسنبدأ بالتحدث عن القوائم الأحادية / و تشبه حبل الغسيل تعلق علية البيانات تتالياً إذا كان الإدخال في نفس الوقت ويوجد عنوان راسي يؤشر إلى أول عنصر من اللائحة ويسمى head ويوجد عنوان نهائي يؤشر إلى أخر عنصر من اللائحة ويسمى tail وكل عقدة تؤشر إلى العقدة التالية وأخر عقدة تكون قيمة المؤشر لها NULL و تكون بالشكل الأتي :-

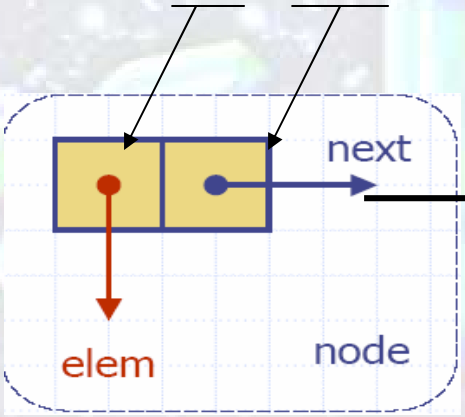


وليس من الضروري أن تكون العقد مرتبة بشكل متتالي في الذاكرة فهي تكون مبعثرة في الذاكرة لان الجهاز الذي يحجزها في الذاكرة وليس اليوزر لكنها متصلة فيما بينها بواسطة المؤشرات وهذا الشكل يبين كيف تكون شكلها .



*) كيفية تعريف الهيكل العام للقوائم الأحادية

```
Struct list{ int elem;list*next;};
```



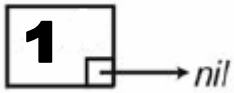
ما معنى هذا الحقل ؟
معناه مؤشر من نوع السجل نفسه أي يؤشر إلى سجل أخر من نفس نوع السجل أي نستطيع الوصول إلى سجل أخر وأخر وووووو إلى مالا نهاية ويكون بداخل العقدة موقع العقدة التي بعدها أو قبلها.
أي لنتذكر أحبائي علبة الحليب حيث أن بداخلها نفس صورة العلبة نفسها و بداخل الصورة نفس الصورة العلبة ووو إلا مالا نهاية والسجل هنا نفس الشيء حيث أن داخلة عقدة وبداخل العقدة عقدة وهكذا .

وللإضافة عدة أنواع
الإضافة من اليمين
الإضافة من اليسار
الإضافة من أي مكان
وسنرى أول مثال لهذه القوائم وهو الإضافة من اليمين للقائمة

```
12)#include<iostream.h>
#include<conio.h>
struct list{int d;list*next;};
void main(void){clrscr();
list *head,*node,*tile;
int i=0;
node=new list;node->next=NULL;
cin>>node->d;tail=head=node;
while(i<4){
node=new list;
cin>>node->d;
node->next=NULL;
tail->next=node;
tail=node;
i++;}
node=head;i=1;
while(node!=NULL){cout<<endl<<i++<<" "<<node->d;node=node->next;}
getch();
}
```

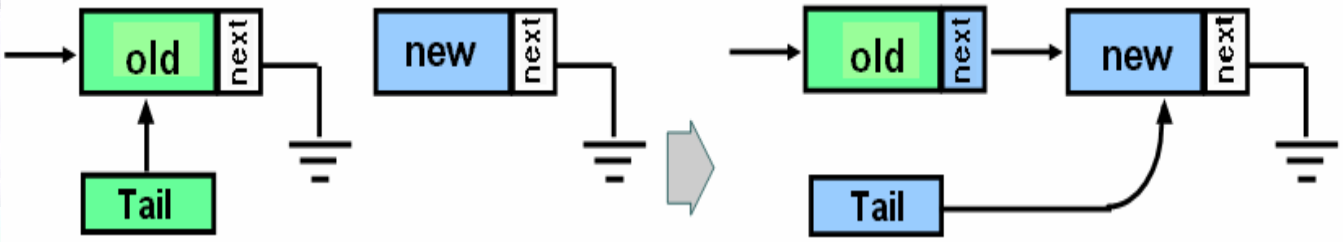
هنا عرفنا head من نوع list مؤشر لسجل وهو الرأس
و tail من نوع list مؤشر لسجل وهو الذيل
و node سنستخدمه كمتغير لإدخال بيانات العقد .

هنا أول خطوة وهي ضرورية ولابد أن تكون منفردة عن أخواتها لكي نساوي الرأس والذيل بأول عقدة .
الأمر node=new list معناه اصنع لي عقدة جديدة أي سجل بالذاكرة
Node->next=NULL أي جعل حقل العقدة لا يؤشر على عقدة أي خالي
ثم ساوينا الذيل والرأس بالعقدة الجديدة حقنا أي بهذا الشكل .



هنا سنكون 4 عقد أضافية بجانب الأولى فيكون لدينا 5 عقد .
في اللوب السطر الأول والثاني قد سبق شرحه أم الثالث
Tail->next=node أي حقل العقدة الأولى يساوي العقدة الجديدة
فبهذه الحالة تمت عملية الربط بين العقدتين بقي علينا نقل الذيل إلى العقدة الجديدة tail=node وهكذا بباقي العقد إلى أن ينتهي عمل اللوب ويمكنك تكوين مئات العقد بهذه الطريقة والشكل في الأسفل بين الشرح .

في عملي طباعة العقد أول شي يجب أن تعمله هو الوصول لأول عقدة فكيف ستعمل لو تتذكر قليل أن أول ما أنشانا أول عقدة ساوينا الرأس والذيل بها وبعد ذلك كان كل ما أضفنا عقدة جديدة تحرك معانا الذيل وأصبح الذيل بمؤخرة العقد والرأس في بداية العقد
إذن نساوي النود بالرأس فنكون وصلنا إلى أول عقدة وما بقي علينا إلى طباعتها والتنقل إلى العقدة التالية بهذه العملية node=node->next
ومعناها أن العقدة الذي واقفين عليها تساوي حقل العقدة نفسها الذي داخلة موقع العقدة التالية فبذلك نكون قد انتقلنا إلى العقدة التالية وتستمر هذه العملية إلى أن تساوي العقدة NULL فينتهي عمل اللوب .



أم الإضافة من اليسار نفس السابق الا أن الاختلاف فقط بعملية إدخال العقد الثانية وما بعدها

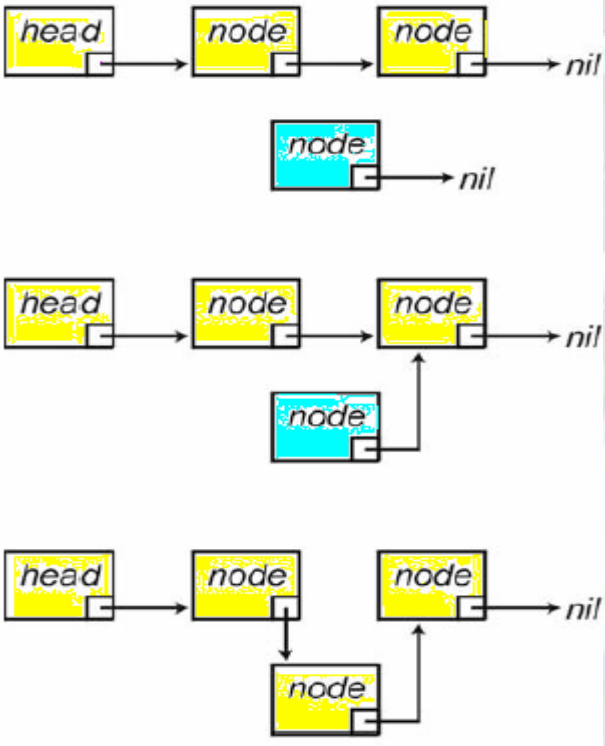
```
while(i<4){
node=new list;
cin>>node->d;
```

ملاحظة
الإضافة من اليمين يكون الرأس متحرك
والإضافة من اليمن يكون الذيل هو المتحرك

```
node->next=head;
head=node;
i++;}
```

وهذا برنامج إضافة عقدة بعد عنصر يريده المستخدم

```
#include<iostream.h>
#include<conio.h>
struct list{int d;list*next;};
void main(void){clrscr();
list *head,*node,*tile,*k;
int i=0,x,y;
node=new list;cin>>node->d;node->next=NULL;
tile=head=node;
while(i<3){
node=new list;
cin>>node->d;
node->next=NULL;
tile->next=node;
tile=node;i++;
}
node=head;i=1;
while(node!=NULL){cout<<endl<<i<<" "<<node->d;i++;node=node->next;}
cout<<"\n ENTER VAL\n";cin>>x;
node=head;
while(node!=NULL)
{if(node->d==x){k=new list;cin>>k->d;k->next=node->next;node->next=k;break;}
node=node->next;}
node=head;i=1;
while(node!=NULL){cout<<endl<<i<<" "<<node->d;i++;node=node->next;}
getch();
```

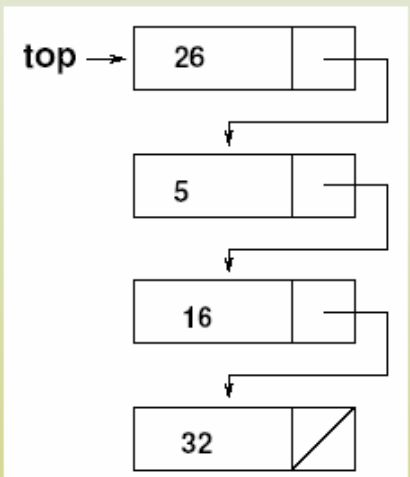


بعد إدخال العقد طلبنا من المستخدم إدخال قيمة فإذا وجدت هذه القيمة بالقائمة سنضع العقدة الجديدة بعدها مباشرة

فعملنا عملية بحث عن العنصر إذا وجد فإننا سنعمل على إنشاء عقدة جديدة وسندخل قيمة العقدة وسنربط حقل مؤشر العقدة الجديدة بالعقدة التي بعد العنصر والعقدة التي مازلنا واقفين عليها تم ربط مؤشرها بالعقدة الجديدة والشكل الذي في الأسفل يبين هذه العملية . .

في الفصل الأول تحدثنا عن الهياكل الاستاتيكيه أي الثابتة وتكلمنا عن المكدسات والطوابير لنعمل على تطبيق تلك الخوارزميات بالقوائم الأحادية ونجعلها متغيرة أي ديناميكية ونتخلص من شيء أسمة المكدس قد امتلئ أو الطابور قد امتلئ وألان سنورد مثال عن المكدس باستخدام القوائم الأحادية والية الإدخال والإخراج قد تكلمنا عنها في الفصل السابق .

واليكم الكود




```
#include<iostream.h>
#include<conio.h>
#include<math.h>
struct stck {int data;stck*last;};
push(stck*&,int);
int pop(stck*&);
void main(){clrscr();int i;stck*top=NULL;
for(i=1;i<6;i++)push(top,i);

while(top!=NULL)cout<<pop(top)<<endl;
getch();}

push(stck*&top,int i){stck*k;
if(top==NULL){k=new stck;k->data=i;k->last=NULL;top=k;}
else
{k=new stck;k->data=i;k->last=top;top=k;}
}
int pop(stck*&top){stck*k=top;top=top->last;return k->data;}
```

وهذا الكود يعمل على ترتيب مكدس بواسطة القائمة

```
#include<iostream.h>
#include<conio.h>
struct list {int a;list*last;};
push(list**,int);
pup(list*);
shlist(list*);
void main(){clrscr();list*top,*node;node=top=NULL;
top=new list;top->a=1;top->last=NULL;
for(int i=1;i<6;i++)push(&top,i+1);
shlist(top);cout<<endl;
pup(top);
shlist(top);cout<<endl;
getch();}

push(list**top,int i){list*k;k=new list;k->a=i;k->last=*top;*top=k;}
shlist(list*p){while(p!=NULL){cout<<p->a<<" "; p=p->last;}}
pup(list*p)
{list *n=p;struct s{int a;s*next;};
s*k,*h,*f,*i,*j;k=new s;k->a=n->a;k->next=NULL;h=f=k;n=n->last;
while(n!=NULL){k=new s;k->a=n->a;k->next=NULL;h->next=k;h=k;n=n->last;}
```

```
for(i=f;i->next!=0;i=i->next)
for(j=i->next;j!=0;j=j->next)
if(i->a>j->a){int k=i->a;i->a=j->a;j->a=k;}

k=f;n=p;
while(k!=NULL){n->a=k->a; k=k->next;n=n->last;}
```

وهذا الكود عبارة عن مكدس بالقائمة فيعمل على جعل الأعداد الفردية بمكدس أخر والزوجية بمكدس أخر

```
#include<iostream.h>
#include<conio.h>
struct list
{int a;list*last;};
push(list**,int);
pup(list*,list*&,list*&);
shlist(list*);
void main(){clrscr();list*top,*top2,*top3;top2=top3=top=NULL;
top=new list;top->a=1;top->last=NULL;
for(int i=1;i<10;i++)push(&top,i+1);
shlist(top);
pup(top,top2,top3);
shlist(top3);
shlist(top2);
getch();}

push(list**top,int i){list*k;k=new list;k->a=i;k->last=*top;*top=k;}
shlist(list*p){while(p!=NULL){cout<<p->a<<" "; p=p->last;} cout<<endl;}
pup(list*top,list*&top2,list*&top3)
{list*t,*r,*p=top;t=r=NULL;t=r=new list;
while(p!=NULL)
{if(p->a%2==0){if(t==NULL){t->a=p->a;t->last=NULL;top2=t;}
            else
            {t=new list;t->a=p->a;t->last=top2;top2=t;}}
    else
            {if(r==NULL){r->a=p->a;r->last=NULL;top3=r;}
            else
            {r=new list;r->a=p->a;r->last=top3;top3=r;}}
 p=p->last;}}
```

وهذا كود بواسطة دالة تستدعي مكدس فتعمل على وضع الإعداد الفردية من جهة والزوجية من جهة أخرى

```
#include<iostream.h>
#include<conio.h>
```

```
struct list{int d;list*last;};
void c(list*&);
void main(){clrscr();list*top,*node;int a[]={2,3,6,1,9,8,5,7,6,1,3,2,55,6,1,7,8,6};
node=new list;node->d=a[0];node->last=NULL;top=node;
for(int i=1;i<13;i++){node=new list;node->d=a[i+1];node->last=top;top=node;}
node=top;
while(node!=NULL){cout<<node->d<<" ";node=node->last;}
c(top);cout<<endl;
node=top;
while(node!=NULL){cout<<node->d<<" ";node=node->last;}
getch();}

void c(list*&node){list *k,*m,*top2=NULL,*top3=NULL,*n;
while(node){
if(node->d%2==0){
if(top2==NULL){m=new list;m->d=node->d;m->last=NULL;top2=m;}else
{m=new list;m->d=node->d;m->last=top2;top2=m;}
        }else
if(top3==NULL){n=new list;n->d=node->d;n->last=NULL;top3=n;}else
{n=new list;n->d=node->d;n->last=top3;top3=n;}
k=node;node=node->last;
delete k;}
while(top3){m=new list;m->d=top3->d;m->last=top2;top2=m;k=top3;
top3=top3->last;delete k;}
node=top2;
}
```

وهذا الكود يعمل على عكس قائمة

```
#include<iostream.h>
#include<conio.h>
struct list{int d;list*next;};
void main(void){clrscr();
list *tail,*head,*node,*k,*p;
int i=0;
node=new list;cin>>node->d;node->next=NULL;tail=head=node;
while(i<3){
node=new list;
cin>>node->d;
node->next=NULL;
tail->next=node;i++;
tail=node;}
node=head;k=new list;k->d=head->d;k->next=NULL;p=k;node=node->next;
while(node!=NULL){
k=new list;k->d=node->d;
k->next=p;p=k;
node=node->next;}
```

```
node=p;
while(node!=NULL){cout<<node->d<<" ";node=node->next;}
getch();
}
```

وجميع ما ذكرناه من تطبيق المكدس بالقائمة الأحادية على القارئ أن يطبق تلك الأمثلة بالطوابير.

وبعد أن تكلمت على عملية الإضافة بجميع أنواعها بالقائمة الأحادية يبقى لنا ألان أن نتكلم عن عملية الحذف وما يدور من تطبيقات حولها.

*)الحذف (DELETE) /هو عملية بسيطة في لائحة الوصل الخطية , فتوصل وصلة العنصر الذي يأتي قبل العنصر المراد حذفه بعنوان العنصر الذي يراد حذفه فتعتبر عملية عكسية لعملية الإضافة .

- **الحماية من ألأخطاء البرمجية المحتملة**

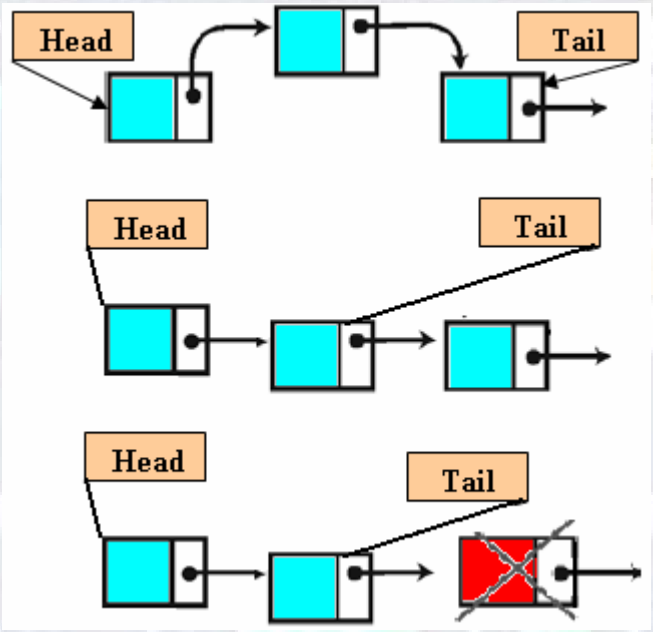
1. عند حذف عقدة يجب أولا الانتقال منها ثم تحذفها بهذا الآمر (delete) فعندما تريد حذف مجلد في الويندوز وهو مفتوح لا يحذف وعندما تريد هدم مبنى فينبغي عليك الخروج من المبنى ثم تهدم المبنى .
2. إن التجاهل في هذه الملاحظة فانه ينتج خطا قاتل يسبب في تعليق الجهاز أو إغلاق مترجم اللغة.

وللحذف عدة أنواع

- الحذف من الرأس
- الحذف من الذيل
- الحذف من أي مكان

وسنرى أول مثال لهذه القوائم وهو الحذف من النهاية للقائمة أي أخر عقدة.

ويتم ذلك جعل العقدة قبل الأخيرة في القائمة مساوية NULL تم ننقل الذيل إلى وراء بمقدار واحد أي العقدة التي قبل الأخير ثم نحذف العقدة الأخيرة .

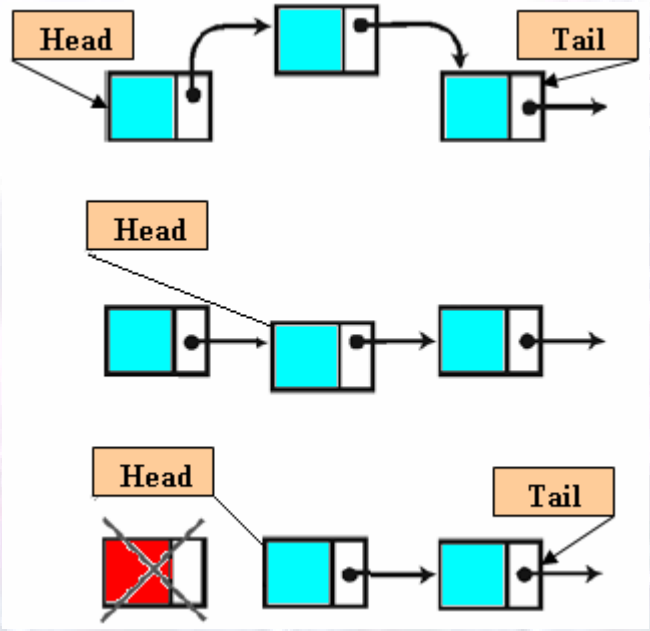



```
#include<iostream.h>
#include<conio.h>
struct list{int d;list*next;};
void main(void){clrscr() ;
list *head,*node,*tile,*p;
int i=0;
node=new list;cin>>node->d;
tile=head=node;
while(i<5){
node=new list;
cin>>node->d;
node->next=NULL;
tile->next=node;
tile=node;i++;
}
p=node=head;
while(node){if(node->next==NULL){p->next=NULL;delete node;break{;
p=node;node=node->next;}

node=head;
while(node!=NULL){cout<<node->d<<" ";node=node->next;}
```

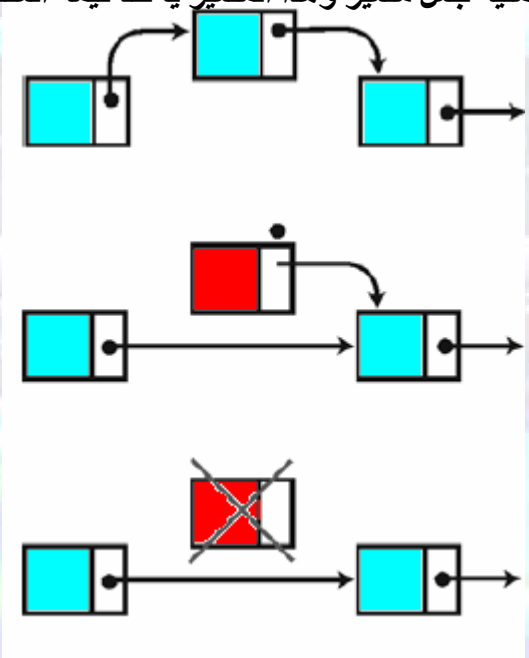
```
getch();
}
```

واليكم هذا الكود لعملية الحذف من الإمام أي أول عقدة .
سننقل الرأس إلى الإمام بمقدار واحد أي للعقدة التي بعدها ثم نحذف أول عقدة .

```
#include<iostream.h>
#include<conio.h>
struct list{int d;list*next;};
void main(void){clrscr();
list *head,*node,*tile;
int i=0;
node=new list;cin>>node->d;
tile=head=node;
while(i<5){
node=new list;
cin>>node->d;
node->next=NULL;
tile->next=node;
tile=node;i++;
}
node=head;
head=head->next;
delete node;
node=head;
while(node!=NULL){cout<<node->d<<" ";node=node->next;}
getch();
}
```

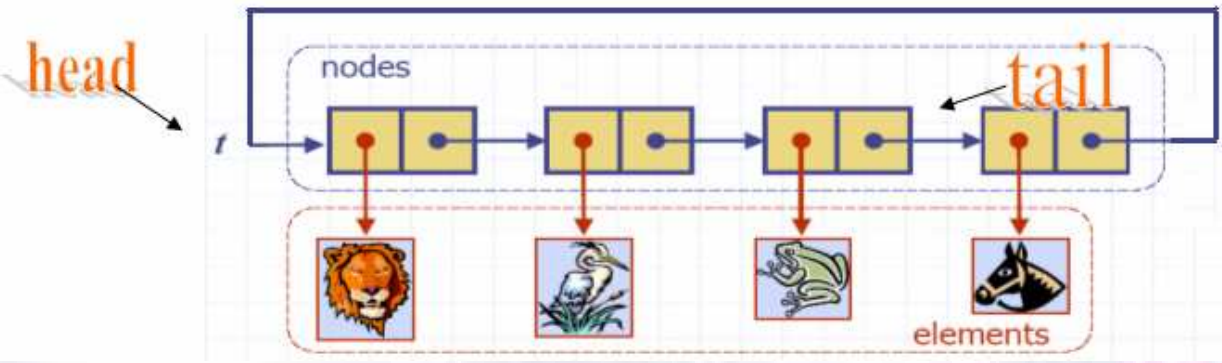


وعمليه الحذف من الوسط يتم البحث عن العقدة المراد حذفها تم نغير حقل المؤشر للعقدة الذي قبلها بالعقدة التي بعدها ويتم ذلك بجعل مؤشر يمشي ورآنا بمقدار واحد بعملية جعل متغير وهذا المتغير يا خد قيمه العقدة تم ننتقل للعقدة التي بعدها .



▪ القوائم الأحادية المتصلة.

حيث يشير مؤشر العقدة الأخيرة إلى العقدة الأولى أي مؤشر الذيل سيؤشر إلى الرأس .
وتستعمل هده القوائم المتصلة كثيراً في أنظمة إدارة بنوك المعطيات وفي البرمجة إذ تسمح بربط العناصر التي تتمتع بنفس الخصائص فيما بينها .




```
#include<iostream.h>
#include<conio.h>
struct list{int d;list*next;};
void main(void){clrscr();
list *head,*node,*tile;
int i=0;
node=new list;
cin>>node->d;node->next=NULL;tile=head=node;
while(i<5){
node=new list;
cin>>node->d;
node->next=head;
tile->next=node;
tile=node;i++;
}
node=head;
do
{cout<<node->d<<" ";
node=node->next;}while(node!=head);
getch();
}
```

وعندما تريد تحويل القائمة الدائرية إلى قائمة الخطية نجعل مؤشر أي عقدة في القائمة مساوياً إلى (NULL) فتتحول إلى قائمة متصلة .

تمارين

*)اكتب برنامج يقوم بترتيب قائمة أحادية ؟

*)اكتب برنامج يقوم بقلب أي عكس قائمة أحادية ؟

*)اكتب برنامج يقوم بقلب المواقع الزوجية مع الفردية أي swap ؟

*)اكتب برنامج يقوم بحذف المواقع الزوجية بالقائمة الأحادية ؟

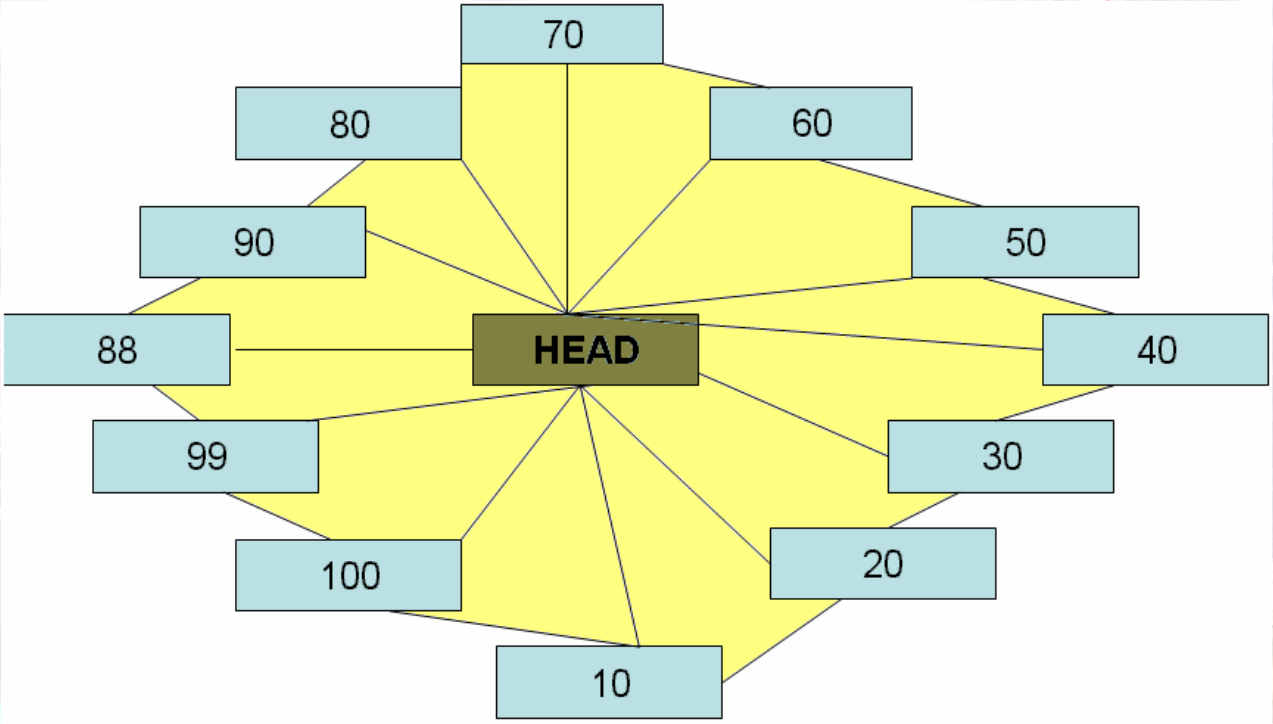
*)اكتب برنامج يقوم بدمج قائمتين أحاديتين وعمل جميع العلاقات الرياضية (تقاطع , اتحاد , طرح) ؟

*)اكتب برنامج يقوم بطباعة ثالث اكبر قيمة من القائمة ؟

*)اكتب برنامج يقوم بترتيب طابور بواسطة القائمة الأحادية ؟

*)اكتب برنامج بواسطة الدوال يقوم بطباعة ثالث مجموع اكبر القيم وطباعة الأعداد بالقائمة الأحادية مثل (1,2,7,9,5,6) فيطبع 21 والإعداد 7,9,5 ؟

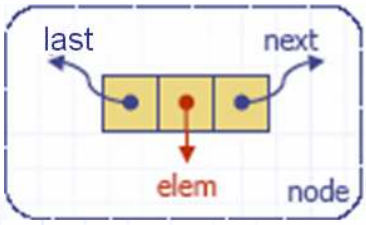
*)اكتب برنامج يقوم بتمثيل بيانات القائمة ألأحادية في الذاكرة بهذا الشكل ؟

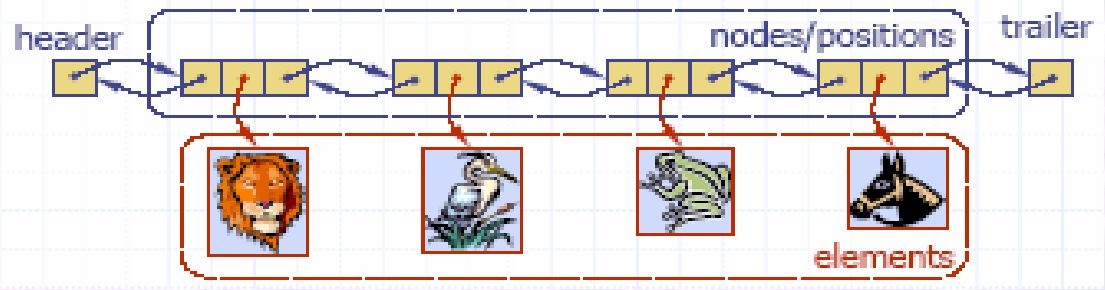


▪ القوائم المذبلة الثنائية.

تعتبر القوائم الثنائية قوائم أحاديه ولكن ليس العكس حيث أن القوائم المذبلة لها مؤشرين مؤشر يؤشر إلى العقدة التالية ويسمى next ومؤشر يشير إلى العقدة السابقة يسمى last .
وتستعمل هذه القوائم عندما نحتاج للرجوع إلى وراء لجلب معلومات معينة ولنتذكر برنامج معالجة النصوص حيث أنة يستطيع العودة إلى الورى لتعديل حرف مثلاً.
ويكون الهيكل العام لها

```
Struct list{int elem,list *next,*last;}  ;
```





ومن الشكل السابق يتضح لنا شكل هذه القوائم .
ونفس العمليات التي طبقت على القوائم الأحادية ستطبق على القوائم المذبلة.
ونبدأ بأول عمليه ألا وهي عملية الإضافة.
وهذا الكود لهذه العملية

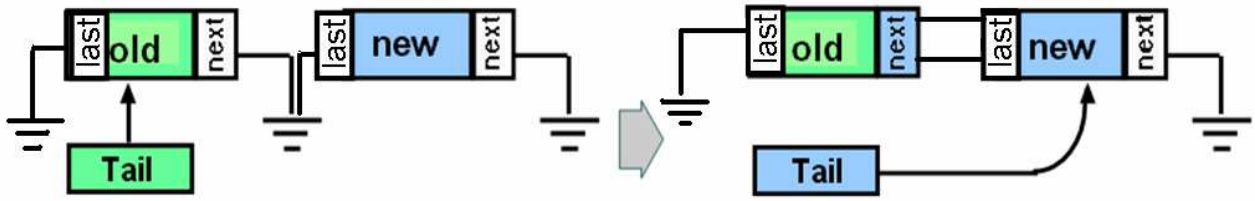
```
#include<iostream.h>
#include<conio.h>
struct list{
int a;
list*next;
list*last;
}
main(){clrscr();
int i,j;list *node,*head,*tile;
node=new list;cin>>node->a;
node->next= node->last =NULL;
tile=head=node;
for(i=1;i<5;i++){node=new list;
cin>>node->a;
node->next=NULL;
node->last=tile;
tile->next=node;
tile=node;
}
node=head;
while(node!=NULL){
cout<<node->a<<" ";
node=node->next;}
```

هنا عرفنا head من نوع list مؤشر لسجل وهو الرأس
و tail من نوع list مؤشر لسجل وهو الذيل
و node سنستخدمه كمتغير لإدخال بيانات العقد .

هنا أول خطوة وهي ضرورية ولابد أن تكون منفردة عن أخواتها لكي نساوي الرأس والذيل بأول عقدة .
الأمر node=new list معناه اصنع لي عقدة جديدة أي سجل بالذاكرة
Node->last= Node->next=NULL أي جعل حقل العقدة لا يؤشر على عقدة أي خالي
ثم ساوينا الذيل والرأس بالعقدة الجديدة حقنا أي بهذا الشكل .

هنا سنكون 4 عقد أضافية بجانب الأولى فيكون لدينا 5 عقد .
في اللوب السطر الأول والثاني قد سبق شرحه أم الثالث
node->last= Tail أي حقل العقدة الثانية يساوي العقدة القديمة إي الذيل.
Tail->next=node هي تحويل حقل الذيل من نل إلى عنوان العقدة الجديدة.
فبهذه الحالة تمت عملية الربط بين العقدتين بقي علينا نقل الذيل إلى العقدة الجديدة tail=node وهكذا بباقي العقد إلى أن ينتهي عمل اللوب ويمكنك تكوين مئات العقد بهذه الطريقة والشكل في الأسفل يبن الشرح .

```
getch();}
```



أم الإضافة من اليسار نفس السابق إلا أن الاختلاف فقط بعملية إدخال العقد الثانية وما بعدها

```
while(i<4){
node=new list;
cin>>node->d;
node->next=head;
head->last=node;
node->last=NULL;
head=node;}
```

ملاحظة
الإضافة من اليمين يكون الرأس متحركا
والإضافة من اليمن يكون الذيل هو المتحرك

ونفس البرامج التي ذكرناها في القوائم الأحادية تطبق على القوائم الثنائية فلاختلاف فقط هو زيادة المؤشر الخلفي وربطة بالعقدة الجديدة.
واليكم هذا الكود لعملية ترتيب قائمة ثنائية

```
#include<iostream.h>
#include<conio.h>
struct list{int a;list*next;list*last;};
main(){clrscr();
int j,i;list *k,*kk,*node,*head,*tile;
node=new list;cin>>node->a;node->next=NULL;node-
>last=NULL;tile=head=node;
for(i=1;i<5;i++){
node=new list;
cin>>node->a;
node->next=NULL;
node->last=tile;
tile->next=node;
tile=node;
}
for(kk=head;kk!=0;kk=kk->next)
for(k=head;k!=0;k=k->next)if(kk->a>k->a)
{j=kk->a;kk->a=k->a;k->a=j;}
node=head;
while(node!=NULL){cout<<node->a<<" ";node=node->next;}
getch();
}
```

عرفنا متغيرين من نوع السجل وتعاملنا بعملية الترتيب كترتيب مصفوفة وهذه الخوارزمية معروفة ولا جديد فيها.

ينبغي على القارئ حل هذا المثال بدون أن ينظر للكود المكتوب

*) اكتب برنامج يعمل على إدخال الإعداد الفردية من اليسار و الزوجية من اليمين ؟

هل اكتشفت فكرة البرنامج فهي سهلة جداً ولا تحتاج إلى جهد وضياع للوقت !

ان لم تتضح لك الفكرة ياعزيزي فصلي على معدن الأسرار ومنبع الأنوار سيدنا محمد وعلى آلة وصحبة الأطهار وتتبع هذا الكود .

```
#include<iostream.h>
#include<conio.h>
struct list{int a;list*next;list*last;};
main(){clrscr();
int j,i;list*node,*head,*tile;
node=new list;
cin>>node->a;
node->next= node->last=NULL; tile=head=node;
for(i=1;i<5;i++){
node=new list;
cin>>node->a;
if(node->a%2==0){
node->next=NULL;
node->last=tile;
tile->next=node;
tile=node;      }
else
{node->next=head;
node->last=NULL;
head->last=node;
head=node;}
}
node=head;
while(node!=NULL){cout<<node->a<<" ";node=node->next;}
getch();
}
```

فكرة البرنامج هي بعد إنشاء أول عقدة يتم إنشاء ثاني عقدة
ويتم إدخال القيمة داخلها وبعد ذلك يتم تفحص القيمة فإذا كانت زوجيه فان الإضافة ستكون من اليمين وإلا ستكون الإضافة من اليسار.
وهذا يتم إلى إن ينتهي إدخال العقد.
وصلى الله على النبي .

وهذا الكود يعمل على حذف إي عقدة بالقائمة وبواسطة الدوال

```
#include<iostream.h>
#include<conio.h>
struct list{int a;list*next,*last;};
void f(list*&,list*&,int);
main(){clrscr();list *node,*head,*h,*tail;
node=new list;
node->a=1;node->next=node->last=
NULL;head=tail=node;
for(int i=1;i<5;i++){
node=new list;
node->a=i+1;
node->last=tail;
node->next=NULL;
```

نلاحظ إننا أرسلنا للدالة بموقع السجل ومؤشر للسجل فلو حصل حذف أو تغير داخل ألداله فسيتغير داخل البرنامج الرئيسي وأيضا لو عرفنا بهذا الشكل
void f(list**,list**,int) فان إي تغير للعقد داخل الدالة فأنة سيتغير داخل البرنامج الرئيسي
أم إذا كان تعريف الدالة بهذا الشكل
void f(list*,list*,int) فان إي عملية حذف للعقد لا تتغير بالبرنامج الرئيسي .
وعلى القارئ إن يطبق جميع ما ذكرته لكي يتأكد.

```
tail->next=node;
tail=node;}
node=head;
while(node!=NULL){cout<<node->a<<" ";;node=node->next;}
cin>>i;
f(*&head,*&tail,i);
node=head;cout<<endl;
while(node!=NULL){cout<<node->a<<" ";node=node->next;}
getch();}
void f(list*&head,list*&tail,int i)
{list*node;node=head;
while(--i>0&&node!=NULL){node=node->next;}
if(node->last==NULL){node->next->last=NULL;head=node->next;delete
node;}
else
if(node->next==NULL){node->last->next=NULL;delete tail;tail=node;}
else
{node->last->next=node->next;
node->next->last=node->last;
delete node;}
}
```

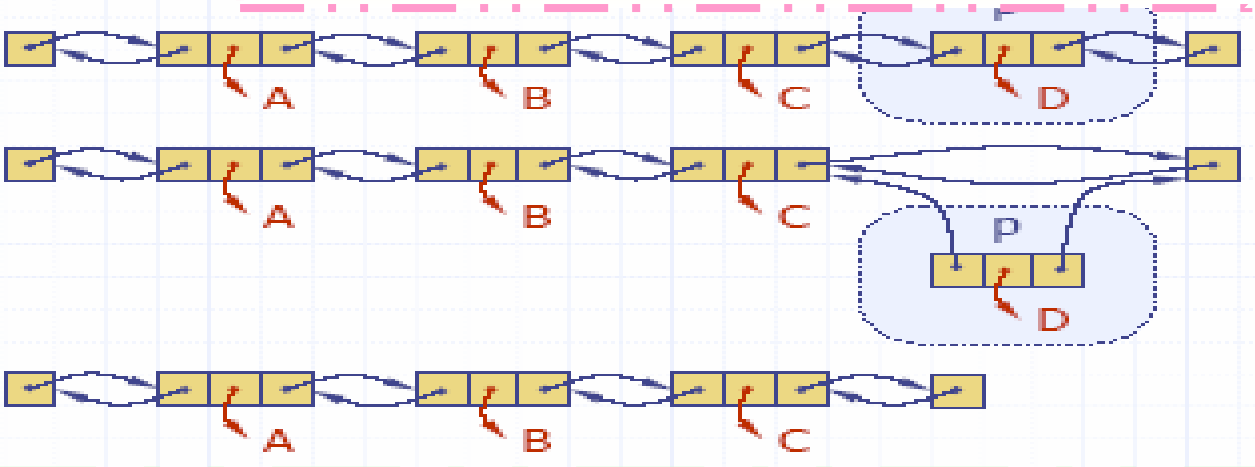
في هذا الشرط نستفسر إذا كانت العقدة هي الرأس فإنها حالة خاصة إي الحذف من البداية وسبق وان تكلمنا عن هذه الحالة وفائدة هذا الشرط هو الحفاظ على مكان الرأس وهو نقلة بمقدار واحد للإمام و بعد ذلك حذف العقدة.

وإلا

في هذا الشرط نستفسر إذا كانت العقدة هي الذيل فإنها حالة خاصةايضاً إي الحذف من النهاية وسبق وان تكلمنا عن هذه الحالة وفائدة هذا الشرط هو الحفاظ على مكان الذيل وهو نقلة بمقدار واحد للخلف و بعد ذلك حذف العقدة

وإلا

ستكون العقدة بين الرأس والذيل فإنها حالة خاصةايضاً وسبق وان تكلمنا عن هذه الحالة فيتم ربط مؤشر العقدة السابقة مع العقدة التالية وربط مؤشر العقدة التالية مع العقدة السابقة والشكل في الأسفل يبن ذلك .



واليكم هذا الكود الذي بعمل على دمج قائمتين ثنائيتين

```
#include<iostream.h>
#include<conio.h>

struct list
{int d;list*next;list*last;};
main(){clrscr();list*head,*tail,*node,*w,*tail2,*head2;
cout<<"ENTRER LIST 1 \n";
node=new list;cin>>node->d;node->next=node->last=NULL;tail=head=node;
for(int i=1;i<5;i++)
{node=new list;cin>>node->d;node->next=NULL;
node->last=tail;tail->next=node;tail=node;}
cout<<"ENTRER LIST 2 \n";
node=new list;cin>>node->d;node->next=node->last=NULL;tail2=head2=node;
for(i=1;i<5;i++)
{node=new list;cin>>node->d;node->next=NULL;
node->last=tail2;tail2->next=node;tail2=node;}

tail->next=head2;head2->last=tail;tail=tail2;
head2=tail2=NULL;delete tail2,head2;
for(w=head;w!=NULL;w=w->next)cout<<w->d<<" ";
getch();}
```

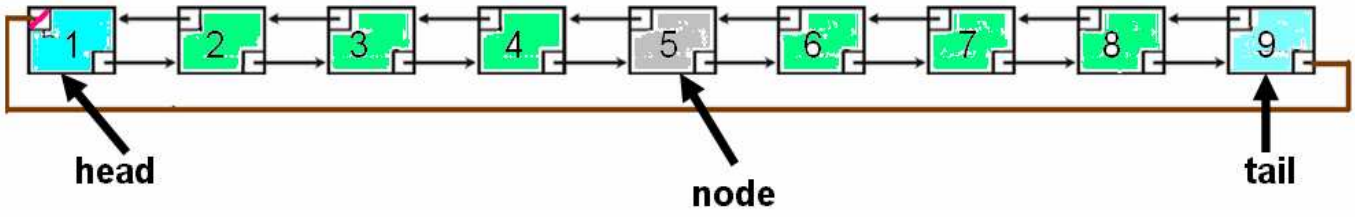
على ما اعتقد أن البرنامج في أتم الوضوح ولا يحتاج إلى شرح أو تعقيب .
وهذا الكود يعمل على حذف المواقع الزوجية في القائمة الثنائية

```
#include<iostream.h>
#include<conio.h>
struct list{int a;list*next,*last;};
main(){clrscr();
list *node,*head,*h,*tail,*k;
node=new list;
node->a=1;node->next-NULL;node->last=NULL;head=tail=node;
for(int i=1;i<20;i++){
node=new list;
node->a=i+1;
node->last=tail;
node->next=NULL;
tail->next=node;
tail=node;}
node=head;while(node!=NULL){cout<<node->a<<" ";node=node->next;}
i=0;cout<<endl;node=head;
while(node!=NULL){
if(i++%2==0)
if(node==head){k=node;head=head->next;head->last=NULL;}
```

```
else
if(node==tail){k=node;tail=node->last;tail->next=NULL;}
else
{k=node;node->last->next=node->next;node->next->last=node->last;}
node=node->next; delete k;}

node=head;while(node!=NULL){cout<<node->a<<" ";node=node->next;}
getch();}
```

إلى هنا قد اتضحت عمل القوائم المذبلة وهذه الاكواد التي كتبت إذا فهمها القارئ فأنني اضمن له أن إي سؤال سيواجه سيجيب علية بلا كلام ولا ضياع للوقت بس يحتاج منه أن يكثر من الصلاة على الحبيب علية أفضل الصلوات والتسليم من اليوم إلى يوم الذين اللهم آمين .

مثال مهم



لنفترض أن لدينا قائمة بداخلها هذه البيانات وهيكل البيانات بهذا الشكل

```
struct list{int d,list*next,*last;};
```

ولدينا ثلاثة مؤشرات node , tail , head

وكان أمر الطباعة

```
Cout<<node->d;                                          =5
Cout<<node->next->next->d;                              =7
Cout<<node->next->last->d;                              =5
Cout<<node-> next-> next-> next-> next-> next->d;       =2
Cout<<head->d;                                          =1
Cout<<head->next->d;                                    =2
Cout<<head->last->last->d;                              =NULL
Cout<<tail->last->last->next->d;                        =8
Cout<<tail->next->last->d;                              =NULL
```

إذا عرف القارئ تتبع هذا البرنامج بدون ما ينظر للحل فأنني أبارك له وأهنيه على هذا ألانجاز فألف ألف مبروك يا عزيزي .

خلاصة

بإمكاننا أن نقدر ثمن خوارزم معالجة اللوائح بواسطة :

- الحجم المشغول في الذاكرة .
- عدد المؤشرات التي من الواجب عبورها أو استعمالها.

تمارين على الفصل

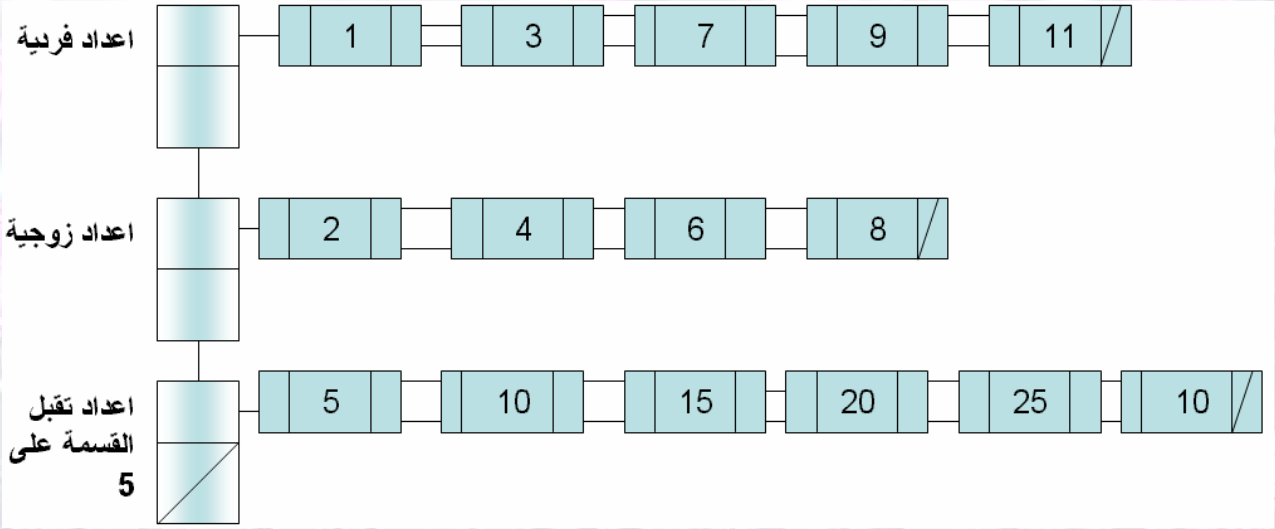
*) هذا السؤال للقارئ النبيل ----- بواسطة القائمة الأحادية اكتب برنامج يعمل عمل القائمة الثنائية إي يكون الهيكل العام لقائمه مكون من متغيرين فقط إي next , int لا يوجد مؤشر يؤشر للخلف مثل last فيستطيع العودة للخلف مرة ومرتين ..... الخ ؟

*) بواسطة القوائم ألثنائيه اكتب برنامج يقوم حذف المواقع الأولية من هذه القائمة ومعروف إن الإعداد الأولية (1,2,3,5,7,9,11,13,.....) ؟

*) بواسطة ألعوديه أو التكرار اكتب برنامج يعمل على طباعة عدد العقد الثنائية ؟

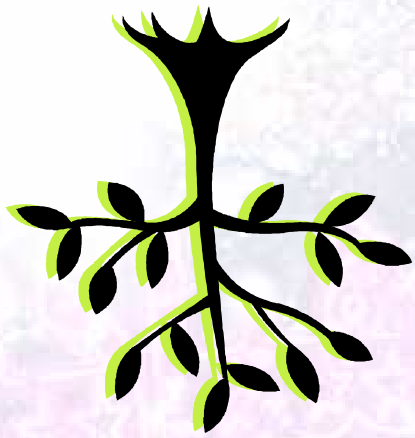
*) جميع الأسئلة التي ذكرت في الفصول السابقة ينبغي على القارئ تطبيقها بالقوائم الثنائية ؟

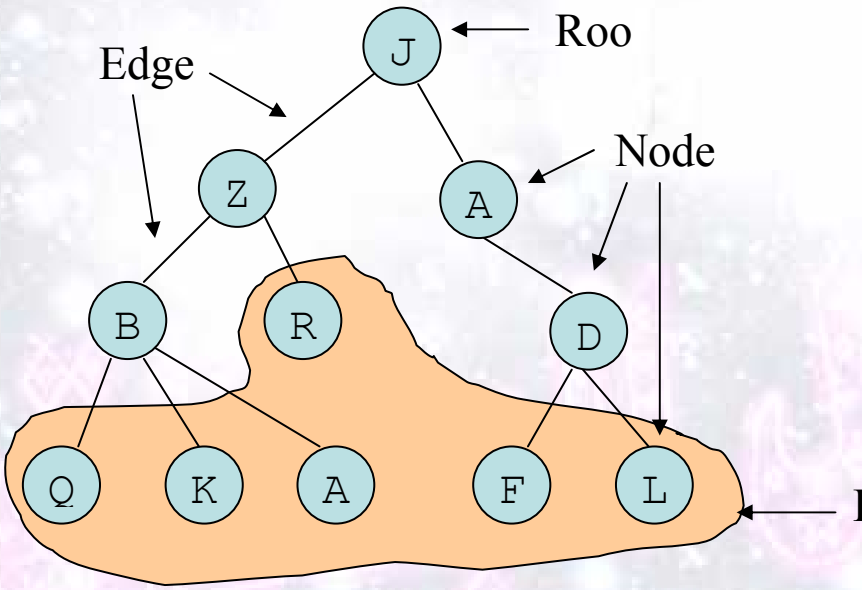
*)اكتب برنامج يقوم بتمثيل بيانات هذا الشكل في الذاكرة مع مراعاة نوع القوائم ؟



.الأشجار Trees

هي نوع من هياكل البيانات المتشعبة إي الغير مرتبة فتتكون من عقد وسيقان وأوراق و عدد محدد من العناصر تسمى نقاط تفرع(nodes) لكل شجرة نقطة تفرع وحيدة تسمى نقطة التفرع الجذرية root node حيث إن باقي النقط الفرعية هي مجموعة من شجرات ..... t1 ,t2,t3,t4 . وإذا لم نجد ذرية لنقطة تفرع فتسمى ورقة (leaf node) فتكون طريقة لربط المعلومات المختلفة فيما بينها على هذا الشكل .





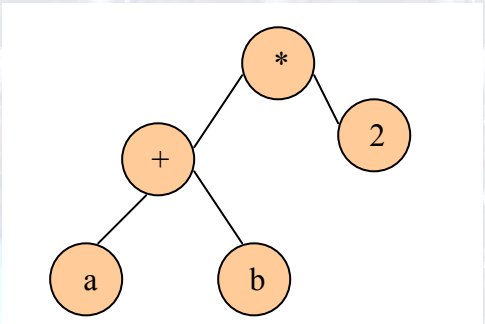
العناصر الأساسية للشجرة

- ✓ الجذر root (J) الموضح بالشكل الأعلى وتدعى عقدة الأصل أو الجد .
- ✓ العقد A ,Z تسمى الأبناء .
- ✓ الربط بين الأصل والابن يسمى (branch) .
- ✓ العقدة التي لا يوجد لها أبناء فتسمى ورقة مثل Q,K,F,L (leave) كما في الشكل الموضح.
- ✓ العقدة Q لها عم وهو R .

الهيكل الفرعي / هو مجموعة جزئية نم الهيكل الأصل وهو نفسه يشكل هيكلاً (tree) وبالنظر للشكل السابق نجد أن Z,A يشكل جدراً لهيكلاً فرعياً من الهيكل الأصلي .

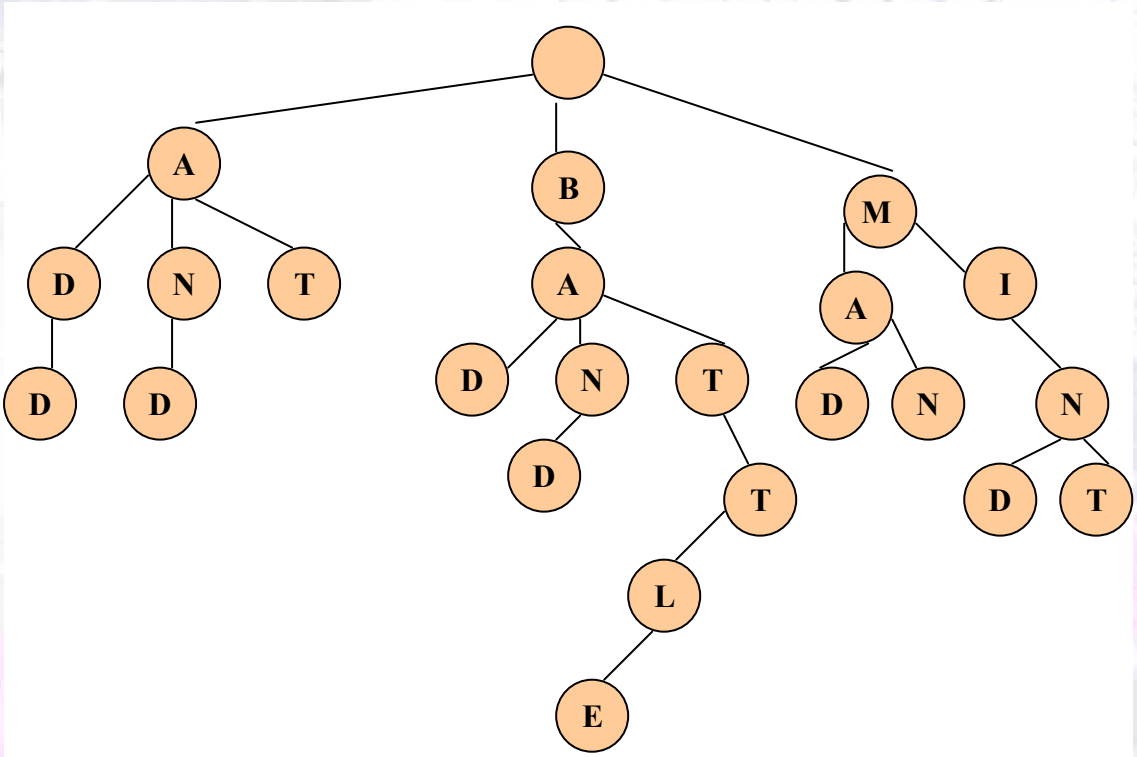
ومن التطبيقات المستخدمة في الأشجار

- ✓ التعابيرالرياضيه مثل 2*(a+b) فيكون



- ✓ السجلات / فيمكن تمثيل السجل بالأشجار.
- ✓ المجموعات .
- ✓ القواميس / فليكن لدينا هذه الكلمات AND,AT,ADD,BDDMBAND,BATTLE

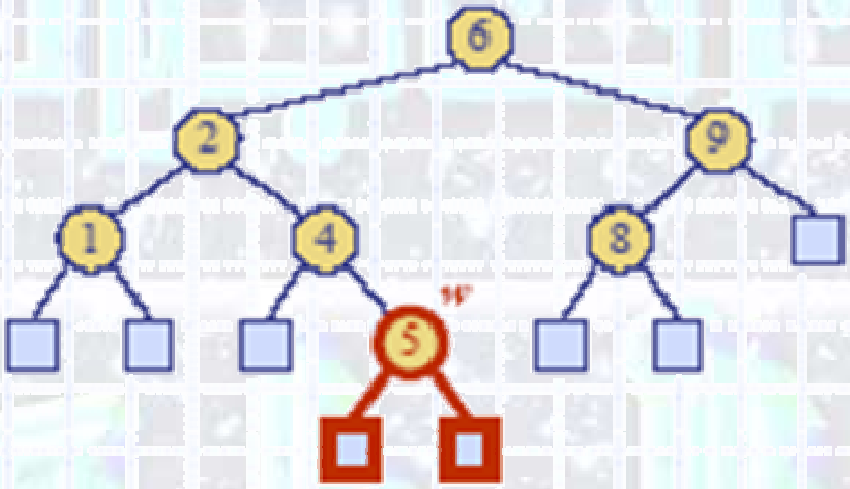
فيكون الشكل الشجري لهذه الكلمات بهذا الشكل



فلاحظ كيف تم تمثيل تلك الكلمات وكان عدد حروف الكلمات يساوي 35 حرفاً بينما عدد العقد أصبح يساوي 24 فهذا يعني انه تم اختصار عدة مواقع لخزن الأحرف في الذاكرة وهذه الو ضيفة من أهم فوائد الأشجار .

- ✓ جدول القرارات / ويعتبر جدول القرارات من الجداول المهمة في تحليل المسائل التي تحتاج إلى برمجة .وتساعد في تحليل الأنظمة المعلوماتية .

الشجرة الثنائية / هي الشجرة التي يكون لكل عنصر منها ذريتين فتسمى الذرية اليسرى وتكون اصغر من الجذر والذرية اليمنى فتكون اكبر من الجذر ومرتبة ولا تسمح بتكرار القيم .



العمليات المختلفة على الهياكل الثنائية وخوارزميتها

- ✓ المرور المتناسق (Inorder taversal) للهيكل الثنائي بشكل ثابت (Infix) .
- ✓ المرور ما قبل الاتساق (preorder traversal) لنفس الهيكل إلى شكل ما قبل الثابت (prefix) للتعبير الرياضي.

✓ المرور ما بعد الاتساق (postorder traversal) لنفس إلى الهيكل إلى شكل ما بعد الثابت (postfix) للتعبير الرياضي .

خلاصة العمليات / يتم عبور الشجرات الثنائية حسب الطرق الأساسية الثلاثة: preorder , inorder ,postorder . كل ترتيب يحدد متى تتم زيارة الجذر .

ففي preorder يزار الجذر أولا , و inorder يزار الجذر بعد الذرية اليسرى , و postorder يزار الجذر اخيراً .

طرق تمثيل الشجرة الثنائية :

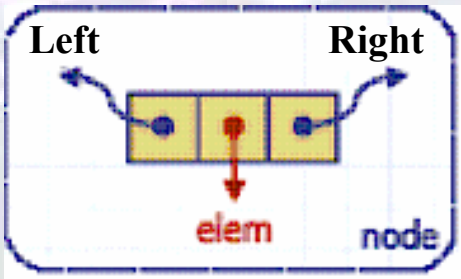
أن الشجرة تمثل بعدة طرق بواسطة القوائم, وتمثل بالمتجهات وسندرس فيما يلي هذه الطرق:

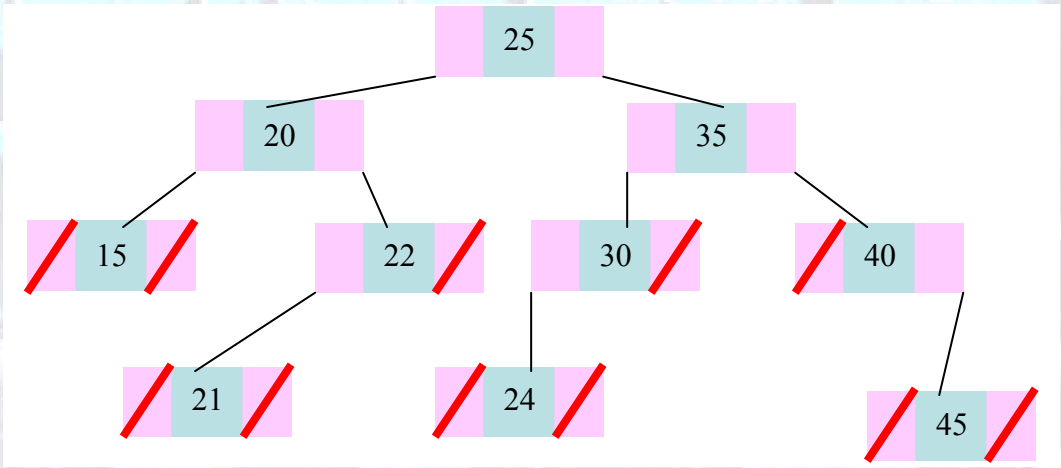
- ❖ إذا كانت الشجرة ثنائية منتظمة بعمق (H) فإن عدد عناصر هذه الشجرة يساوي (2^(H+1)-1) وعلية فإنها تمثل بمصفوفة أحادية البعد و عدد عناصر المتجة الذي يمثل الشجرة يساوي (2^(H+1)-1) . ومميزات هذه الطريقة
  - ☒ السهولة فإذا أعطيت موقع العقدة الابن فمن السهل تحديد موقع الأب بالنسبة لها. فلو كانت العقدة الابن في الموقع n من المصفوفة فإن موقع الأب يكون صحيحاً(n/2).
  - ☒ تطبق بسهولة في لغات البرمجة , مثل بيسك وفورتران حيث تكون مواقع الذاكرة الثابتة متوفرة مباشرة .

وعيوب هذه الطريقة

- ☒ عملية الإضافة والحذف تؤدي إلى تحريك البيانات إلى أعلى وأسفل في المصفوفة وهذا يضيع وقت المعالجة .
- ☒ تكون هنالك مواقع ذاكرة غير مستغلة

- ❖ تمثيل الشجرة بواسطة القوائم :





ونلاحظ من الشكل السابق أن كلما اتجهنا إلى اليمين يكون العدد اكبر من السابق وكل ما اتجهنا إلى اليسار كان العدد اصغر من الأب أو العقدة السابقة وهكذا تكون الأعداد مرتبة تصاعدياً وتنازلياً .

ونكرر أن الأعداد لا تتكرر في الشجرة الثنائية.

ونلاحظ أن من عيوب هذه الطريقة

- ☒ تحتوي على فراغ في فضاء الذاكرة غير مستعمل نتيجة لاستخدام مؤشر صفرية NULL.

☒ خوارزمية التطبيق لها أصعب اللغات في اللغات التي لا تعطي تقنية ذاكرة متحركة (ديناميكية) .

خوارزمية بناء الشجرة الثنائي

أن بناء الشجرة الثنائية يعتمد على طريقة عبورها, وسابقاً قد سقنا عدة طرق لعبور الشجرة بالاعتماد على المؤشرات وفي الخوارزميات التالية جميعها سنفترض الشجرة الثنائية ذات مؤشر الإباء بالأبناء (Father Link) . وعملية إضافة عقدة في الشجرة الثنائية تعتمد على قيمة تلك العقدة , فإن كانت القيمة اكبر من الجذر اتجهنا إلى اليمين والعكس نتجه إلى اليسار

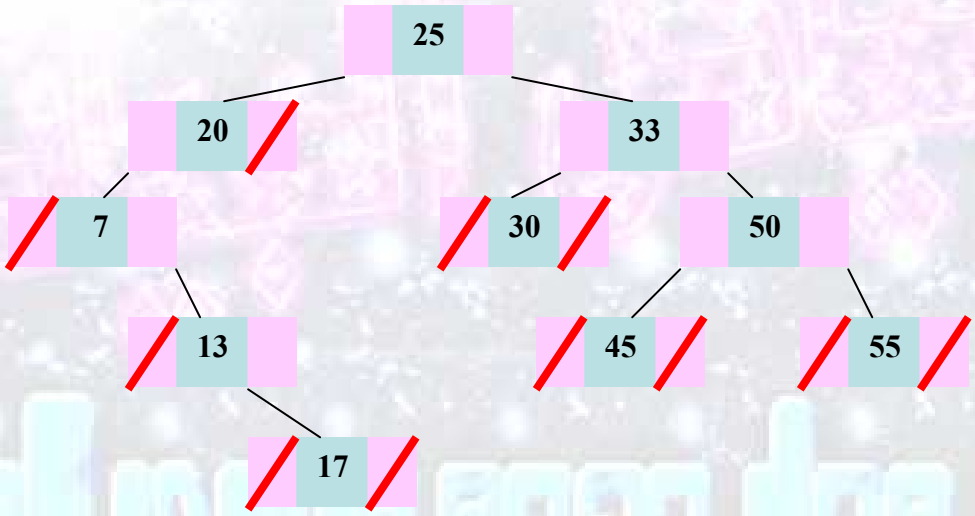
وخلاصة هذه الخوارزمية تتلخص بالاتها

☒ ضع العنصر الأول على أساس أنة العقدة الأول في الهيكل (الجذر) ز

☒ والعدد الأتي إذا كان العنصر المراد إدخاله اكبر من الجذر سنضعه على يمين الجذر .

☒ وإلا على يسار الجذر .

والشكل التالي يبين ذلك . فإن أدخلنا هذه القيم (25,20,7,13,33,50,45,17,30,55) ستكون الشجرة بهذا الشكل



لقد بدئنا بالجدر (25) ثم انتقلنا إلى اليسار بالعدد 20 لأنة اصغر من 25 ثم انتقلنا إلى يسار 25 و 20 بالعدد 7 لأنة أصغر من 20 ثم انتقلنا بالعدد 13 إلى اليسار من 25 وهكذا إلى نهاية الأعداد

وفيما يلي البرنامج الذي ينفذ جميع ما سبق

```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*);
void main(){clrscr();
int h;
tree*node,*s,*p,*root=NULL;
for(int i=1;i<9;i++){node=new tree;
cin>>node->d;node->left=node->right=NULL
if(root==NULL)root=node;
else
insert(root,node);}
print(root);
getch();}
```

قمنا اولاً بعمليه البحث عن الموقع التي سنضع العقدة حسب خوارزمية الاشجار

فإذا كانت القيمة اكبر اتجهنا يساراً وإلا اتجهنا يميناً الى ان يصل s =NULL

وفائدة p هو عبارة مؤشر يؤشر بمقدار واحد للخلف .

بعد الانتهاء من بحث الموقع نقوم بعمليه استفسار فإذا كانت القيمه اكبر من العقدة التي عثرنا عليها بواسطة p فأننا نضع العقدة على اليمين وإلا نضعها على اليسار وهكذا لباقي العقد .

```
void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}

print(tree*node){
if(node!=NULL){
print(node->left);
print(node->right);
cout<<node->d<<" ";
}}
```

عملية الطباعة هنا تكون بالاستدعاء الذاتي
فهي اسهل من غيرها

خوارزمية البحث عن عنصر معين بداخل شجرة ثنائية :

لمعرفة فيما إذا كانت قيمة معينة موجودة بداخل شجرة معينة يجب أن نكتب خوارزمية البحت التالي

- ☒ إذا كانت القيمة المسجلة في الجذع تساوي القيمة المدخلة فاننا قدحصلنا عليها فنتوقف عن البحث .
- ☒ وإلا فانه سينتج حالتين
  - إذا كانت القيمة المدخلة اكبر من العقدة فسنذهب الى اليمين وان وجدت نتوقف .
  - إذا كانت القيمة المدخلة اصغر من العقدة فسنذهب الى اليسار وان وجدت نتوقف .

خوارزمية عد عقد الشجرة :

إن حجم الشجرة يساوي عدد العقد في الشجرة الفرعية اليمنى مضافاً اليها عدد العقد في الشجرة الفرعية اليسرى مضافاً إليها عقدة الجذر وهذا الكود الذي سيقوم بهذه العملية

```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*);int x=0;
void main(){clrscr();
int h;
tree*node,*s,*p,*root=NULL;
for(int i=1;i<9;i++){node=new tree;
cin>>node->d;node->left=node->right=NULL;
if(root==NULL)root=node;
else
insert(root,node);}
print(root);cout<<endl<<"TOTAL NODE ="<<x;
getch();}

void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}
print(tree*node){
if(node!=NULL){
```

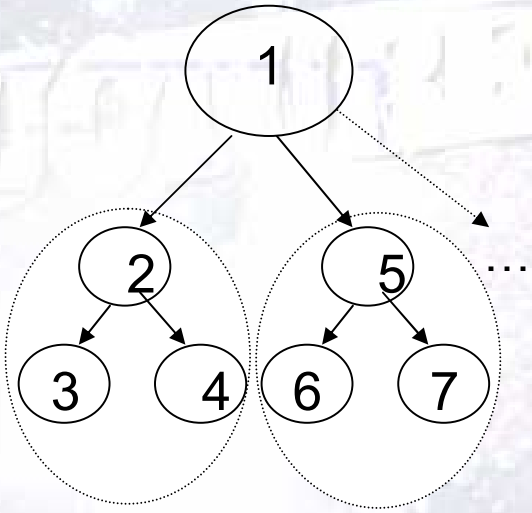
```
print(node->left);
print(node->right);
cout<<node->d<<" ";
x++;
}}
```

واليكم هذا الكود فانة يعمل على طباعة اب واخ العدد المدخل

```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*,int);
sc(tree*,int);
void main(){clrscr();
int h,a[15]={49,83,28,18,40,11,19,32,44,71,97,69,72,92,99};
tree*node,*s,*p,*root=NULL;
for(int i=0;i<15;i++){node=new tree;
node->d=a[i];node->left=node->right=NULL;
if(root==NULL)root=node;
else
insert(root,node);}cin>>i;
print(root,i);
getch();}
void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}

print(tree*node,int x){tree*p;int t=0;
while(node!=NULL)
{if(node->d==x){t=1;break;}p=node;if(node->d>x)node=node->left;else
node=node->right;}
if(t==1){
cout<<p->d<<"   FATHER";
if(p->left->d==x)cout<<endl<<p->right->d<<"   BRATHER";else
cout<<endl<<p->left->d<<"   BRATHER";}
else
cout<<"NOT FOUND NUMBER";
}
```

للنظر الشكل الاتي

شرط الحلقة هو طالما node ما تساوي نل وكمان node ما تساوي العدد نفسه المدخل .
هذا المتغير p يمشي ورآك بمقدار خطوة إي قبل أن ننتقل إلى اليمين أو إلى اليسار فأننا نخزن موقعنا فيه ثم ننتقل فبهذا الشكل سيكون p هو الأب ل للعقدة ..



فلو ادخلنا الرقم 6 فانة سوف يطبع لنا الاب وهو 5 ويطبع لنا الاخ وهو 7 .

مثال

يراد منك طباعة العقد التي في الجهة اليمنى فقط فماذا ستفعل كيف ستضع الكود فكر وفكر وصلي على نبيك صلاة دائمة وتتبع الكود في الأسفل

```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*);
sc(tree*,int);int k=0;
void main(){clrscr();
int h,a[15]={49,83,28,18,40,11,19,32,44,71,97,69,72,92,99};
tree*node,*s,*p,*root=NULL;
for(int i=0;i<15;i++){node=new tree;
node->d=a[i];node->left=node->right=NULL;
if(root==NULL)root=node;
else
insert(root,node);}
print(root);
getch();}

void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}

print(tree*node){static tree*p=node;
if(node!=NULL){
print(node->left);
if(node->d>=p->d)cout<<node->d<<" ";
print(node->right);
```

```
}}
```

هذا الكود المكتوب لا يوجد بة إي جديد اللهم أضفنا جملة static ومعناها حفظ قيمة المتغير في الذاكرة عندما تعيد الداله نفسها فتبقي قيمة المتغير محفوظة في الذاكرة هذا وصلى الله على الهادي وسلم.

وهذا الكود يعمل على بحث عن عقدة وطباعة موقعها بين العقد

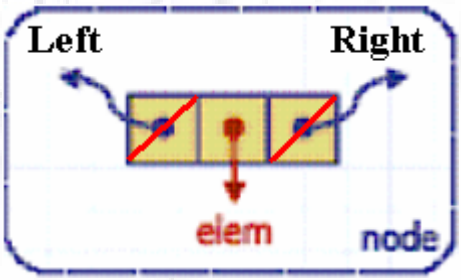
```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*);
sc(tree*,int);
void main(){clrscr();
int h;
tree*node,*s,*p,*root=NULL;
for(int i=1;i<9;i++){node=new tree;
cin>>node->d;node->left=node->right=NULL;
if(root==NULL)root=node;
else
insert(root,node);}
print(root);
cin>>h;
sc(root,h);
getch();}

void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}

print(tree*node){
if(node!=NULL){
print(node->left);
cout<<node->d<<" ";
print(node->right);
}}
sc(tree*node,int h){int j=0,t=0;
while(node!=NULL){j++;
if(node->d==h){cout<<"yes  "<<j ;t=1;}
if(node->d<h)node=node->right;else node=node->left;}
if(t==0)cout<<endl<<"no";
}
```

خوارزمية حساب عدد أوراق الشجرة الثنائية :

إن الورقة في الشجرة هي العقدة التي ليس لها أبناء كما قلنا سابقاً , ولحساب عدد أوراق الشجرة , لابد اولاً من التحقق من أن العقدة هي ورقة أم لا وذلك بهذا الشرط

if(node->left==NULL&&node->right==NULL)

فيكون شكل الورقة بهذا الشكل



وهذا الكود لهذه العملية

```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*);
sc(tree*,int);int k=0;
void main(){clrscr();
int h;
tree*node,*s,*p,*root=NULL;
for(int i=1;i<9;i++){node=new tree;
cin>>node->d;node->left=node->right=NULL;
if(root==NULL)root=node;
else
insert(root,node);}
print(root);cout<<endl<<"TOTAL NODE  = "<<k;
getch();}

void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}

print(tree*node){
if(node->left==NULL&&node->right==NULL)k++;
if(node!=NULL){
print(node->left);
cout<<node->d<<" ";
print(node->right);
}}
```

وهذا الكود يطبع جميع أباء العدد المدخل

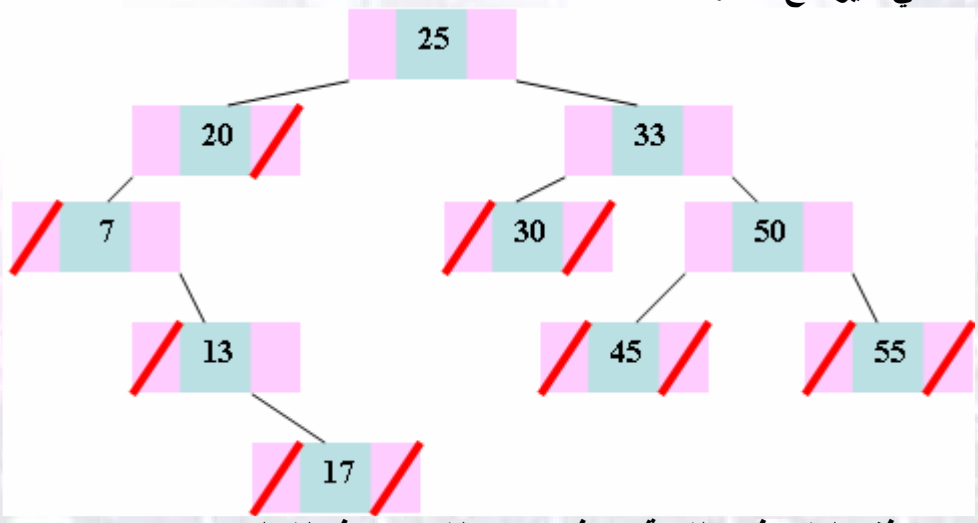
```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
```

```
void insert(tree*,tree*);
print(tree*);
int s(tree*,int);
void main(){clrscr();int t[10]={50,69,100,4,3,9,74,15,68,33};
int h;
tree*node,*k,*b,*root=NULL;
for(int i=0;i<10;i++){node=new tree;
node->d=t[i];node->left=node->right=NULL;
if(root==NULL)root=node;
else
insert(root,node);}
cin>>h;b=root;k=NULL;
while(b){if(h<b->d)b=b->left;else
        if(h>b->d)b=b->right;else break;k=b;}
b=root;
while(k){if(k==b)break;
cout<<b->d<<" ";if(k->d<b->d)b=b->left;else b=b->right;}

cout<<endl;print(root);
getch();}

void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>=s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}

print(tree*node){
if(node!=NULL){
print(node->left);
print(node->right);
cout<<node->d<<" ";
}}
```

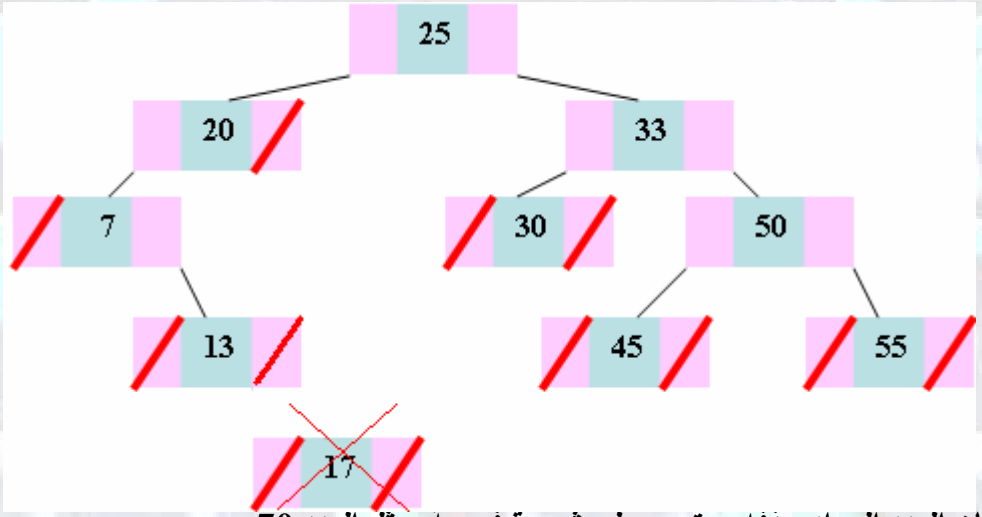
خوارزمية حذف العقد :

- إذا كان العدد المراد حذفه ورقة لا يحتوي على أبناء فالأمر سهل ولا توجد مشقة

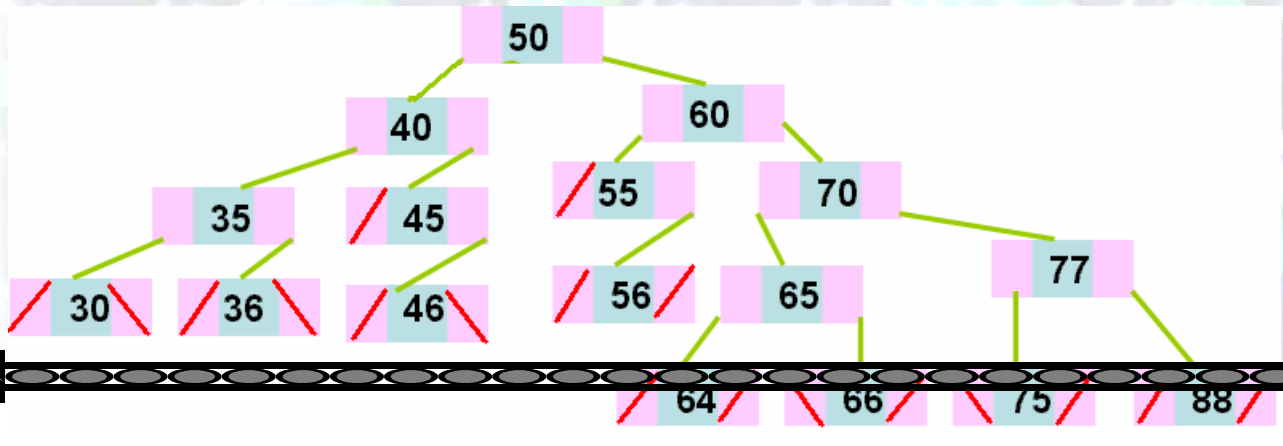
1. نبحث عن العدد وقبل التنقل إلى اليمين أو إلى الشمال نخزن موقعنا في متغير ثم ننتقل وهذا المتغير سيكون يمشي ورآنا بمقدار واحد للخلف إي يكون أب الموقع الحالي .
2. إثناء عمليه التنقل نستفسر عن نوع العقدة فإذا كانت ورقة أم لا بهذا الشرط if(node->left==NULL&&node->right==NULL) .
3. بعد العثور على العدد بقي علينا أن نتنين هل هو على يسار الأب أم على يمينه فأن كان على يساره نجعل حقل left=NULL فنكون عزلنا العقدة من الشجرة ثم نحذف العقدة ومن الشكل الأتي سيوضح ذلك .



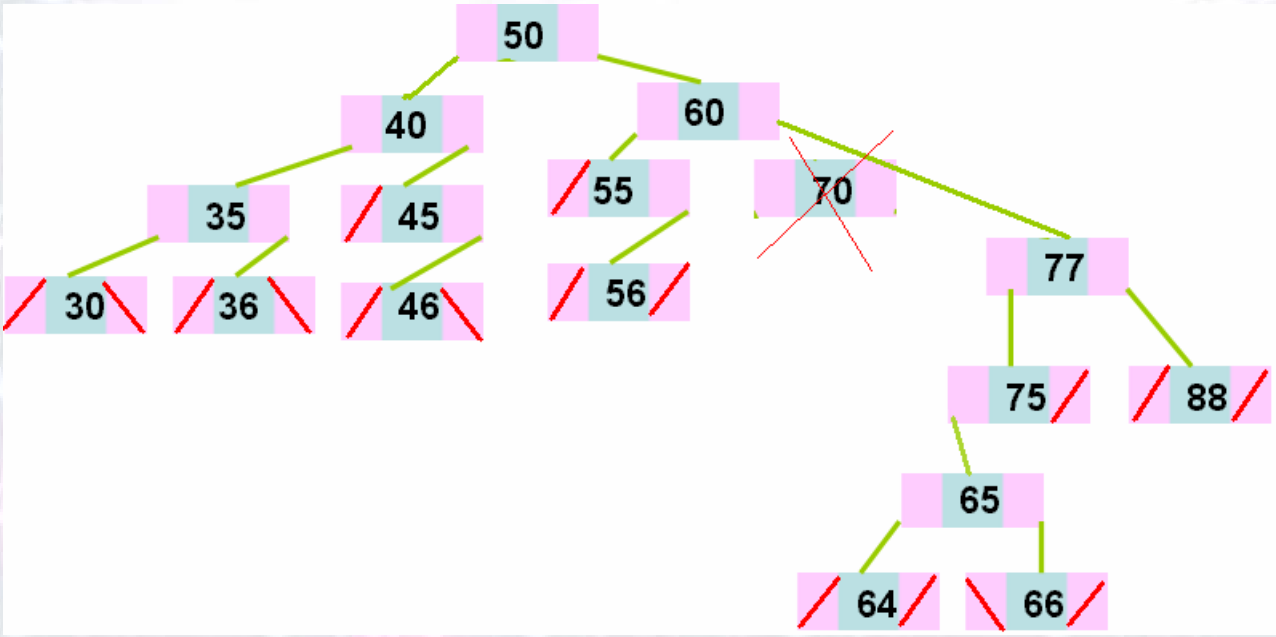
نريد حذف العقدة ذات القيمة 17 فستصبح الشجرة بهذا الشكل



- إذا كان العدد المراد حذفه يحتوي على شجرة فرعيه مثل العدد 70



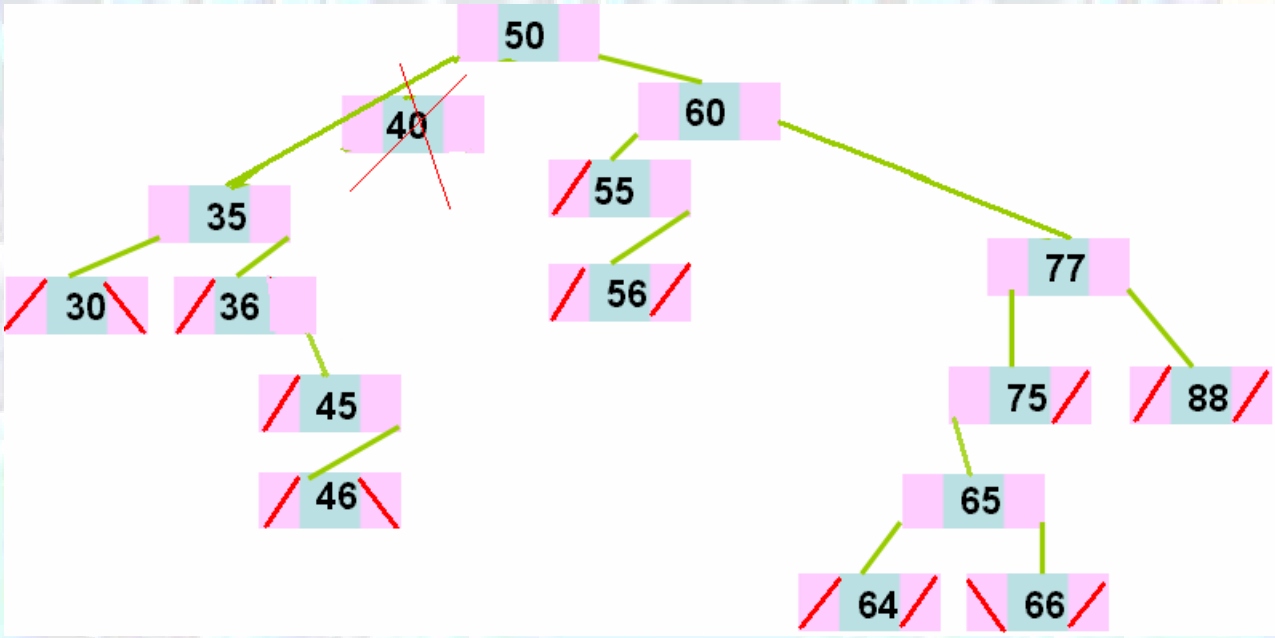
1. نحدد موقع العدد هل هو على يسار الأب أم على يمينه if(60>70).
2. هل العدد يوجد على يمينه فرضاً أعداد وشجرة فرعية if(node->right!=NULL) فإذا تحقق الشرط فأننا نربط يمين الأب بيمين الابن فنكون في هذه الحالة عزلنا 70 بقي علينا ربط أجزاء الابن وهو نصل لأصغر قيمة في العقدة 77 ونربط يسارها بالعقد التي كانت متصلة بالعدد المراد حذفه فتصير الشجرة بهذا الشكل .



3- تم نحذف العقدة .

هذا إن وجدت على يمين العدد عقد وان لم توجد نربط يمين الأب بيسار الابن مباشرة

- ❖ إذا كان العدد اصغر من الأب نفس الخطوات السابقة .
- ❖ إذا كان العدد يوجد بيساره تفرع تربط يسار الأب بيسار الابن ونصل لأكبر عدد داخل التفرع ونربطه بيمين العدد مثل العدد 40 في الشكل السابق فتصير الشجرة بهذا الشكل

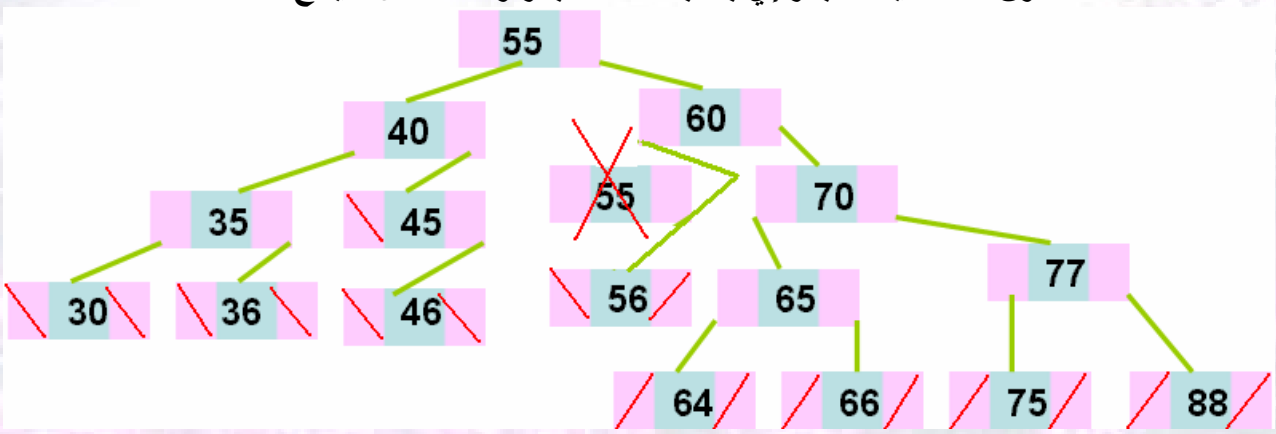


وإلا نربط يسار الأب بيمين الابن , ثم نحذف العقدة .

❖ بقي علينا حالة وهي أن أراد حذف الجذر هي بعض الشيء مربكة إلا أنها بسيطة

☒ نبحث أولا عن اصغر عقدة في الجذع الأيمن للجذر ونطبق جميع الشروط التي ذكرناها سابقاً .

☒ ونأخذ قيمة العقدة ونساوي قيمة الجذر بها ثم نحذف العقدة التي بحثنا عنها وبهذا نكون أحللنا قيمة الجذر إي بمثابة حذفنا الجذر وهذا الشكل سينتج .



☒ وإن لم يوجد تفرع يمين للجذر فإننا ننقل الجذر بمقدار واحد لليسار ونحذف العقدة .

وهذه كل الخطوات يمكن دمجها ببرنامج شامل يستطيع أن يحذف من إي مكان واليكم الكود هدية مني ولا يوجد في أي مرجع .

```
#include<iostream.h>
#include<conio.h>
struct tree{int d;tree*left,*right;};
void insert(tree*,tree*);
print(tree*);
del(tree*&,int);
void main(){clrscr();
int k,h,a[]={50,60,70,55,56,65,64,66,75,77,80,40,45,35,46,36,30};
tree*node,*s,*p,*root=NULL;
for(int i=0;i<=6;i++){node=new tree;
node->d=a[i];node->left=node->right=NULL;
if(root==NULL)root=node;else insert(root,node);}
print(root);cout<<endl<<"ENTER VAL DELETET\n";
for( i=0;i<=7;i++){cin>>k;
del(root,k);
print(root);cout<<endl;}
getch();}

void insert (tree*root,tree*node){tree*p,*s;
s=root;
while(s!=NULL){p=s;if(node->d>s->d)s=s->right;else s=s->left;}
if(node->d>p->d)p->right=node;else p->left=node;}
```

```
print(tree*node){
if(node!=NULL){
print(node->left);
print(node->right);
cout<<node->d<<" ";}
}
del(tree*&node,int h){int r=h,b,j=0;tree *t,*tt,*p,*pp;
if(node->d==h){if(node->right!=NULL)t=node->right;else t=node;
while(t!=NULL){if(t->left==NULL)break;t=t->left;}b=1;r=h=t->d;}
t=node;
while(t!=NULL){
if(t->d==h){j=1;break;}tt=t;if(t->d>h)t=t->left;else if(t->d<h)t=t->right;}

if(j==1&&t->left==NULL&&t->right==NULL){j=3;if(t->d>tt->d)tt-
>right=NULL;else tt->left=NULL;}
if(j==1){
        if(t->d>tt->d){
        if(t->right!=NULL){tt->right=t->right;
        p=t->right;while(p!=NULL){pp=p;p=p->left;}
        pp->left=t->left;}else tt->right=t->left;}
        else
        {if(t->left!=NULL){tt->left=t->left;
        p=t->left;while(p!=NULL){pp=p;p=p->right;}
        pp->right=t->right;}else tt->left=t->right;}
        }
if(b==1)node->d=r;
if(j==0)cout<<"\nNOT FOUND "<<h<<endl;else delete t;
} .
```

إلى هنا ينتهي حديثنا عن الأشجار بقي علينا بسرد هذه التمارين وعلى القارئ يتدرب عليها ويجيب عليها

تمارين الفصل

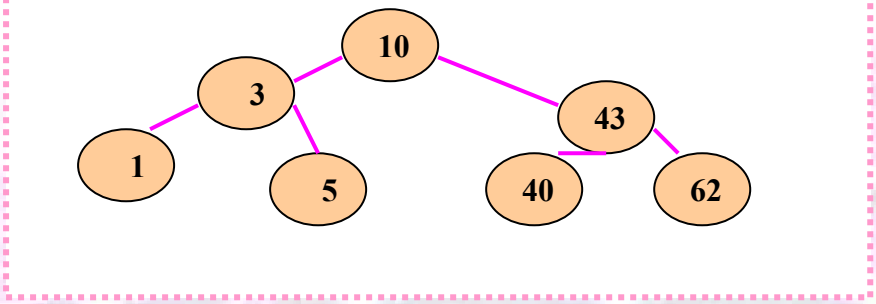
*) اكتب برنامج بواسطة الأشجار الثنائية يقوم بطباعة الشجرة بدون استخدام الاستدعاء الذاتي ؟

*) اكتب برنامج بواسطة الأشجار الثنائية يطبع مجموع ما تحت العدد المدخل وكم أعداد اكبر منة وكم أعداد اصغر منة ؟

*) اكتب برنامج بواسطة الأشجار الثنائية يقوم بطباعة الشجرة مرتبة تصاعدي ومرة تنازلي ؟

*) اكتب برنامج بواسطة الأشجار الثنائية يطبع كم عدد الآباء الذين لديهم أبناء اثنان وكم الذين لديهم ابن واحد وكم الذين ليس لهم أبناء ؟

*) ليكن لديك البيانات التالية (10,3,43,62, 5,1,40)



المطلوب منك طباعه هذه الشجرة أو إي شجرة بالشكل الأتي

```
            62
        43
            40
10
            5
        3
            1
```

*) اكتب برنامج يقوم بتحويل شجرة ثنائية إلى طابور بشرط أن تدخل البيانات مرتبة للطابور وبدن استخدام إي خوارزمية ترتيب ؟

*) اكتب برنامج يقوم بتحويل طابور إلى شجرة ثنائية علماً أن البيانات في الطابور متكررة والشجرة الثنائية لا تقبل القيم المتكررة ؟

*) لديك الكلمات التالية (SAMI , AMMAR , AHMED , BASSAM , SANAD , MOSTAFA , READ , ALI , KAMAL ,AMIN) المطلوب تكوين شجرة ثنائية مثل المثال الذي تكلمنا علية سابقاً ؟

*) اكتب برنامج يقوم بحذف الأعداد الأولية من الشجرة الثنائية ؟

المصادر

- أصول البرمجة بلغة C++ / عمار محمد عيسى الدبعي .
- مدخل إلى هياكل المعطيات بلغة باسكال / د. عبد الحسن الحسيني .
- هياكل البيانات والبرمجة بلغة باسكال / مسعود عمر سعيد نصرو .
- مقدمة إلى لغة السي / ستيف كرنسون , تأليف / عامر بواب .
- www.pitt.edu/~stephenp/INFSC0015
- http://www.csd.abdn.ac.uk/~bjin/CS2005Tutorial

الخاتمة

إلى كل من انتفع بهدى الكتاب

إلى كل مسلم ومسلمة

إلى كل من يشهد أن لإله إلا الله محمد رسول الله

ارجوا الدعاء لي ولمن يصلي على النبي الأمي

صلوات الله وسلامه عليك يا حبيبي محمد ابن

عبد الله

والحمد لله.