

كتاب تعليمي

خطوة بخطوة لتعلم لغة C, C++

Learn C++ or C

Hussien Ahmmed Taleb



Step by Step



الأهداء

اهدي هذا الكتاب إلى أمي ومن غير أمي يستحق هذا الإهداء فلك يا أمي اهدي هذا الكتاب
يا أطيب من رأت عينا في الدنيا فجزأك الله عني وأخوتي خير الجزاء أطل الله لنا بعمر ك

حسين احمد طالب الربيعي

العراق / جامعة ديالى

هندسة الحاسبات والبرمجيات

المرحلة الرابعة

2011/9/1

المدونة

<http://hussienahmmed.blogspot.com/>

البريد الالكتروني

hussien89aa@yahoo.com

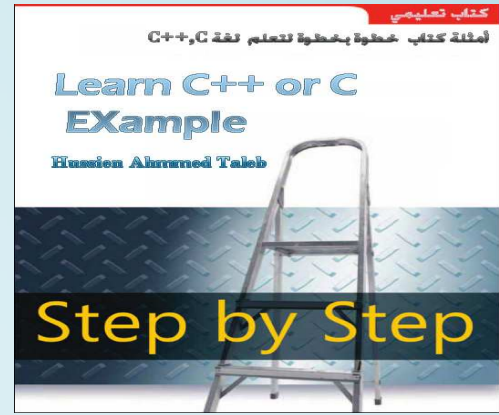
عن الكتاب

يتناول هذا الكتاب لغة (C,C++) بأسلوب تدريسي وبشكل تفصيلي جدا حيث يستطيع منه المبتدأ جدا البدء بتعلم هذه اللغة وكتابة البرامج والمطور يطور قدراته أكثر وتجد انه عند كتابة أي برنامج سيوضح لك الخطوات البرمجية وكيف حدثت

المرفقات مع الكتاب

هناك مرفقات مع الكتابة لكي تكتمل الطريقة التعليمية بشكل صحيح وهي

- مجموعة كبيرة من الأمثلة المحولة بكتاب مرفق



- برنامج تعليمي مصمم (vb.net2008) يقوم باختبار قدراتك في لغة البرمجة (c ,c++)

الأسئلة وحلولها | طباعة متخصصة | اختبار ذكاءك ٢٠

ماهي أكواد الأسطر البرمجية المعقودة في حل السؤال التالي
برنامج يدخل عشرين رقم وبين أولي وبحسب عدد الاعداد الاولية التي ظهرت (ورد في النهائي)

سؤال جديد

الأجابة الصحيحة

درجة الاختبار: سهل متوسط صعب

```
1. #include<stdio.h>
2.
3.
4. int a[20];
5. h=t=0;
6. for(x=0;x<20;x++)
7. scanf("%d",&a[x]);
8. for(x=0;x<20;x++) {
9.
10. if(a[x]%i==0)
11. t=1;
12. if (t==1)
13. printf("%d is no prime\n",a[x]);
14.
15. h=h+1;
```



كيف تبرمج او كيف تكتب برنامجا

كثيرا منا من يجد صعوبة في كتابة البرامج أو انه يعرف كل شيء وفاهم لكل مكونات تلك اللغة ولكن لا يعرف كيف يربط بين تلك المحتويات كتلك التي تعرف مكونات كل أكلة وكيفية تكوينها لكنها لا تجيد الطبخ اعلم أن الكثير ممن سيقراً هذه الأسطر يجد أنها تطبق عليه ولا عجب فانا كنت كذلك يوما ما إذن فأين الحل ؟..

مادما قريبين من الطبخ كانت أمي إذا أرادت أن تحضر أكلة (الدولمة) كانت تحضر أولا مكونات هذه الطبخة وهي البصل وورق العنب والرز وما إلى ذلك وبعد أن تنتهي من تحضير كل تلك الأدوات تبدأ بتكوين هذه الأكلة فطابت يديك يا أماه فالبرمجة مشابهة تماما لصناعة أكلة ما فعندما نسال سؤال بداية نحلل السؤال ونحضر متطلباته ثم نقوم بربط هذه المتطلبات مع بعضها ونكون برنامجا أي شيء شبيه بالخوارزمية لكل حل فعلى سبيل المثال لو جاءنا سؤال يطلب فيه جمع عددين يدخلهما المستخدم فبداية التحليل من السؤال نفهم يدخل عددين لذلك نحتاج إلى متغيرين كل متغير يمثل عدد معين من الذي سوف ندخله وبما انه قال يدخلهما المستخدم يجب أن نعمل دالة إدخال من شاشة التنفيذ إلى هذان المتغيران تم يأتي بعدهما من متطلبات السؤال جمع أي لا بد من وجود وتعريف متغير ثالث نخزن فيه نتيجة الجمع التي سوف نقوم بها ثم نطبع هذه النتيجة. الآن بعد تحليل المتطلبات نسلسل الخطوات على ورقة بشكل مشابه لكتابة البرنامج بشكل التالي

١. تعريف متغيرات (a,b,c) ٢. ندخل (a,b) من شاشة التنفيذ ٣. نجمعهم (c=a+b) ٤. نطبعهم

الآن نحول هذه المتطلبات إلى برنامج

```
#include<iostream.h>
    mian()
{
1.int a,b,c;
2.cin>>a>>b;
3.c=a+b;
4.cout<<c;}
```

كما لاحظت كيف كونا البرنامج وصحيح 100%

المحتويات

الفصل الأول : التعرف على أساسيات اللغة وطريقة كتابة أول برنامج لك

الفصل الثاني: الجمل الشرطية والعبارات الشرطية (if ,Switch)

الفصل الثالث: عبارات أو جمل التكرار (For , While , Do—While)

الفصل الرابع: المصفوفات وأنواعها (Array)

الفصل الخامس: الدوال (Function)

الفصل السادس: المؤشرات (pointer)

الفصل السابع: التراكيب (Structures)

الفصل الثامن:الملفات (File)

الفصل الأول

التعرف على أساسيات اللغة وطريقة كتابة أول برنامج لك

المستوى المطلوب

مبتدئ جداً لا يعرف أي شيء عن هذه اللغة فما فوق ذلك

الأهداف:

عندما يكتمل الفصل تكون ياذن الله قد أتممت التعرف على أساسيات هذه اللغة ومبادئها وطرق تسلسل

خطوات البرنامج

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل إتقان كامل لان بقية الفصول جميعها تعتمد بشكل مباشر على هذا الفصل

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب : أربعة ساعات

الهيكلة العامة للبرنامج :-

للبرنامج في لغة (C , C++) شكل عام عند كتابته وهو ثابت تقريبا في أجزائه الرئيسية في كل البرامج وتكون طريقة كتابته بشكل التالي

```
C++ الكود بلغة
منطقة التعريفات العامة واستدعاء مكاتب للبرنامج
# include<iostream.h>
الدالة التي تكتب بداخلها اكواد البرنامج
Main()
{
ساحة الاكواد أو منطقة كتابة اكواد البرنامج والقراءة والطباعة
}
```

```
C الكود بلغة
منطقة التعريفات العامة واستدعاء مكاتب للبرنامج
# include<stdio.h>
الدالة التي تكتب بداخلها اكواد البرنامج
Main()
{
ساحة الاكواد أو منطقة كتابة اكواد البرنامج والقراءة والطباعة
}
```

- منطقة التعريفات العامة واستدعاء مكاتب للبرنامج: في هذا المكان يتم كتابة جميع المكاتب التي سنحتاج إليها داخل البرنامج وكذلك المتغيرات التي تعرف بشكل عام لكل البرنامج والسجلات والدوال على سبيل المثال دالة القراءة والطباعة (scanf ,printf) في لغة C تقع ضمن المكتبة <stdio.h> لذلك يجب استدعاء هذه المكتبة لكي تعمل هذه الدوال ودالة (cout , cin) تقع ضمن مكتبة <iostream.h>.
- دالة (main): هذه الدالة يسلم نظام التشغيل العمل لها وعندما تنتهي وظيفتها ترجع له قيمة
- ساحة كتابة الأكواد: هي المنطقة التي يتم بداخلها كتابة الأكواد البرمجية وتعريفات وغيرها
- يجب وضع فارزة منقوطة في نهاية أي تعبير مبرمج من قبل المستخدم للدلالة على أن التعبير انتهى .

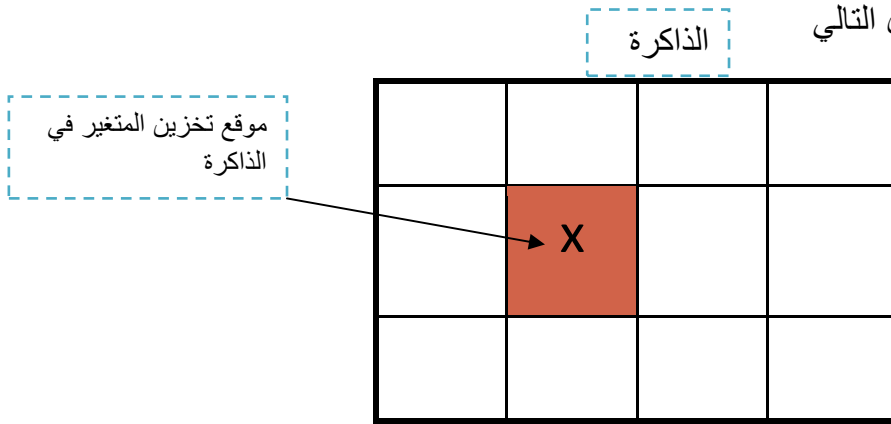
المتغيرات :-

هي مواقع في الذاكرة تخزن فيها قيم معينة قد تتغير أثناء البرنامج أو قد تبقى ثابتة. وأسماء المتغيرات تكون مفتوحة حسب الرأي الشخصي المبرمج ممكن أن يسميها إي اسم لكن يجب أن لا يكون الاسم من الأسماء المحجوزة مثل (if,for,while) أو أي أسم آخر محجوز من قبل المترجم ورغم أن تسمية المتغيرات مفتوحة لكن يفضل أن تكون تسمية المتغير دالة عليه مثلا لو كان المتغير يدل على الوقت فيفضل تسميته (time) حتى تكون اكوادك واضحة وحتى لا يكون برنامجك متشابك كمعكرونة الاسبكتي لا يعرف القارئ بدايته من نهايته.

- المتغير الذي يتم تعريفه داخل البرنامج يجب تحديد نوعه.
- ✓ فمثلا لو كان المتغير (x) يحوي قيمة متغير رقمي بدون فارزة بعد الصفر فيجب تعريفه تحت الدالة main() بأنه متغير من نوع (integer) هكذا

البرمجة بلغة c++	البرمجة بلغة c
<pre># include<iostream.h> Main() { // هنا تعرف المتغيرات int x; }</pre>	<pre># include<stdio.h> Main() { // هنا تعرف المتغيرات int x; }</pre>

ويخزن المتغير (x) في الذاكرة بشكل التالي



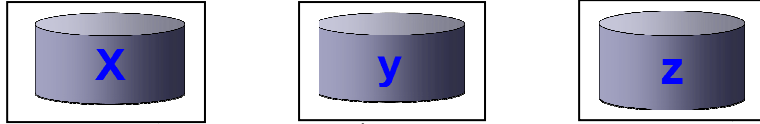
- كل موقع في الذاكرة يكون مرقم برقم معين يختلف عن غيره من المواقع

✓ هناك عدة أنواع من المتغيرات يمكن تعريف المتغيرات بها وهي

تعريف المتغير	استخدامه
Int var;	يستخدم لتعريف المتغير من نوع integer أي رقمي مثلا (int x=5)
Float var;	يستخدم لتعريف المتغير من نوع كسري مثلا (Float var=5.4;)
Char var;	يستخدم لتعريف المتغير من نوع حرفي مثلا (Char var="a";)
Double var;	يستخدم لتعريف المتغير من نوع Double أي حقيقي
Void var;	يستخدم لتعريف المتغير من نوع Void أي لا يرجع أي قيمة

✓ يمكن تعريف أكثر من متغير في سطر واحد بوضع فارزة بينهم

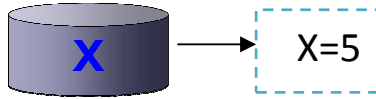
```
الكود
char x,y,z;
```



✗ المتغيرات (x,y,z) هي متغيرات تستطيع أن تخزن في داخلها حرف

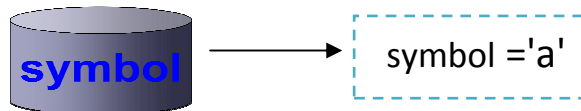
✓ يمكن أن يسند المتغير بقيمة مباشرة إثناء وقت التعريف هكذا

```
الكود
int x=5;
```



✓ قد يسند المتغير بقيمة معينة بعد التعريف في أي مكان في البرنامج .

```
الكود
char symbol;
symbol = 'a';
```



✗ المتغير (symbol) تم تخزين في داخله الحرف (a) .
• الحروف عندما تخزن تضع بين علامة تنصيفية واحد

✓ قد نسند قيمة متغير إلى متغير آخر

```
الكود
int first ,second;
first =51;
second= first ;
```

✗ أصبح المتغير second يحوي نفس قيمة المتغير first في الخطوة رقم (٣)

✓ أو قد يسند التغير بقيمة في وقت الإدخال من لوحة المفاتيح (سنتطرق عليه لاحقا)

إسناد قيم للمتغيرات: تكون طريقة إسناد إي قيمة أو تعبير رياضي إلى متغير بشكل التالي

متغير أو تعبير رياضي=اسم المتغير

- في الطرف الأيمن من المساواة يوجد فقط اسم المتغير الذي نريد إسناد قيم إليه
- في الطرف الأيسر من المساواة نستطيع كتابة إي تعبير رياضي أو متغير أو قيمة معينة

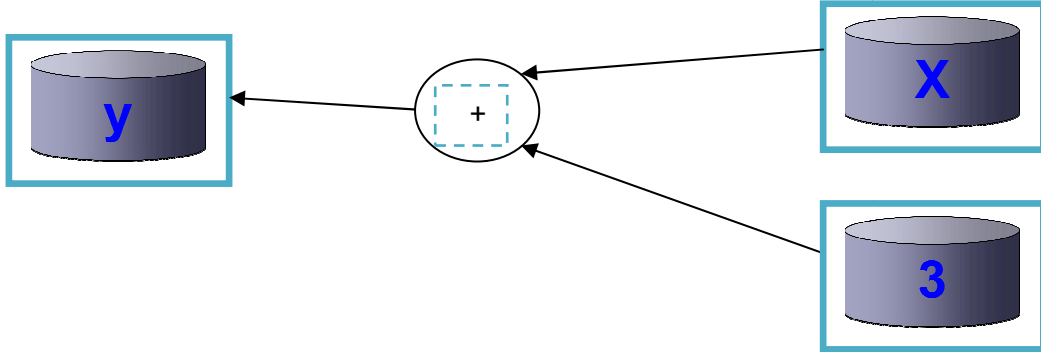
مثال : إسناد قيمة تعبير رياضي إلى متغير (بطريقة صحيحة)

الكود

```
1.int y, x=5;  
2.y=x+3;
```



- في الخطوة رقم (٢) أسندنا القيمة الناتجة من تعبير رياضي وهو (x+3) إلى المتغير (y)



- لو تلاحظ أن الطرف الأيمن مكون من متغير فقط والطرف الأيسر مكون من تعبير رياضي

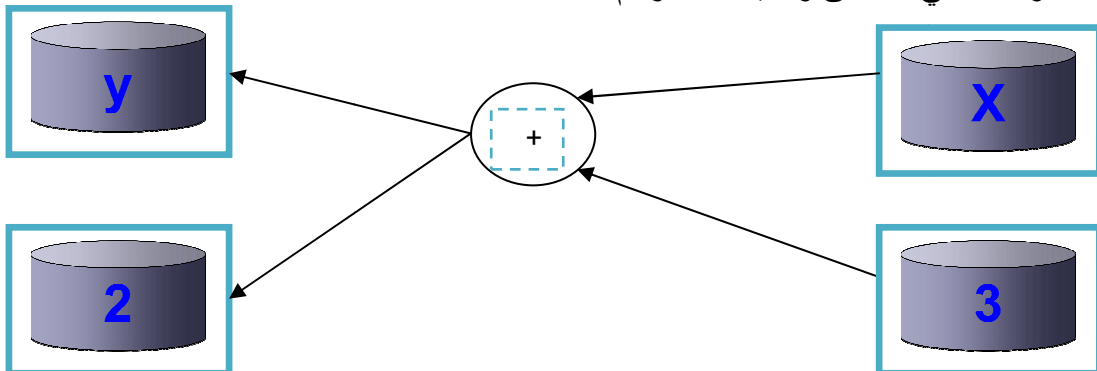
مثال : إسناد قيمة تعبير رياضي إلى متغير (بطريقة خاطئة)

الكود

```
1.int y, x=5;  
2.y+2=x+3;
```



- في الخطوة رقم (٢) أسندنا القيمة الناتجة من تعبير رياضي وهو (x+3) إلى تعبير رياضي آخر وهذه الشيء خاطئ ولا يقبله المترجم



المتغيرات الثابتة: هي متغيرات تبقى قيمتها ثابتة أثناء تنفيذ البرنامج ولا تتغير أبدا وتعرف بوضع كلمة **(const)** قبل تعريف نوع المتغير. وفائدتها نستخدمها للمتغيرات التي لا نريد أن تتغير قيمتها أثناء عمل البرنامج أبدا. مثلا قيمة (pi=3.14) هذه قيمة رياضية ولا تتغير ابد مهما حدث ودائما نحتاجها في البرامج الرياضية لذلك نعرفها **(const)** ونعطيها قيمة (3.14) سنضمن لأنفسنا أنها لا تتغير مهما حدث وأينما نحتاجها نكتب فقط (pi)

الكود

```
const float pi=3.14;
```



احد أكثر الأسئلة أهمية وهي كيفية تنفيذ البرنامج في لغات البرمجة ؟

هذا السؤال قاتل كسيف إذا لم تفهمه لن تفهم شيء من لاقهيه. ويبدأ تنفيذ البرنامج من الدالة main() ينفذ البرنامج سطر ثم ينتقل إلى السطر الذي يليه (لاحظ تسلسل الترقيم في المثال) ويستمر هكذا حتى يصل إلى نهاية البرنامج وأي مكتبة يحتاجها أو أي دالة يحتاجها يتجه ليبحث عنها خرج هذه الدالة

كيفية تسلسل تنفيذ خطوات البرنامج

```
# include<iostream.h> //or #include <stdio.h> for user of C language
```

```
Change_position()
```

2

```
{
6.
7.
8.
```

عندما انتهى من تنفيذ ما موجود في الدالة يعود إلى البرنامج الرئيسي جاعلا خطوة التنفيذ التالية بعد هي (٨) وهي (٩) ويستمر البرنامج

```
}
```

من هنا يبدأ تنفيذ البرنامج خطوة بخطوة (هذه أول خطوة)

```
Main()
```

1

```
{
1.
2.
3.
4.
```

لاحظ في السطر الخامس احتاج دالة تقع خارج Main نقل تنفيذ البرنامج لها أصبح الخطوة (٦) عندها

```
5.Change_position()
```

3

```
9.
10
11.
}
```



ماذا يحدث لو ساوينا متغير من نوع integer بأخر من نوع float كل الذي يحصل هو أن المتغير integer سوف يأخذ الرقم فقط قبل الصفر وبهمل الذي بعده

مثال: برنامج لتحميل قيمة متغير من نوع integer إلى متغير من نوع float

البرمجة بلغة c++	البرمجة بلغة c
<pre>#include<iostream.h> main() { int fixOnly; float fixAndPint=3.5; fixOnly =fixAndPint; cout<<fixOnly; }</pre>	<pre>#include<stdio.h> main() { int fixOnly; float fixAndPint=3.5; fixOnly =fixAndPint; printf("%d",fixOnly); }</pre>

توضيح الخطوات :

١. في السطر الأول استدعينا مكتبة التعاريف التي تخص اللغة

٢. في السطر الثاني دالة Main التي وصفناها سابقا وفي السطر الثالث فتحنا قوس بداية البرنامج

٣. في السطر الرابع عرفنا متغير fixOnly من نوع integer

٤. في السطر الخامس عرفنا متغير float fixAndPint من نوع float وأسندنا له قيمة ٣.٥

٥. في السطر السادس ساوينا المتغير fixOnly من نوع integer بالمتغير float fixAndPint من نوع float فأصبح المتغير fixOnly من نوع integer يحمل فقط قيمة الجزء الصحيح للمتغير float fixAndPint من نوع float

٦. في السطر السابع طبعنا قيمة المتغير fixOnly والسطر الثامن أغلقنا البرنامج

✓ نلاحظ إن المتغير fixOnly سوف تصبح قيمته (٣) فقط رغم الذي ساوينا فيه هو (٣.٥) والذي سيظهر في شاشة التنفيذ هو (٣)

✓ نلاحظ وجود (cout,printf) هذه الدوال تستخدم لعرض الناتج على شاشة التنفيذ للمستخدم وتسمى دوال الإخراج وهذه ما سنتناوله الآن.

✓ في دالة الطباعة في لغة C استخدمنا الرمز "%d" لأن ما سنطبعه هو متغير من نوع integer وهذه ما ستفهمه من الجدول رقم ١.

دوال الإخراج:

هي دوال تستخدم لعرض نواتج العمليات أمام المستخدم في شاشة التنفيذ والدوال هي.

C++ الدالة المستخدمة في لغة

```
cout<<var;
```

C الدالة المستخدمة في لغة

```
printf("%symbbleToVar",var);
```

- أي لغة لها دوال خاصة في الإدخال وهو الاختلاف الأكثر شيوعا بين هاتين اللغتين هي ودوال الإدخال أما بقية الدوال فتنشابه كثيرا جدا في ما بينها.
- **(var)** هو اسم المتغير الذي سوف نقوم بطباعة قيمته
- **(symbbleToVar)** هي رموز تستخدم للدلالة على نوع المتغير المراد طباعته وهذا جدول بالرموز

الرمز	وصفه (هذا الجدول فقط موجود بلغة C) جدول (١)
<code>printf ("%d",&var);</code>	المتغير var عبارة عن متغير رقمي
<code>printf ("%f",&var);</code>	المتغير var عبارة عن متغير كسري
<code>printf ("%c",&var);</code>	المتغير var عبارة عن حرف
<code>printf ("%s",&var);</code>	المتغير var عبارة عن سلسلة

☒ لتنفيذ البرنامج نضغط (ctrl+f9) بعد كتابة الكود. لنرى النتائج

مثال: لو أردنا طباعة قيمة المتغير (data3) وهو من نوع integer

C++ البرمجة بلغة

```
cout<< data3;
```

C البرمجة بلغة

```
Printf("%d", data3);
```

☒ استخدمنا الرمز ("%d") في لغة (C) لان المتغير الذي سنطبع قيمته من نوع integer

كتابة برنامجك الأول:

مثال : لو أردنا طباعة (hi hussien ahammed taleb) أمام المستخدم فيكون الكود كالتالي

C++	البرمجة بلغة	C	البرمجة بلغة
<pre>#include<iostream.h> main() { cout<<" hi hussien ahammed taleb"; }</pre>		<pre>#include<stdio.h> main() { printf(" hi hussien ahammed taleb"); }</pre>	

الآن لننفذ البرنامج بالضغط على (ctrl+f9) أو (run) من القائمة سيظهر الشكل التالي في شاشة التنفيذ

```
(Inactive C:\TCWIN45\BIN\NONAME0:
hi hussien ahammed taleb
```



نلاحظ: أي جملة نصية يجب وضعها بين علامتي تنصيص عند طباعتها في دوال الطباعة.

- لو أردنا وضع كل كلمة في سطر فقط نستخدم القالب "\n" بين كل كلمة حيث يقوم هذه القالب بإنزال المؤشر في شاشة التنفيذ إلى السطر التالي ولذلك ما يتم طباعته بعده يطبع في السطر التالي الذي عليه المؤشر

البرمجة بلغة	c	البرمجة بلغة	C++
	#include<stdio.h> main() { Printf(" hi\nhussien\nahammed\ntaleb"); }		#include<iostream.h> main() { cout<<" hi\nhussien\nahammed\ntaleb"; }

الآن لننفذ البرنامج سيظهر الشكل التالي في شاشة التنفيذ

```
(Inactive C:\TCWIN45\B
hi
hussien
ahammed
taleb
```

لو تلاحظ كل ("\n") يقوم بإنزال مؤشر الطباعة إلى سطر جديد ليظهر ما بعده في السطر الجديد

✓ هناك بعض الرموز التي تستخدم في دوال الطباعة لترتيب شاشة الطباعة أمام المستخدم بطرق مختلفة فمنها من يضيف فراغات ومنها من يرتب عمودي وغيرها وهذا جدول بتلك الرموز

بعض العلامات المهمة في الطباعة وترتيب الشاشة أمام المستخدم (هذه العلامات مهمة في الطباعة)

الرمز	وظيفته
v	ترتيب عمودي
h	ترتيب أفقي
n	أنزال المؤشر إلى سطر جديد
t	وضع أربع فراغات خلف العنصر التي تمت طباعته حالياً

في نفس المثال السابق لو أردنا أن يطبع بين كل كلمة وأخرى أربع فراغات فقط نبديل "\n" ب "\t" في دالة الطباعة

البرمجة بلغة	c	البرمجة بلغة	C++
	Printf(" hi\t hussien\t ahammed\t taleb");		cout<<" hi\t hussien\t ahammed\t taleb";

مثال: إذا أردنا طباعة قيمة متغير تم إسناد قيمة إليه وقت تعريفه .

البرمجة بلغة	c	البرمجة بلغة	c++
	<pre>#include<stdio.h> Main() { float fixAndPint=3.5; printf("%f", fixAndPint) ; }</pre>		<pre>#include<iostream.h> Main() { float fixAndPint=3.5; cout<< fixAndPint ; }</pre>

١. بما أن المتغير من نوع (float) لاحظنا وجود "%f" في لغة (C) دلالة على أن المتغير كسري الذي سوف يتم طباعته

٢. الناتج في شاشة التنفيذ لهذا البرنامج يكون فقط (3.5) لاشيء آخر

• لو أردنا أن يعرض في شاشة التنفيذ هكذا

```
Number_is=3.5 $
```

في لغة C++ الموضوع سهل فقط نطبع (=Number_is) بشكل سلسلة قبل المتغير (fixAndPint) ونطبع (\$) بعد المتغير أيضا بشكل سلسلة وبهذا يصبح الكود هكذا

الكود بلغة	C++
	<pre>#include<iostream.h> Main() {float fixAndPint=3.5; cout<<" Number_is="<< fixAndPint<<"\$" ;}</pre>



إما في لغة (C) فإن إي سلسلة قبل ("%symbleToVar") الخاص بالمتغير يطبع قبل المتغير في شاشة التنفيذ وأي سلسلة بعده تطبع بعد المتغير

الدالة المستخدمة في لغة	C
	<pre>printf("%symbleToVar",var);</pre>

ونعلم انه كل متغير عند طباعته له (symbleToVar) خاص به حسب نوعه. (وبهذه يصبح الكود بلغة (C) هكذا)

الكود بلغة	C
	<pre>#include<stdio.h> Main() {float fixAndPint=3.5; printf(" Number_is=%f\$", fixAndPint) ;}</pre>

دوال الإدخال :

تستخدم دوال الإدخال لإدخال معلومات من قبل المستخدم من شاشة التنفيذ وإسناد قيم إلى المتغيرات المعرفة داخل البرنامج من خلال إدخال المستخدم لقيمتها لغرض معالجتها القيام بالعمليات المطلوبة.

الكود بلغة	C++
	<code>Cin>>var;</code>

الكود بلغة	C
	<code>scanf("%symbbleToVar",&var);</code>

- **(var)** هو قيمة المتغير الذي سوف يقوم المستخدم بادخاله.
- **(symbbleToVar)** هي رموز تستخدم للدلالة على نوع المتغير المدخل وهذا جدول بالرموز.

الرمز	وصفه (هذا الجدول فقط موجود بلغة C) جدول (٢)
<code>scanf("%d",&var);</code>	المتغير var عبارة عن متغير رقمي
<code>scanf("%f",&var);</code>	المتغير var عبارة عن متغير كسري
<code>scanf("%c",&var);</code>	المتغير var عبارة عن حرف
<code>scanf("%s",&var);</code>	المتغير var عبارة عن سلسلة

نستطيع إدخال أكثر من متغير في دالة إدخال واحدة

الكود بلغة	C++
	<code>cin>>var1>>var2;</code>

الكود بلغة	C
	<code>scanf("%symbbleToVar %symbbleToVar",&var1,&var2);</code>

- كما مبين بالرسم المؤشر بالأسهم كل متغير وله (%symbbleToVar) خاص به



مثال: إذا كان لدينا المتغير (x) من نوع (float) وأردنا إدخال قيمه له من شاشة التنفيذ سيكون الكود بشكل التالي

الكود بلغة	C++
	<code>cin>> x;</code>

الكود بلغة	C++
	<code>scanf("%f",& x);</code>

☒ استخدمنا الرمز ("%f") في لغة (C) لان المتغير الذي سنطبع قيمته من نوع (float)

مثال: لو أردنا أن يقوم المستخدم بإدخال حرف ويطبع الحرف أمامه بين قوسين إي لو ادخل a سوف يطبع في شاشة التنفيذ (a)

البرمجة بلغة c++	البرمجة بلغة c
<pre>#include<iostream.h> main() { 1.char enterchar; 2.cin>> enterchar; 3.cout<<" ("<< enterchar <<")"; }</pre>	<pre>#include<stdio.h> main() { 1.char enterchar; 2 scanf("%c",& enterchar); 3.printf(" (%c)", enterchar) ; }</pre>

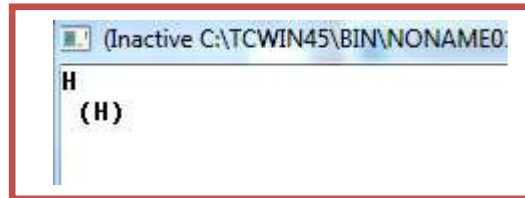
توضيح الخطوات:

١. خطوة رقم (١) عرفنا متغير من نوع حرفي

٢. خطوة رقم (٢) قمنا بإدخال قيمة للمتغير من شاشة التنفيذ ويدخل فقط حرف واحد. بما انه المتغير من نوع character في لغة (C) يقرأ ويطبع بدلالة "%C"

٣. خطوة رقم (٣) قمنا بطباعة قيمة المتغير التي أدخلت في خطوة رقم (٢) بين قوسين

ما سيظهر في شاشة التنفيذ هو



أبدال بين قيم متغيرين: لإبدال بين قيمة متغيرين نحتاج إلى متغير ثالث من نفس نوع المتغيرين حتى نخزن به نتيجة احد المتغيرين ثم نبدل لان في حال عدم وجود متغير ثالث لا نستطيع الإبدال ستضيع احد القيمتين

مثال : أبدال بين قيمة المتغير (a) والمتغير (b)

```

كود الإبدال بين قيمة المتغير (a) والمتغير (b)
int a=12,b=30,c; //a=12 , b=30
c=a; // نخزن قيمة المتغير a في متغير ثالث (c)
a=b; // نضع قيمة المتغير b في المتغير a
b=c; // نضع القيمة المخزنة في المتغير الثالث وهي قيمة a في المتغير b

```

العمليات الحسابية وطرق تمثيلها وأولوياتها (الأسبقية):
العمليات الرياضية:-

الأسبقية	الوظيفة	الرمز
١	الزيادة والنقصان	++ أو --
٢	الإشارة السالبة	-
٣	الضرب والقسمة وباقيها	* أو / أو %
٤	الجمع أو الطرح	+ أو -
٥	التساوي	=
٦	زيادة أو نقصان للعدد	++ أو - المتأخرة بعد الرمز

الأدوات Bowties:-

الأسبقية	الوظيفة	الرمز
١	إشارة النفي	~
٢	إزاحة للعدد يمين أو يسار	<< أو >>
٣	عملية (and)	&
٤	الرفع لقوى	^
٥	عملية (or)	

الأدوات المنطقية تستخدم في العبارات الشرطية

الأسبقية	الوظيفة	الرمز
١	النفي	!
٢	عملية منطقية (and)	&&
٣	عملية منطقية (or)	

✓ تكون نواتج الأدوات المنطقية ومقارنتها إما (True) أو (False) أي إما واحد أو صفر. إذا كان احد طرفي المقارنة رقم غير الصفر مثلا سبعة فأى رقم غير الصفر يعتبره واحد

☒ تستخدم الأدوات المنطقية كشرط مقارنة في العبارات الشرطية هي وأدوات (Bowties)

☒ في الأسبقيات الرقم الأقل أعلى أسبقية مثلا أسبقية الضرب أعلى من أسبقية الجمع لأن رقم الضرب في جدول الأسبقية هو (٣) و رقم الجمع هو (٤)

تمثيل العمليات الرياضية:

تمثل العمليات الرياضية برمجيا بطريقة مشابهة لطريقة تمثيلها رياضيا مع تغير طفيف بالرموز الرياضية لما يكافئها من الرموز البرمجية ولاحظ هذا الجدول التوضيحي للعمليات وتمثيلها رياضيا وبرمجيا

المثال على أساس عندنا متغيران هما (a,b) وناتج العملية الرياضية يخزن في (c)

الرمز و الوظيفة	تمثيله رياضيا	تمثيله برمجيا
الجمع (+)	$C=a+b$	$C=a+b;$
الطرح (-)	$C=a-b$	$C=a-b;$
القسمة (/)	$C=\frac{a}{b}$	$C=a/b;$
الضرب (*)	$C=a*b$	$C=a*b;$
باقي القسمة (%)	$C=a \text{ mod } b$	$C=a\%b;$

⊠ لاحظ أن التمثيل البرمجي مشابه تقريبا للتمثيل الرياضي مع أبدال بعض الرموز بما يكافئها ووضع فارزة منقوطة في نهاية التعبير .

مثال: لو كان لدينا متغيرين من نوع (integer) وكلاهما يحمل قيمة معينة يدخل قيمهم المستخدم من شاشة التنفيذ قم بعملية جمع لهما.

البرمجة بلغة c++	c	البرمجة بلغة
1.#include<iostream.h>	1.#include<stdio.h>	
2.main()	2.main()	
3.{ int first, seconds, sum;	3.{ int first, seconds, sum;	
4.cin>> first;	4 scanf("%d", &first);	
5. cout<<"\n";	5. printf("\n");	
6. cin>> seconds;	6. scanf("%d", & seconds);	
7.sum= first+ seconds;	7.sum= first+ seconds;	
8.cout<<"-----\nsum="<< sum ;	8.printf("-----\nsum=%d", sum);	
9. }	9. }	

توضيح الخطوات:

١. خطوة رقم (١) هي تعريف للمكتبة الخاصة بدوال الإدخال والإخراج. خطوة رقم (٢) هي دالة main()

٢. خطوة رقم (٣) فتحنا قوس بداية البرنامج. عرفنا المتغير الأول. و عرفنا المتغير الثاني و عرفنا متغير الجمع

٣. خطوة رقم (٤) قرئنا قيمة المتغير الأول من شاشة التنفيذ. وخطوة رقم (٥) طبعنا علامة الجمع على شاشة التنفيذ لزيادة جمالية البرنامج. خطوة رقم (٦) قرئنا قيمة المتغير الثاني من شاشة التنفيذ

٤. خطوة رقم (٧) قمنا بجمع المتغير (first) والمتغير (seconds) بداخل متغير آخر اسمه (Sum) وطبعنا قيمة هذه المتغير في خطوة رقم (٨) وعرفنا المتغير (Sum) من نوع (integer) وهو على أساس جمع عددين من نوع (integer) يكون الناتج من نوع (integer) ولو كان احد العددين غير (integer) لكان يجب تعريف Sum بطريقة بلانم كي يحمل نتيجة جمع هذان العددين إي لو كان احدهما كسري لكان يجب تعريفه من نوع (float)

٥. خطوة رقم (٩) قمنا بإغلاق قوس البرنامج

```
(Inactive C:\TCWIN45\BIN\N...
30
+
25
-----
sum=55
```

ما سيظهر في شاشة التنفيذ هو. ←

- ☒ ونستطيع إجراء بقية العمليات الرياضية على المتغيرين بنفس الطريقة فقط نبدل إشارة الجمع في الخطوة رقم (٧) إلى إشارة ضرب أو طرح أو قسمة ؟ إي ان
- ✓ لو أردنا إجراء عملية طرح للرقمين المدخلين فقط نغير خطوة رقم (٧) إلى

كود

```
7.sum= first- seconds;
```

مع تغير رمز الجمع في خطوة رقم (٥) إلى رمز الطرح

- ✓ لو أردنا إجراء عملية ضرب للرقمين المدخلين فقط نغير خطوة رقم (٧) إلى

كود

```
7.sum= first* seconds;
```

مع تغير رمز الجمع في خطوة رقم (٥) إلى رمز الضرب

- ✓ لو أردنا إجراء عملية قسمة للرقمين المدخلين فقط نغير خطوة رقم (٧) إلى

كود

```
7.sum= first/ seconds;
```

مع تغير رمز الجمع في خطوة رقم (٥) إلى رمز القسمة

- ☒ في عملية القسمة يتم تعريف المتغير (div) بدل المتغير (sum) على انه متغير كسري لان عملية قسمة إي عددين قد ينتج عنها رقم بعد الفارزة

كما لاحظنا أن العمليات الرياضية برمجا تمثل نفسها في الطريقة الاعتيادية لكن برموز أخرى دالة عن نوع العملية في بعض الحالات

أيجاد باقي القسمة: باقي القسمة هو ما تبقى من قسمة عدديين على سبيل المثال باقي قسمة الإعداد (يستخدم الرمز % للدلالة على باقي القسمة) .

باقي قسمة الإعداد التالية

$$3\%2=1$$

$$6\%2=0$$

$$9\%3=0$$

$$10\%8=2$$



لاحظ أن إذا كان الرقم الأول اقل من الثاني فلا يتقسم عليه أصلا كله يبقى كباقي قسمة

$$9\%12=9$$

$$7\%8=7$$

✓ لو أردنا إيجاد باقي قسمة للرقمين المدخلين في المثال السابق فقط نغير خطوة رقم (٧) إلى

كود

```
7.sum= first % seconds;
```

الرموز العلائقية: هي رموز تستخدم لمعرفة العلاقة بين الرقمين إي هل يساويه أو أكبر منه أو لا يساوي أو اصغر منه وهذا جدول بهذه الرموز وتكون نتيجة المقارنة إما (True) او (False)

الرمز	الوظيفة	مثال
>	علامة اكبر	(a>b)
>=	علامة اكبر أو يساوي	(a>=b)
<	علامة أصغر	(a<b)
<=	علامة أصغر أو يساوي	(a<=b)
==	علامة اليساوي	(a==b)
!=	علامة لا يساوي	(a!=b)



الأسبقيات وطرق معاملتها:

في جداول الرموز وضعنا أمام كل رمز الأسبقية الخاصة؟ على سبيل المثال إذا جاءت عملية ضرب وجمع في تعبير واحد فيكون للضرب أسبقية على الجمع (لأن أسبقيته الضرب هي (٣) وأسبقية الجمع هي (٤) إي الضرب أعلى أسبقية. الرقم الأقل أعلى أسبقية) وينفذ قبله وهذا هو من أصول عمل المترجم لذلك يجب فهم الأسبقيات حتى لا تخطئ في طريقة تحليل التعبير الرياضي لأي مسألة . شاهد تغلب الأسبقيات في المثال التالي هو $(a-b/d)$ ؟

مثال

يبدأ المترجم تنفيذ العمليات من اليسار إلى اليمين

$$\text{Reslt}=a - c / d$$

يبدأ بمقارنة كل عمليتين رياضيتين معا وأيها له أسبقية أعلى تنفذ أولاً

إذا كان $(a=5,c=10,d=2)$ فيكون تسلسل تنفيذ الخطوات

1.

$$\text{Reslt}=5 - 10 / 2$$

عند مقارنة عملية القسمة و الطرح فوجد انه القسمة له أسبقية أعلى من الطرح لذلك ستنفذ القسمة أولاً فيقسم $(10/2=5)$

2.

$$\text{Reslt}=5 - 5$$

تنفذ عملية الطرح بشكل اعتيادي لأنها آخر عملية رياضية متبقية قبل المساواة ويكون الناتج هو صفر وأخر عملية ستنفذ هي المساواة فتصبح قيمة

$$(\text{Reslt}=0)$$

لو لاحظنا كيف تغلبت عملية القسمة على عملية الطرح في المثال وربما نحن كنا نريد أن تنفذ عملية الطرح أولاً لكن المترجم نفذ حسب الاسبقية لذلك يجب مراعاة التعبير والاسبقيات.

ملاحظة مهمة: عند مقارنة عمليتان ويوجد أن الاسبقيتان متساويتان سننفذ من اليسار إلى اليمين أول عملية تقع في اليسار تنفذ أولاً

$$\text{Reslt}=a * c + d$$

في هذا المثال نحن نقصد في تعبيرنا أن (c,d) يجمعون أولاً ثم تضرب النتيجة في (a) لكن الواقع غير ذلك حسب الأسبقيات أن الضرب له أسبقية على الجمع لذلك سوف يضرب (a,c) وتجمع النتيجة مع (d) ويكون الناتج غير صائب.....! لاحظ تسلسل العمليات الناتجة عن هذا التعبير

يبدأ تنفيذ العمليات من اليسار إلى اليمين

$$\text{Reslt}=a * c + d$$

يبدأ بمقارنة كل عمليتين رياضيتين معا وأيها له أسبقية أعلى تنفذ أولاً

إذا كان (a=5,c=10,d=5) فيكون تسلسل تنفيذ الخطوات

1.

$$\text{Reslt}=5 * 10 + 5$$

عند مقارنة عملية الجمع والضرب فوجد انه الضرب له أسبقية أعلى من الجمع لذلك ستنفذ الضرب أولاً فيضرب
(5*10=50)

2.

$$\text{Reslt}=50 + 5$$

تنفذ عملية الجمع بشكل اعتيادي لأنها آخر عملية رياضية متبقية قبل المساواة ويكون الناتج هو (55) وآخر عملية ستنفذ هي المساواة فتصبح قيمة

$$(\text{Reslt}=55)$$

مهم

رياضياً لحل هذه المشكلة نضع أقواس حول العمليات التي لها أسبقية أقل ونريدها أن تنفذ أولاً وهو بالضبط ما نعمله برمجياً أيضاً نستخدم أقواس (لأن الأقواس لها أسبقية على جميع باقي العمليات) لذلك سوف ينفذ ما في داخلها تم يتعامل مع الخارج القوس وتكون النتيجة صائبة؟
ويصبح شكل المثال السابق

$$\text{Reslt}=a *(c + d)$$

ويكتب برمجيا هكذا

Reslt=a *(c +d);

لاحظ تسلسل العمليات الرياضية الآن

يبدأ تنفيذ العمليات من اليسار إلى اليمين

Reslt=a * (c + d)

يبدأ بمقارنة كل عمليتين رياضيتين معا وأيها له أسبقية أعلى تنفذ أولاً

إذا كان (a=5,c=10,d=5) فيكون تسلسل تنفيذ الخطوات

1. Reslt=5 * (10 + 5)

عند مقارنة عملية الضرب والأقواس فوجد انه الأقواس له أسبقية أعلى من الضرب لذلك ستنفذ ما بين الأقواس أولاً فيجمع (10*2=12)

2. Reslt=5 * 15

تنفذ عملية الضرب بشكل اعتيادي لأنها آخر عملية رياضية متبقية قبل المساواة ويكون الناتج هو (75) وأخر عملية ستنفذ هي المساواة فتصبح قيمة

Reslt=75

هل لاحظت كم هناك اختلاف بين النتيجتين.....؟

ملاحظة: إذا كان ما بين الأقواس أكثر من عملية رياضية ايظا داخل الأقواس تعمل الأسبقيات فأيهما له اسبقية أعلى ينفذ أولاً على سبيل المثال لو كان المثال

Reslt=a * (c + d/f)

تسلسل تنفيذ العمليات هي

١. يقسم (d/f)

٢. ويجمع ناتج القسمة مع (c)

٣. ويضرب ناتج مع (a)

٤. يساوي النتيجة ب reslt

$$y = \frac{5 + A}{D} - \frac{B}{C}$$

ونحن نريد أن نقصد في تعبيرنا أن (5,A) يجمعون أولاً ثم تقسم النتيجة في (d) لكن الواقع غير حسب الأسبقيات أن القسمة لها أسبقية على الجمع لذلك سوف القسمة إي يقسم (5/D) وتجمع النتيجة مع (A) ويكون الناتج غير صائب.....!

لاحظ تسلسل العمليات بدون أقواس وحسب التعبير (Y= 5+A /D - B/C)

$$y=5+A/d-b/c$$

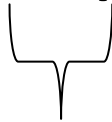


يبدأ بمقارنة كل عمليتين رياضيتين معا وأيها له أسبقية أعلى تنفذ أولاً

إذا كان (c=2,A=4,b=4,d=4) فيكون تسلسل تنفيذ الخطوات

1.

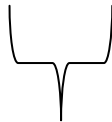
$$y=5 + 4 / 4 - 4 / 2$$



عند مقارنة عملية الجمع والقسمة فوجد انه القسمة له أسبقية أعلى من الجمع لذلك سنقسم أولاً فيقسم (4/4=1)

2.

$$y=5 + 1 - 4 / 2$$



عند مقارنة عملية الجمع مع الطرح وجد أن الاسبقيتان متساويتان لذلك سننفذ من اليسار إلى اليمين ومن اليسار أول عملية تقع هي الجمع لذلك سينفذ الجمع أولاً (5+1=6)

3.

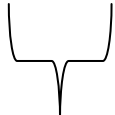
$$y=6 - 4 / 2$$



عند مقارنة عملية القسمة مع الطرح وجد أن الأسبقية القسمة أعلى لذلك سننفذ القسمة أولاً (4/2=2)

4.

$$y=6 - 2$$



تنفذ عملية الطرح بشكل اعتيادي لأنها آخر عملية رياضية متبقية قبل المساواة ويكون الناتج هو (4) وآخر عملية ستنفذ هي المساواة فتصبح قيمة

$$y=4$$

رياضيا لحل هذه المشكلة نضع أقواس حول العمليات التي لها أسبقية اقل ونريدها أن تنفذ أولا وهو بالضبط ما نعمله برمجيا أيضا نستخدم أقواس حول عملية الجمع لذلك سوف ينفذ الجمع تم يقسم النتيجة على (D) وبما أم الناتج مطروح من عملية قسمة سوف ينفذ القسمة B على C وي طرح النتيجة وتكون النتيجة صائبة؟

ويكتب برمجيا هكذا

$$Y= (5+A) / D - B / C ;$$

لاحظ تسلسل العمليات بوجود أقواس وحسب التعبير

$$y=(5+A)/d-b/c$$



يبدأ بمقارنة كل عمليتين رياضيتين معا وأيها له أسبقية أعلى تنفذ أولاً

إذا كان (c=2,A=4,b=4,d=4) فيكون تسلسل تنفيذ الخطوات

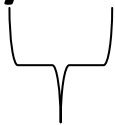
1.

$$y=(5 + 4) / 4 - 4 / 2$$



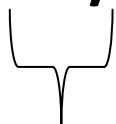
عند مقارنة الأقواس والقسمة فوجد انه الأقواس له أسبقية أعلى من الجمع لذلك سينفذ ما بين الأقواس أولاً أولاً فيجمع (5+4=9)

2. $y = 9/4 - 4 / 2$



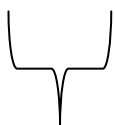
عند مقارنة عملية القسمة مع الطرح وجد أن القسمة أعلى فستنفذ أولاً (9/4=2.25)

3. $y = 2.25 - 4 / 2$



عند مقارنة عملية القسمة مع الطرح وجد أن الأسبقية القسمة أعلى لذلك سننفذ القسمة أولاً (4/2=2)

4. $y = 2.25 - 2$



الآن تنفذ عملية الطرح بشكل اعتيادي والنتيجة تصبح

$$y = 25$$

الآن الأسبقيات أصبحت واضحة ولمعلوماتك..؟

- ✓ ١ يبدأ تنفيذ العمليات من اليسار إلى اليمين
- ✓ ٢ يقارن كل عمليتين معا العملية التي لها أسبقية أعلى تنفذ أولاً
- ✓ ٣ إذا تساوت أسبقيتان يبدأ بالتنفيذ من اليسار إلى اليمين
- ✓ ١ مابين الأقواس ينفذ أولاً لأنه أعلى أسبقية من غيره (وما في داخل مابين الأقواس إي إذا كان أكثر من عملية بين الأقواس تعامل حسب الأسبقية الذي أسبقته أعلى ينفذ أولاً)

⊗ ملاحظة: الأسبقيات بالنسبة للأدوات Bowties والأدوات المنطقية هي نفس طريقة في العمليات الرياضية أيضا الذي له أسبقية أعلى ينفذ أولا

مثال

True && False || True

جد النتيجة.....؟

1. True && False || True

نعم انه and له أسبقية أعلى من or لذلك سينفذه أولا (True And False=False)

False || True

2. الآن عملية or بشكل اعتيادي (False or True=True) فالنتيجة تكون (True)

نفس الطريقة بالنسبة لباقي الأدوات تنفذ حسب الأسبقية

جدول الحقيقة ل (AND, OR, NOT)

A	B	AND	OR	NOT فقط ل (A)
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False



✓ في عمليات المقارنة إي رقم غير الصفر يعتبره واحد سوا كان موجب او سالب اذ ان تكون المقارنة

$(3+2) \&\& (4*2) || (4\%2)$

جد النتيجة.....؟

 $(3+2) \&\& (4*2) || (4\%2)$

1.

بما انه موجود تعبير رياضية وعمليات منطقية سينفذ العمليات الرياضية التي بين الأقواس أولا أول عملية سينفذها هي الضرب لأن أسبقيتها أعلى من الجمع وباقي القسمة هي $(4*2)$ وتساوي ثمانية

 $(3+2) \&\& 8 || (4\%2)$

2.

سينفذ باقي القسمة لأن أسبقيته أعلى من الجمع وباقي قسمة $(4\%2)$ هو صفر

 $(3+2) \&\& 8 || 0$

3.

سينفذ عملية الجمع $(2+3)$ يساوي خمسة

 $5 \&\& 8 || 0$

4.

الآن يقارن عملية (and) وعملية (or) نعلم انه and له أسبقية أعلى من or لذلك سينفذ أولا في عمليات المقارنة إي رقم غير الصفر يعتبره واحد سوا كان موجب أو سالب إذن تكون المقارنة بشكل التالي

 $(5 \&\& 8)$ $(1 \&\& 1=1)$

النتيجة هي واحد

 $1 || 0$

5.

آخر عملية مقارنة هي (or) بين الواحد والصفر والنتيجة هي واحد
 $(1 || 0)=1$

مهم

ما الفرق بين && و & أو || و |

الفرق يقع بين && و & : انه (&&) يمثل عملية منطقية بين موقعين في الذاكرة إذا صح التعبير أو متغيرين بشكل كامل وليس بشكل جزئي أو تعبيرين نتائجهما (True أو False) وتكون نتائج إي عملية فيها (&& أو ||) إما True أو False . أي ان المقارنة تكون بين (True أو False) بطرفي (&&).
مثلا:

```
Int a=5;
```

```
Int b=7;
```

```
(A >0) &&(b>0)
```

```
(True) && (True)→ True
```

يكون الناتج هو True لان المتغيران كلاهما أعلى من صفر . اي عندما قارن (A>0) كانت النتيجة نعم اكبر من الصفر إي (True) وعندما قارن (b>0) كانت النتيجة أيضا (True) وعندما عمل (and) بين النتيجتين كانت النتيجة (True) إي واحد
✓ * فلو كتبنا الكود التالي (a && b) فسيقارن (1 && 1 =1) تكون النتيجة واحد

إما (&) فيمثل عملية منطقية بين كل بت مع البت الذي يقابله في المتغير المقابل وليس مع المتغير كاملا لذلك تسمى **Bowties** .وناتج العملية ممكن أن يكون إي رقم لنعد نفس المثال:-

```
Int a=5;
```

```
Int b=7;
```

```
(A & b)
```

ماذا سيحصل؟ سوف يحول (٥ و ٧) إلى ما يقابلهما ب Binary ثم يعمل بينهم عملية and

```
0111 → 7
```

```
0101 → 5
```

```
0101 → 5
```

نلاحظ أن ناتج هذه العملية هو (5) وليس (True أو False) إي ممكن أن يكون الناتج إي رقم وكذلك بالنسبة ل (| و ||)

إضافة تعليقات-

التعليقات مهمة جدا بجانب الأسطر البرمجية حتى في ما بعد إذا أردنا فهم البرنامج الذي كتبناه في وقت مضى مجرد نقرا التعليقات بجانب الأسطر نفهم ما يعني الكود في لغة (C) نضع التعليق بين / * تعليق * /
وفي لغة (C++) نضع التعليق بعد // تعليق //

C++

لكود بلغة

```
#include<iostream.h> // declaration of Lab we will use it
```

C

لكود بلغة

```
#include<stdio.h> /* declaration of Lab we will use it*/
```

$(3|2) \&\& (4\&2) || (4<2)$

جد النتيجة.....؟

 $(3|2) \&\& (4\&2) || (4<2)$

1.

بما انه موجود تعابير وعمليات منطقية سينفذ العمليات التي بين الأقواس أولا
 أول عملية سينفذها هي (&) (4&2) يحولها الى الباينري ويعمل بين البتات (and)
 $(0100)\&(0010)=(0000)$
 النتيجة هي صفر

 $(3|2) \&\& 0 || (4<2)$

2.

سينفذ عملية الأصغر وان الاثنان ليس اصغر من أربعة فتكون النتيجة False اي صفر

 $(3|2) \&\& 0 || 0$

3.

سينفذ عملية or بين ثلاثة واثنان يحولها الى الباينري ويعمل بين البتات or
 $(0011)|| (0010)=(0011)$
 النتيجة هي ثلاثة

 $3 \&\& 0 || 0$

4.

الآن يقارن عملية (and) وعملية (or) نعلم انه and له أسبقية أعلى من or لذلك سينفذه أولا
 في عمليات المقارنة إي رقم غير الصفر يعتبره واحد سوا كان موجب أو سالب إذن تكون المقارنة
 بشكل التالي

 $(1 \&\& 0)$ $(1 \&\& 0=0)$

النتيجة هي واحد

 $0 || 0$

5.

آخر عملية مقارنة هي (or) بين الصفر والصفر والنتيجة هي صفر
 $(0 || 0)=0$

مهم

مؤثرات الزيادة والنقصان: هي متغيرات تعرف داخل البرنامج بقيمة معينة تم تغيير قيمتها أثناء تنفيذ البرنامج كأن نجعلها تزداد بمقدار واحد او اي إي مقدار معين وهي على نوعين..؟
مؤثرات الزيادة: هي متغيرات من اسمها تزداد بمقدار معين وشكلها

`a = a+1;`

معناه المتغير (a) ازداد بمقدار واحد (اي جمع قيمته السابقة مع (1) اي لو كان قيمته مثلا 2 يصبح 3)

-----مثال توضيحي-----

`int a=0`

عرفنا المتغير وأعطينا قيمته `a=0`

`a = a+1;`

جمعت قيمته السابقة وهي صفر مع الواحد وأصبحت قيمته `a=1`

`a = a+1;`

جمعت قيمته السابقة وهي واحد مع الواحد وأصبحت قيمته `a=2`

`a=a+5;`

جمعت قيمته السابقة وهي اثنان مع خمسة وأصبحت قيمته `a=7`

لاحظت كيف تغيرت قيمة المتغير a من الصفر إلى الواحد ثم إلى الاثنان بمجرد كتابته هذا التعبير هناك طريقة أخرى لتمثيل مؤثرات الزيادة وهي

نكتب `a++` بدلا أن نكتب `(a=a+1;)`
أو `a+=1`
`++ a`

وتدل الرموز السابقة اي أن المتغير a قد ازداد بمقدار واحد.
لو أردنا أن يزداد المتغير قيمة غير الواحد قد يكون 5 أو إي رقم آخر ف

نكتب `a+=5` بدلا أن نكتب `(a=a+5;)`

-----يصبح المثال توضيحي بالرموز البرمجية-----

`int a=0`

عرفنا المتغير وأعطينا قيمته `a=0`

`a ++;`

أصبحت قيمته `a=1`

`a +=1;`

أصبحت قيمته `a=2`

`a+= 5;`

أصبحت قيمته `a=7`



ما الفرق بين $(a++)$ و $(++a)$ ؟..

الاختلاف هو انه $(a++)$ معناه نفذ الخطوة الحالية على قيمتك السابقة تم غير قيمتك بعد أن تنفذ الخطوة الحالية (على سبيل المثال قد تكون قيمة الزيادة هي واحد وقيمتها في الخطوة السابقة (٢) ينفذ الخطوة التي هو بها على انه (٢) وعندما ينتقل إلى الخطوة اللاحقة تصبح قيمته (٣) .

```
1.a=2; المتغير بقيمة ٢ قبل الزيادة
2.a++; يبقى قيمة المتغير في هذه الخطوة ٢ ويصبح ثلاثة على الخطوة القادمة
3. أصبح قيمة المتغير ثلاثة.
```

☒ كأننا نقول أن إشارتي الجمع بعد المتغير مثلا $(a++)$ لا تؤثر بزيادتها على الخطوة الموجود بها (كما في هذه المثال خطوة رقم ٢) إنما تؤثر على الخطوة اللاحقة

إما $(++a)$ معناه غير قيمتك السابقة أولاً تم نفذ الخطوة التي أنت بها لذلك نراه في جدول الأسبقيات مقدم على باقي العمليات في أسبقيته. لو أعدنا المثال السابق

```
1.a=2; المتغير بقيمة ٢ قبل الزيادة
2.a++; يصبح قيمته ثلاثة في هذه الخطوة
```

مثال: يبين كيفية تأثير بمقدار التغير في الخطوة الموجود فيها والخطوة التي تليها ؟.

البرمجة بلغة C++	البرمجة بلغة C
<pre>1.#include<iostream.h> 2.main() 3.{ 4.int a=2; 5.int item; 6.item=3+a++; 7.cout<<"FirstTry="<< item ; 8.item=3+ a; 9.cout<<"\nSecondTry="<< item ; 10.}</pre>	<pre>1.#include<stdio.h> 2.main() 3.{ 4.int a=2; 5.int item; 6.item=3+a++; 7.printf("FirstTry=%d", item); 8.item=3+ a 9.printf("\nSecondTry=%d", item) ; 10.}</pre>

توضيح....؟

١. في السطر الربع عرفنا متغير (a) وأعطيناها قيمة بدائية وهي ٢ وفي السطر الخامس عرفنا متغير item ولم نعطه إي قيمة

٢. في السطر السادس حدثت عملية رياضية جمعت بين (٣) وقيمة المتغير (a) متأثراً بزيادة لكن كما بينا سابقاً أن هذه الزيادة الجديدة لا تؤثر على نتائج الخطوة الموجودة فيها إنما تؤثر على نتائج الخطوة التي تليه . لذلك يبقى المتغير (a) محتفظاً بقيمته في هذه الخطوة السادسة على قيمته البدائية (أو مقدار قيمته قبل الخطوة السادسة وهي ٢) لذلك ناتج جمع (٣+٢) هو خمسة وهو ما ظهر

تمثيل رياضي

```
6.item=3+ a++;
Item=3+2
Item=5
```

✗ وأصبح قيمة المتغير (a=3) بعد أن نفذ الخطوة رقم (٦) لأنه ازداد بمقدار واحد

لو كانت الخطوة السادسة هي

الكود

```
6.item=3+ ++a;
```

✗ لكان المتغير (a) تغيرت قيمته إلى ٣ قبل تنفيذ الخطوة رقم (٦) مؤثراً بقيمته الجديدة على

الخطوة التي هو بها وتصبح النتيجة قيمة المتغير item (٣+٣) وهي ٦

✗ لو نلاحظ في كلا الحالتين أصبح قيمة المتغير (a=3) لكن اختلفا في تأثيرهما في التي هما بها

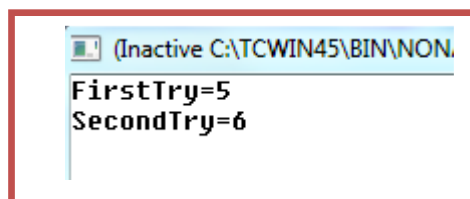


٣. السطر السابع طبعنا قيمة المتغير item وهي كما تظهر في شاشة التنفيذ ٥

٤. السطر الثامن قمنا بعملية رياضية جديدة جمعت بين ٣ وقيمة المتغير a بدون أي تغير في مقدار قيمة هذه المتغير لكن رغم ذلك كانت النتيجة ٦ وذلك لا المتغير a تغيرت قيمته في السطر السادس إلى ثلاثة لكنه بقا محتفظاً بقيمته لم يؤثر في الخطوة التي هو بها إنما أثر في الخطوة التي تليه فأصبحت النتيجة ٦ وليس ٥

تمثيل رياضي

```
8.item=3+ a
Item=3+3
Item=6
```



ناتج تنفيذ البرنامج

مثال تتبعي يبين أنواع تغير مقدار قيم المتغيرات بعمليات الزيادة والنقصان (مهم جدا)

```
main()
{
1. int a=2;
2. int b=0;
3. int item=0;
( a=2,b=0,item=0) هي الخطوة السابقة في بعد المتغيرات في بعد الخطوة السابقة هي
4. item=1+a++;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=3,b=0,item=3) وذلك لان قيمة المتغير a تبقى ثابتة في
الخطوة السابقة على قيمها في (الخطوة ١) وتتغير بعد (الخطوة ٤) إلى ٣
5. ++a;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=4,b=0,item=3) لان جميع المتغيرات لا تتأثر في
(الخطوة ٥) فقط المتغير a يتأثر ليصبح ٤ (المتغيرات التي لا تتأثر بالخطوة تبقى محتفظة بقيمتها السابقة)
6. item=item + ++b
// قيم المتغيرات في بعد الخطوة السابقة هي (a=4,b=1,item=4) لان المتغير b يتغير ال ١ مؤثرا في
الخطوة التي هو فيها وقيمة المتغير item تجمع مع قيمتها السابقة مع قيمة المتغير b
7. a++;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=5,b=1,item=4) لان جميع المتغيرات لا تتأثر في
(الخطوة ٧) فقط المتغير a يتأثر بتغير مقداره واحد ليصبح ٥ (المتغيرات التي لا تتأثر بالخطوة تبقى محتفظة بقيمتها
السابقة)
8. ++b;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=5,b=2,item=4) لان جميع المتغيرات لا تتأثر في
(الخطوة ٨)
فقط المتغير b يتأثر بتغير مقداره واحد ليصبح 2 (المتغيرات التي لا تتأثر بالخطوة تبقى محتفظة بقيمتها السابقة)
9. item=item + a++ - b++;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=6,b=3,item=7) المتغيران (a,b) يتغيران ويزدادان بمقدار
واحد لكن لا يؤثران على الخطوة التي هما بها والمتغير item يجمع قيمته a السابقة ويطرح من b السابقة
10. item=item + a++ - ++b;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=7,b=4,item=9) المتغير b يزداد ويؤثر في الخطوة التي هو
بها إما a يزداد لكن لا يؤثر بالخطوة التي هو بها والمتغير item يجمع قيمته السابقة مع a ويطرحها من b
11. item=++b;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=7,b=5,item=5) المتغير a يبقى ثابت المتغير b يزداد بمقدار
واحد ويؤثر في الخطوة التي هو بها والمتغير item يأخذ قيمة ال b الجديدة
12. item+=5;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=7,b=5,item=10) فقط قيمة المتغير item تتغير لتجمع
قيمتها السابقة مع الرقم ٥
13. b=b+5;
// قيم المتغيرات في بعد الخطوة السابقة هي (a=7,b=10,item=10) فقط قيمة المتغير b تتغير لتجمع
قيمتها السابقة مع الرقم ٥
}
```

مؤثرات النقصان: هو نقصان من قيمة المتغير بمقدار واحد أو أكثر

توضيح

`a = a-1;`

معناه المتغير (a) نقص بمقدار واحد (أي طرح من قيمته السابقة بمقدار 1) أي لو كان قيمته مثلا 2 يصبح 1

-----مثال توضيحي-----

`int a=2`

عرفنا المتغير وأعطينا قيمته `a=0`

`a = a-1;`

طرحنا من قيمته السابقة وهي اثنان مقدار الواحد وأصبحت قيمته `a=1`

`a = a-1;`

طرحنا من قيمته السابقة وهي واحد مقدار الواحد وأصبحت قيمته `a=0`

`a=a-5;`

طرحنا من قيمته السابقة وهي صفر مقدار خمسة وأصبحت قيمته `a=-5`

لاحظت كيف تغيرت قيمة المتغير a من الاثنان إلى الواحد إلى الصفر إلى سالب خمسة بمجرد كتابته هذا التعبير. هناك طريقة أخرى لتمثيل مؤثرات النقصان وهي

بدلاً من نكتب `(a=a-1);` نكتب `a--` أو `a -= 1` أو `--a`

وتدل الرموز السابقة على أن المتغير a قد نقص بمقدار واحد. لو أردنا أن ينقص المتغير قيمة غير الواحد قد يكون 5 أو أي رقم آخر فنكتب

بدلاً من نكتب `(a=a-5);` نكتب `a-=5`

----- يصبح المثال توضيحي بالرموز البرمجية -----

`int a=2`

عرفنا المتغير وأعطينا قيمته `a=0`

`a --;`

أصبحت قيمته `a=1`

`a -=0;`

أصبحت قيمته `a=2`

`a-= 5;`

أصبحت قيمته `a=-5`

ما الفرق بين (a--, --a) ؟..

الاختلاف هو انه (a--) معناه نفذ الخطوة التي أنت بها على قيمتك السابقة ثم غير قيمتك (كأن يكون قيمة النقصان هي واحد وقيمه في الخطوة السابقة ٢ ينفذ الخطوة التي هو بها على انه ٢ وعندما ينتقل إلى الخطوة اللاحقة تصبح قيمته ١)

```
1.a=2; المتغير بقيمة ٢ قبل النقصان
2.a--; يبقى قيمة المتغير في هذه الخطوة ٢ ويصبح واحد على الخطوة القادمة
3. أصبح قيمة المتغير واحد
```

إما (--a) معناه غير قيمتك السابقة أولاً ثم نفذ الخطوة التي أنت بها لذلك نراه في جدول الأسبقيات مقدم على باقي العمليات في أسبقيته . لنعد المثال السابق

```
1.a=2; المتغير بقيمة ٢ قبل النقصان
2.a++; يصبح قيمته واحد في هذه الخطوة
```

مثال: يبين كيفية تأثير بمقدار التغير في الخطوة الموجود فيها والخطوة التي تليها ..؟

c++	البرمجة بلغة c	البرمجة بلغة c++
1.#include<iostream.h> 2.main() 3.{ 4.int a=2; 5.int item; 6.item=3+a--; 7.cout<<"FirstTry="<< item ; 8.item=3+ a; 9.cout<<"\nSecondTry="<< item ; 10.}	1.#include<stdio.h> 2.main() 3.{ 4.int a=2; 5.int item; 6.item=3+a--; 7.printf("FirstTry=%d", item) ; 8.item=3+ a; 9.printf("\nSecondTry=%d", item) ; 10. }	

توضيح....؟

١. في السطر الربع عرفنا متغير (a) وأعطيناها قيمة بدائية وهي ٢ وفي السطر الخامس عرفنا متغير item ولم نعطه إي قيمة

٢. في السطر السادس حدثت عملية رياضية جمعت بين (٣) وقيمة المتغير (a) متأثراً بنقصان لكن كما بينا سابقاً أن هذه النقصان الجديد لا تؤثر على نتائج الخطوة الموجود فيها إنما تؤثر على نتائج الخطوة التي تليه . لذلك يبقى المتغير (a) محتفظاً بقيمته في هذه الخطوة السادسة على قيمته البدائية (أو مقدار قيمته قبل الخطوة السادسة وهي ٢) لذلك ناتج جمع (٣+٢) هو خمسة وهو ما ظهر

تمثيل رياضي

```
6.item=3+ a--;  
Item=3+2  
Item=5
```

✗ وأصبح قيمة المتغير (a=1) بعد أن نفذ الخطوة رقم (٦) لأنه تناقص بمقدار واحد

لو كانت الخطوة السادسة هي

الكود

```
6.item=3+ --a;
```

✗ لكان المتغير (a) تغيرت قيمته إلى واحد قبل تنفيذ الخطوة رقم (٦) مؤثراً بقيمته الجديدة على

الخطوة التي هو بها وتصبح النتيجة قيمة المتغير item (٣+١) وهي ٤

✗ لو نلاحظ في كلا الحالتين أصبح قيمة المتغير (a=1) لكن اختلفا في تأثيرهما في التي هما بها



٣. السطر السابع طبعنا قيمة المتغير item وهي كما تظهر في شاشة التنفيذ ٥

٤. السطر الثامن قمنا بعملية رياضية جديدة جمعت بين ٣ وقيمة المتغير a بدون أي تغير في مقدار

قيمة هذه المتغير لكن رغم ذلك كانت النتيجة (4) وذلك لا المتغير a تغيرت قيمته في السطر

السادس إلى واحد لكنه بقا محتفظاً بقيمته لم يؤثر في الخطوة التي هو بها إنما اثر في الخطوة التي تليه

فأصبحت النتيجة ٤ وليس ٥

تمثيل رياضي

```
8.item=3+ a  
Item=3+1  
Item=4
```

```
FirstTry=5  
SecondTry=4
```

ناتج تنفيذ البرنامج

بعض دوال الإدخال والإخراج في لغة (C) في مكتبة <stdio.h>

1. (getchar) تأخذ هذه الدالة حرف واحد يدخله المستخدم من شاشة التنفيذ ويظهر هذا الحرف أمام المستخدم. نضغط مفتاح (enter) بعد إدخال الحرف لكي ينفذ الخطوة.

2. (putchar) يطبع حرف واحد فقط في شاشة التنفيذ (هذا مثال على كيفية استخدام هاتان الدالتان).

C	الكود بلغة
	<pre>1.#include<stdio.h> 2.main(){ 4.char symbol; 5. symbol=getchar; 6. Putchar(symbol); }</pre>

بعض دوال الإدخال في مكتبة <conio.h>

1. (getch) تقرأ هذه الدالة حرف واحد يدخله المستخدم من شاشة التنفيذ ولا يظهر هذا الحرف أمام المستخدم (كما كان يظهر في دوال الإدخال الأخرى). ولا نضغط مفتاح (enter) بعد إدخال الحرف فقط نكتب الحرف وهو ينفذ.

هذا مثال على كيفية استخدام هذه الدالة (يجب تضمين المكتبة conio في الحل لأننا نستخدم دوالها)

c++	البرمجة بلغة	c	البرمجة بلغة
	<pre>#include<iostream.h> #include<conio.h> main(){ char symbol; symbol=getch(); cout<<"The symbol your enter is=" << symbol; }</pre>		<pre>#include<stdio.h> #include<conio.h> main(){ char symbol; symbol=getch(); Printf("The symbol your enter is=%c", symbol); }</pre>

لاحظ ما سيظهر في شاشة التنفيذ عند الضغط على حرف (w) من لوحة المفاتيح بدون ضغط مفتاح (enter). لم يظهر الحرف الذي أدخلته إنما فقط نفذ وظهرت الرسالة

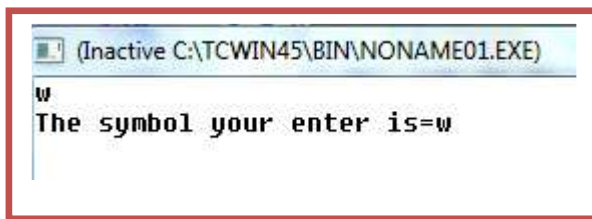


2.(getche) تقرأ هذه الدالة حرف واحد يدخله المستخدم من شاشة التنفيذ و يظهر هذا الحرف أمام المستخدم .ولا نضغط مفتاح (enter) بعد إدخال الحرف فقط نكتب الحرف وهو ينفذ.

هذا مثال على كيفية استخدام هذه الدالة

البرمجة بلغة	c	البرمجة بلغة	C++
	<pre>#include<stdio.h> #include<conio.h> main(){ char symbol; symbol=getche(); Printf("\nThe symbol your enter is=%c", symbol); }</pre>		<pre>#include<iostream.h> #include<conio.h> main(){ char symbol; symbol=getche(); cout<<"\nThe symbol your enter is=" << symbol; }</pre>

لاحظ ما سيظهر في شاشة التنفيذ عند الضغط على حرف (w) من لوحة المفاتيح بدون ضغط مفتاح (enter)



دوال الرياضية في مكتبة <math.h>

تستخدم دوال هذه المكتبة في حل العمليات الرياضية كإيجاد جيب أو جيب تمام أو قيمة مطلقة وغيرها وهذا شرح لبعض دوالها

1.(sin) تستخدم هذه الدالة لإيجاد جيب الزاوية بالنظام ال (rad) لذلك لإيجاد جيب الزاوية بالنظام (deg) فقط نضرب قيمة الزاوية ب $\pi/180$

الكود (إيجاد جيب الزاوية ٩٠)

```
Float sin x;
X=sin(90*(3.14/180) );
```


2.cos تستخدم هذه الدالة لإيجاد جيب تمام الزاوية بالنظام ال (rad) لذلك لإيجاد جيب الزاوية بالنظام (deg) فقط نضرب قيمة الزاوية ب $\pi/180$

الكود (إيجاد جيب تمام الزاوية ٩٠)

```
Float x;  
X=cos(90*(3.14/180));
```

لإيجاد بقية الدوال المثلثية جميعها تكون مشتقة من جيب وجيب تمام. إي إننا نحول إي دالة مثلثيه إلى جيب أو جيب تمام حسب تحويلها بالدوال المثلثية .
مثلا لإيجاد الضل الزاوية فقط نقسم جيب على جيب تمام الزاوية

عملية رياضية لإيجاد ضل ٩٠

```
Float Tanx;  
Tanx =sin (90*(3.14/180) ) / cos(90*(3.14/180));
```

وكذلك بقية الدوال المثلثية بنفس الطريقة

3.pow تستخدم هذه الدالة لإيجاد قيمة رقم مرفوع إلى أس (مثلا $3^2=9$). طريقة تمثيل هذه الدالة
X=pow(number,hispower);

حيث أن **number** هو الرقم و **hispower** هو الأس المرفوع له
مثال: لو كان لدينا (3^9) إي ثلاثة مرفوع لأس تسعة فيكتب برمجيا باستخدام هذه الدالة هكذا

الكود

```
X=pow(3,9);
```

4.abs هي القيمة المطلقة للرقم

الكود

```
X=abs(-3); // x=3
```

5.sqrt هي دالة تستخدم لإيجاد جذر الرقم

الكود

```
X=sqrt(25); // x=5
```

دوال أخرى!

1.(size of) تجد هذه الدالة الحجم الذي يشغله المتغير في الذاكرة . عدد البايتات التي يحجزها له.

الكود

```
X=sizeof(int); // 2 byte is the size of integer
```

2.(int) تحول هذه الدالة المتغيرات من أنواع أخرى إلى متغير من نوع integer. وأيضا تجد قيمة (AScii) للحرف. الاسكي كود: كل حرف من الأحرف الانكليزية له أسكي كود خاص به يختلف عن غيره من الأحرف مثلا أسكي كود الحرف (a) صغيرة هو (97) ويختلف عن أسكي كود (A) كبيرة الذي هو (65). وترتيب أسكي كود الأحرف بالتسلسل إي أن أسكي كود (b) هو (98) ويزداد بالتتالي (c=99,d=100,e=101-----)

الكود (تحويل متغير كسري إلى متغير من نوع integer)

```
Int x;  
x=int(3.5); // x=3
```

الكود (الحصول على أسكي كود الحرف)

```
Int x;  
x=int('a'); // x=97
```

3.(char) تحول هذه الدالة الرقم إلى قيمة الأس كي كود الخاصة به.

الكود (الحصول على الحرف من أسكي كود)

```
char x;  
x=int(97); // x=a
```

مثال : برنامج ندخل حرف أو رمز ويعطيك الاسكي كود له ..؟

البرمجة بلغة	c	البرمجة بلغة	c++
	<pre>#include<stdio.h> #include<conio.h> main(){ 1.char symbol; 2.symbol=getche(); 3.printf("\nAScii=%d", symbol);}</pre>		<pre>#include<iostream.h> #include<conio.h> main(){ 1.char symbol; 2.symbol=getche(); 3.cout<<"\n AScii=" << int(symbol);}</pre>

توضيح المثال: في الحل بلغة (c) فقط في خطوة رقم (3) وضعنا رمز الطباعة (%d) حتى يطبع اسكي كود الحرف وليس الحرف نفسه.

يختلف الحرف الكبير عن الحرف الصغير ب (32) رقم إي أسكي كود (a) صغيرة هو (97) و (A) كبيرة هو (65) للتحويل من كبير إلى صغير نزيد 32 والعكس نطرح 32

الكود (تحويل حرف صغير إلى حرف كبير مثلا (a) أصبح (A))

```
char x='a';  
x=char(int('a')-32) // x=A
```

ماذا اكتشفت بعد ان انتهينا من الفصل

أن لغة (C) ولغة (C++) تتشابهان في اغلب تعابيرهما الرياضية وأكثر الاختلافات بينهما هي المكتبات ودوال الإدخال ودوال الإخراج أي لو أبدلت فقط (اسم المكتبة ودوال الإدخال ودوال الإخراج تستطيع تحويل البرنامج من لغة (C) إلى لغة (C++) والعكس صحيح

ودوال الإخلاف هي

الاختلافات	لغة (C)	ولغة (C++)
المكتبة	#include<stdio.h>	#include<iostream.h>
دوال الإدخال	scanf	cin
دوال الإخراج	printf	Cout

وفي بعض الأمثلة سوف لا اكتبها بالغتتين على حدة اكتب بلغة واحدة وأنت تستطيع التحويل وإذا كان هناك أكثر من هذه الاختلافات الثلاث أنا سوف اكتب البرنامج بلغتين

الفصل الثاني

الجملة الشرطية والعبارات الشرطية (Control Structures)

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصل الأول وفهما كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على الجملة الشرطية وطريقة استخدام العبارات الشرطية ومواقعها وأنواعها

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب : ثلاث ساعات

١. عبارة (if) الشرطية الاعتيادية:-

هي عبارة أو جملة لا ينفذ ما في داخلها (إي statement الموجود بين قوسين العبارة الشرطية) إلا بتحقق الشرط الموجود بعد عبارة if وهو (condition) أي يجب أن تكون نتيجة مقارنة شروط هي (True) حتى ينفذ ما بين قوسي العبارة الشرطية.

شكل عبارة if الشرطية بطريقة اعتيادية

```
If ( condition )
{
statement
}
```

أي أن حسب مخطط الفصل الأول الخاص بتسلسل تنفيذ خطوات البرنامج بالتتابع. في العبارة الشرطية (if)

- إذا لم يحقق الشرط (condition) أي كانت نتيجة المقارنة هي (False) سوف يعبر جميع الخطوات الموجودة بين قوسي العبارة الشرطية (if) ولا ينفذها (وعرفنا في الفصل الأول كيف تكون مقارنة الشروط)
- وإذا تحقق الشرط (condition) أي كانت نتيجة المقارنة هي (True) ينفذ الخطوات الموجودة بين قوسي العبارة الشرطية (if) بشكل اعتيادي لاحظ المخطط التوضيحي لسير البرنامج

✓ إذا لم نضع أقوس خلف العبارة الشرطية معناه يتبعها فقط السطر الذي يليها أما إذا وضعنا أقواس خلفها فكل الذي بين القوسين يكون تابع للعبارة الشرطية تنفذ إذا تحقق الشرط ولا تنفذ إذا لم يتحقق الشرط

❖ برنامج يدخل المستخدم من شاشة التنفيذ وإذا كان الرقم اكبر من خمسة يعطيه رسالة انه اكبر من خمسة

C++

الكود بلغة

```
#include<iostream.h>
من هنا يبدأ تنفيذ البرنامج خطوة بخطوة/هذه أول خطوة
Main()
{
الخطوات الموجودة ضمن هذه الدالة ينفذها تباعا
1.int a;
2.cin>>a;
If ( a>5 )
{3.cout<<"is greter than 5";
}
4. a=a+2;
5.cout<<"a"<<a ;
}
```

```
#include<stdio.h>
من هنا يبدأ تنفيذ البرنامج خطوة بخطوة (هذه أول خطوة)
main()
{
الخطوات الموجودة ضمن هذه الدالة ينفذها تباعا
1.int a;
2 scanf("%d",&a);
If ( a>5 )
{3.printf("is greter than 5");
}
4. a=a+2;
5.printf("a=%d",a);
}
```



توضيح الخطوات:

١. يبدأ البرنامج بتنفيذ خطوة رقم (١) بتعريف متغير (a) من نوع integer
 ٢. ثم ينفذ خطوة رقم (٢) ويطلب فيها من المستخدم إدخال قيمة للمتغير (a) من شاشة التنفيذ
 ٣. ثم يتحقق من الخطوة رقم (٣)
- ✓ إذا كان الرقم الذي ادخله المستخدم يحقق الشرط بين قوسي العبارة الشرطية (if) (أي الرقم اكبر من خمسة) أي انه سوف تكون نتيجة المقارنة (condition) هي (True) ومثلاً
- لنفرض انه أدخل الرقم ٦ فتكون المقارنة هكذا.



$(a>5) \rightarrow (6>5) \rightarrow \text{True}$

سوف ينفذ ما موجود في قوسي العبارة الشرطية (if) أي سينفذ الخطوة رقم (٣) ثم ينفذ خطوة رقم (٤) و ثم (٥) فتكون تسلسل تنفيذ خطوات البرنامج

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

- ✓ وإذا كان الرقم لا يحقق الشرط بين قوسي العبارة الشرطية (if) (أي الرقم اصغر أو يساوي خمسة) أي انه سوف تكون نتيجة المقارنة (condition) هي (False) ومثلاً
- لنفرض انه أدخل الرقم 2 فتكون المقارنة هكذا.

$(a>5) \rightarrow (2>5) \rightarrow \text{False}$



سوف لا ينفذ ما موجود في قوسي العبارة الشرطية (if) أي لا ينفذ الخطوة رقم (٣) إنما ينتقل إلى ما بعد القوس العبارة الشرطية لينفذه أي سينفذ بعد الخطوة (٢) مباشرة الخطوة رقم (٤) و ثم (٥) فتسلسل تنفيذ خطوات

$1 \rightarrow 2 \rightarrow 4 \rightarrow 5$

العلاقات التي تستخدم كشروط (condition) في العبارة الشرطية (if) هي

الرمز	الوظيفة	طريقة تمثيله في if الشرطية
>	علامة الأكبر	if(a>b) مقارنة بين متغيرين (a,b) ينفذ الجملة التابعة لعبارة if إذا كان a اكبر من b
<	علامة الأصغر	if(a<b) ينفذ الجملة التابعة لعبارة if إذا كان a اصغر من b
>=	علامة اكبر أو يساوي	if(a>=b) ينفذ الجملة التابعة لعبارة if إذا كان a اكبر أو يساوي b
<=	علامة اصغر أو يساوي	if(a<=b) ينفذ الجملة التابعة لعبارة if إذا كان a اصغر أو يساوي b
==	علامة التساوي	if(a==b) ينفذ الجملة التابعة لعبارة if إذا كان a يساوي b
!=	علامة لا يساوي	if(a!=b) ينفذ الجملة التابعة لعبارة if إذا كان a لا يساوي b
&&	جمع شرطين ب and	if((a>b)&&(a>c)) ينفذ الجملة التابعة لعبارة if إذا كان a اكبر من b وأيضا يكون a اكبر من C إي يجب أن يتحقق الشرطان حتى تنفذ الجملة
	جمع شرطين ب OR	if((a>b) (a>c)) ينفذ الجملة التابعة لعبارة if إذا كان a اكبر من b أو يكون a اكبر من C إي إذا تحقق واحد من هذان الشرطان تنفذ الجملة

✓ حيث أن (>,<,>=,<=,==,!=) تستخدم كعلاقة بين متغيرين أو قيمتين أو متغير وتعبير رياضي
 ✗ شكل تمثيل الشروط في العبارة الشرطية (if)



✓ التعبير الرياضي ممكن أن يكون إي عمليات رياضية ونتيجة التعبير تدخل في المقارنة
 ✓ (&&,||) تستخدم علاقة بين مجموعتين كل متغيرين على حدة (وان And له أسبقية على OR)

✗ ما هي نتائج مقارنة العمليات التالية

(3>2) → True
 (5!=7) → True
 (23>=120) → False
 (21 <11) → False
 (4==2) → False
 (43<=76) → True
 ((3+2)>4) → (5>4) → True

وأن الشروط داخل العبارة الشرطية هي أيضا تنفذ حسب الأسبقيات أي أن أي شرط أسبقيته أعلى ينفذ أولا وهذا ما شرح عنه مفصلا في الفصل الأول في الأمثلة على الأسبقيات .

بعض الأمثلة التوضيحية على طريقة استخدام العبارة الشرطية (if).

مثال ١: لو كان عندنا متغيران (a , b) يدخل المستخدم قيمهما من شاشة التنفيذ وكان المطلوب طبع رسائل تبين متى كان (a أكبر من b ، ، a أكبر أو يساوي b ، ، a اصغر من b ، ، a اصغر أو يساوي b ، ، a لا يساوي b) ؟.

تحليل السؤال: يوجد متغيران يجب تعريفهما في بداية البرنامج وقراءتهما من شاشة التنفيذ وبعدها التحقق من الشروط الموجود في السؤال على هذان المتغيران وطبع رسائل لكل شرط

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> main() { 1.int a,b; 2.cin>>a>>b; if (a>b) 3.cout<<"a is large than b"; if (a>=b) 4.cout<<"\na is large than or equal b"; if (a<b) 5.cout<<"\na is less than b"; if (a<=b) 6.cout<<"\na is less than or equal b"; If (a==b) 7.cout<<"\na is equal than b"; if (a!=b) 8.cout<<"\na is not equal than b"; }</pre>	<pre>#include<stdio.h> main() { 1.int a,b; 2.scanf("%d%d",&a,&b); if (a>b) 3.printf("a is large than b"); if (a>=b) 4. printf("\na is large than or equal b"); if (a<b) 5. printf("\na is less than b"); if (a<=b) 6. printf("\na is less than or equal b"); if (a==b) 7. printf("\na is equal than b"); if (a!=b) 8. printf("\na is not equal than b"); }</pre>

توضيح الخطوات :

١. تنفذ أولاً خطوة رقم (١) هي تعريف للمتغيرات (a,b)
٢. ثم تنفذ خطوة رقم (٢) هي قراءة للمتغيرات التي ستدخل قيمها من قبل المستخدم لأن لاختبار أن ندخل قيم من شاشة التنفيذ ونرى النتائج
- ☒ لو أدخلنا (a=3,b=2) كما لاحظت قد تنفذ الخطوات رقم (3,4,8) لأن شروطها تحققت **وذلك لأنه**

١. الشرط التابع للخطوة رقم (٣) هو أن يكون قيمة a أكبر من قيمة b وفعلاً أن قيمة a التي أدخلناها كانت ٣ وقيمة b=2 لذلك نفذ الخطوة الثالثة لاحظ التحقق من الشرط

$(a>b) \rightarrow (3>2) \rightarrow \text{True}$

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
3 2
a is large than b
a is large than or equal b
a is not equal than b
```


٢. الشرط التابع للخطوة رقم (٤) هو أن يكون قيمة a اكبر أو يساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ أي كانت اكبر $b=2$ لذلك نفذ الخطوة (٤) * وشرط ($>=$) يتحقق إذا كان الرقم اكبر منه أو يساويه ينفذ عبارة التابعة للشرط

$(a>=b) \rightarrow (3>2) \rightarrow \text{True}$

٣. الشرط التابع للخطوة رقم (٨) هو أن يكون قيمة a لا يساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ وقيمة $b=2$ أي كانت لا تساوي لذلك نفذ الخطوة رقم (٨)

*** وبقية الخطوات لم تنفذ لأنها لم تتحقق شروطها

على سبيل المثال لماذا لم تنفذ الخطوة رقم (٥) ؟
لأن شرطها أن يكون a اصغر من قيمة b وان هذا الشرط لا يتحقق لأن قيمة a اكبر من قيمة b

$(a<b) \rightarrow (3<2) \rightarrow \text{False}$

نتيجة المقارنة هي (false) لذلك سوف لا ينفذ الخطوة التي تتبعه لان الشرط لم يتحقق

☒ لو أدخلنا $(a=3, b=6)$ كما لاحظت قد تحققت الخطوات رقم (5,6,8) لأن شروطها تحققت

وذلك لأنه

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)

```
3 6
a is less than b
a is less than or equal b
a is not equal than b
```

١. الشرط التابع للخطوة ٥ هو أن يكون قيمة a اصغر من قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ وقيمة $b=6$ لذلك نفذ الخطوة رقم (٥)

$(a<b) \rightarrow (3<6) \rightarrow \text{True}$

٢. الشرط التابع للخطوة 6 هو أن يكون قيمة a اصغر أو يساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ أي كانت اصغر $b=6$ لذلك نفذ الخطوة 6 * وشرط ($<=$) يتحقق إذا كان الرقم اصغر منه أو يساويه ينفذ عبارة التابعة للشرط

$(a<=b) \rightarrow (3<=6) \rightarrow \text{True}$

٣. الشرط التابع للخطوة ٨ هو أن يكون قيمة a لا يساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ وقيمة $b=6$ أي كانت لا تساوي لذلك نفذ الخطوة ٨

$(a!=b) \rightarrow (3!=6) \rightarrow \text{True}$

*** وبقية الخطوات لم تنفذ لأنها لم تتحقق شروطها

(Inactive C:\TCWIN45\BIN\NONAME01)

```
3 3
a is large than or equal b
a is less than or equal b
a is equal than b
```

☒ لو أدخلنا $(a=3, b=3)$ كما لاحظت قد تحققت الخطوات رقم (4,6,7)

وذلك لأنه

١. الشرط التابع للخطوة ٤ هو أن يكون قيمة a اكبر أو يساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ وقيمة $b=3$ متساويتان $b=3$ لذلك نفذ الخطوة ٤

$(a>=b) \rightarrow (3>=3) \rightarrow \text{True}$

٢. الشرط التابع للخطوة 6 هو أن يكون قيمة a اصغر أو يساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ أي كانت تساوي b=3 لذلك نفذ الخطوة 6 * وشرط (<=) يتحقق إذا كان الرقم اصغر منه أو يساويه ينفذ عبارة التابعة للشرط

$(a <= b) \rightarrow (3 <= 3) \rightarrow \text{True}$

٢. الشرط التابع للخطوة ٧ هو أن يكون قيمة a تساوي قيمة b وفعلا أن قيمة a التي أدخلناها كانت ٣ أي كانت تساوي b=3 لذلك نفذ الخطوة ٧

$(a == b) \rightarrow (3 == 3) \rightarrow \text{True}$

*** وبقية الخطوات لم تنفذ لأنها لم تتحقق شروطها

كما لاحظت في المثال لم نضع أقواس للأسطر البرمجية التي تتبع كل عبارة if لان كل عبارة لم يتبعها أكثر من سطر برمجي واحد وكما قلنا سابقا إذا كانت عبارة الشرط يتبعها سطر برمجي واحد لا حاجة لوضع أقواس وإذا أكثر من سطر نضع أقواس

مثال ٢: برنامج ندخل رقم ويطبع رسالة إذا كان الرقم بين (٥-١٠٠) إذا كان الرقم ضمن هذه الفترة وإذا خارج الفترة لا نطبع أي شيء ؟.

تحليل:

١. لدينا رقم ندخله من شاشة التنفيذ لذلك نحن بحاجة تعريف متغير يخص الرقم ودالة لإدخاله من شاشة التنفيذ وليكن اسم المتغير هو (a)

٢. لدينا شرط أن يقع ضمن فترة (٥-١٠٠) في مثل هذه الأسئلة نحن بحاجة إلى دمج أكثر من شرط في عبارة شرطية واحد كأن نقول له في الشرط أن يكون الرقم خمسة و اصغر من مئة

شرط اكبر من خمسة بسيط وهو

الكود

```
If ( a > 5 )
```

وشرط اصغر من مئة أيضا بسيط وهو

الكود

```
If ( a < 100 )
```

لكن في السؤال يقول ضمن الفترة وليس اكبر من خمسة على حدة واصغر من مئة على حدة إذن كيف ندمج هذان الشرطان؟ بما انه قال اكبر من خمسة واصغر من مئة واستخدم عبارة (و) معنا توجد عملية (and) بين هذان الشرطان أي لا تنفذ هذه الجملة الشرطية إلا بتحقق هذان الشرطان هكذا

الكود

```
If ( ( a>5 )&&(a<100) )
```

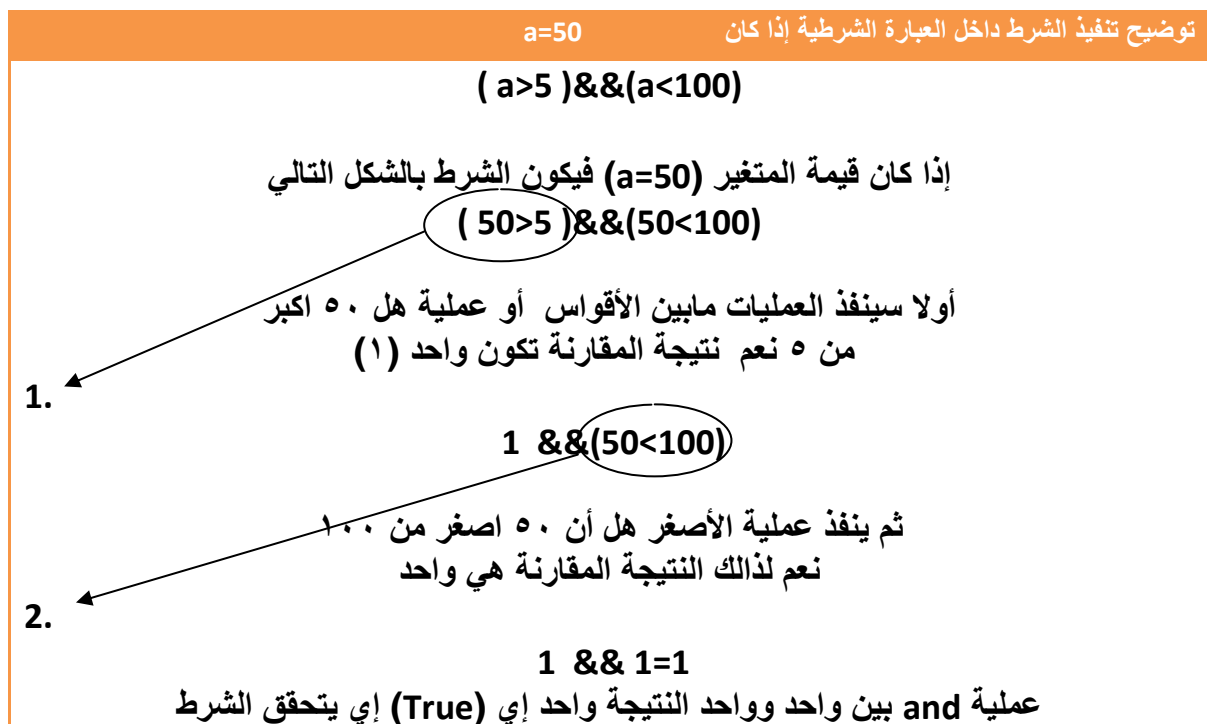
مهم

وراجع جدول (and) لفهم الفكرة أكثر لأنه لو قلنا عملية OR بين الشرطين لكان هناك اختلاف كبير في الحل أي لو تحقق احد الشرطان في جملة OR تنفذ عبارة if إما and الجملة لا تنفذ إلا بتحقق الشرطين. ليصبح الحل هكذا

البرمجة بلغة	c	البرمجة بلغة	c++
	<pre>#include<stdio.h> main() 1.{int a; 2 scanf("%d",&a); 3.if ((a>1)&&(a<100)) 4.printf("the number in this range"); }</pre>		<pre>#include<iostream.h> main() 1.{int a; 2.cin>>a; 3.if ((a>5)&&(a<100)) 4.cout<<"the number in this range"; }</pre>

توضيح الخطوات :

- خطوة رقم (١) عرفنا متغير من نوع integer لكي نحملة بالقيمة المدخلة
- خطوة رقم (٢) هي إدخال قيمة للمتغير من قبل المستخدم
- خطوة رقم (٣) هو عملية التحقق من الشرط
- فلو ادخل المستخدم الرقم (٤٠) فتكون المقارنة بشكل التالي



✘ فلو ادخل المستخدم الرقم (١٥٠) فتكون المقارنة بالشكل التالي

توضيح تنفيذ الشرط داخل العبارة الشرطية إذا كان $a=50$

$(a>5) \&\& (a<100)$

إذا كان قيمة المتغير $(a=150)$ فيكون الشرط بالشكل التالي

$(150>5) \&\& (150<100)$

أولا سينفذ العمليات ما بين الأقواس أو عملية هل ١٥٠ اكبر من ٥ نعم نتيجة المقارنة تكون واحد (١)

1.

$1 \&\& (150<100)$

ثم ينفذ عملية الأصغر هل أن ١٥٠ اصغر من ١٠٠ كلا لذلك النتيجة المقارنة هي صفر

2.

$1 \&\& 0=0$

عملية and بين واحد وصفر النتيجة صفر إي (False) إي لا يتحقق الشرط

مثال ٣- الشرط المطلوب هو أن قيمة المتغير (c) يجب أن تقبل القسمة على ثلاثة وعلى ستة أو تقبل القسمة على تسعة ليحقق الشرط ؟

التحليل. الشرطان الأولان أن يقبل القسمة على (٣ و ٦) إي ناتج باقي قسمتهما هو صفر والى جانب هذان الشرطان أن يقبل القسمة على تسعة إي إما يقبل القسمة على (٣ و ٦) أو يقبل القسمة على ٩ إي يوجد بين شرطا تحقق قبول القسمة على (٣ و ٦) عملية and لأنه واجب تحققهما معا وناجتهما داخل على عملية OR مع شرط قبول القسمة على ٩

الكود

```
If ( ( ( c%3==0) && (c%6==0) ) || (c% 9==0) )
```

مهم

*إذا لم نضع أقواس حول العمليات المطلوب تحققهما معا قد لا يحقق المطلوب لأنه تحكمه الأسبقية الشرط الذي له أسبقية أعلى ينفذ أولا

لو عدنا إلى نفس الشرط السابق وفرضنا أن بين شرط قبول القسمة على (٣ و ٦) يوجد عملية OR وليس عملية AND وواجب تحقق احدهما والناتج لهما يدخل على AND مع شرط تحقق القبول القسمة على ٩ وليس OR (ولم نضع أقواس حول شرطين تحقق قبول القسمة على ٣ أو ٦)

انظر ماذا سينتج

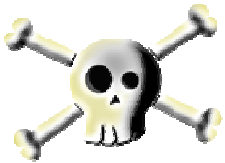
الكود

```
if ( ( c%3==0 ) || ( c%6==0 ) && ( c% 9==0 ) )
```

فالذي ينتج سوف تحدث عملية AND قبل عملية OR لان أسبقيتها أعلى منه ويصبح السؤال كأننا نقول يجب أن يتحقق قبول القسمة على ٩ مع قبول القسمة على ٦ معا وناجتهما داخل على عملية OR مع شرط قبول القسمة على ٣ . ويقلب حل السؤال ويكون الجواب خاطئا لذلك تجنب لمثل هذه المشاكل ضع أقواس حول الشروط التي يجب تحققها معا لتجنب مشاكل الأسبقيات كما في الشكل التالي

الكود

```
if ( ( ( c%3==0 ) || ( c%6==0 ) ) && ( c% 9==0 ) )
```



لماذا نستخدم بين أكثر من شرط *and* أو *OR* لماذا لا نجمع أكثر من شرط معا؟

الإجابة : شاهد المثال التالي الذي نستخدم فيه أكثر من شرط بدون (*and* أو *OR*)

الكود

```
int a=6;  
if ( a>5>3)
```

فالذي سوف يحدث كما تشاهد في المخطط انه سيقارن قيمة (*a*) مع (*٥*) ووجد انه قيمته اكبر (*a>5=True*) ثم سيقارن نتيجة مقارنة وهي (*True*) مع قيمة (*٣*) وبرمجيا قيمة (*True=1, False=0*) اي سيقارن قيمة (*١*) مع (*٣*) ووجد انه قيمة واحد اقل (*1>3=False*) وأصبحت النتيجة خاطئة وليست صائبة

ورغم أنه قيمة المتغير (*a*) هو اكبر من (*٣*) واكبر من (*٥*) لكن في البرمجة عند مقارنة متغيرين أو قيمتين يعطيك نتيجة مقارنة ونتائج المقارنة إما تكون (*True* أو *False*) وعند مقارنة نتيجة مقارنة مع عدد تكون نتائج غير صحيحة

وهذا توضيح لما سيجري من عمليات مع هذا الشرط

$$a > 5 > 3$$

إذا كان قيمة المتغير (a=6) فيكون الشرط بالشكل التالي

$$6 > 5 > 3$$

بما أن الأسبقيات متساوية سيبدأ بتنفيذ العمليات من اليسار إلى اليمين
فيقارن هل 6 أكبر من 5 نعم نتيجة المقارنة هي (True) أي واحد

1.

$$1 > 3$$

سيتحقق من الشرط التالي هل الواحد أكبر من ثلاثة كلا فتكون نتيجة هي (false) أي صفر

2.

$$1 > 3 = 0$$

لذلك فالحل الصحيح بما إننا نريد تحقق الشرطان معا نستخدم بينهم عملية and كأن نقول يجب أن تكون قيمة (a) أكبر من (3) و أكبر من (5).

```
int a=6;
if ( ( a>5) && (a>3) )
```

لو رسمنا المخطط التوضيحي للمثال بعد التعديل يكون بشكل التالي

$$(a > 5) \&\& (a > 3)$$

إذا كان قيمة المتغير (a=6) فيكون الشرط بالشكل التالي

$$(6 > 5) \&\& (6 > 3)$$

سيبدأ أولاً بتنفيذ ما بين الأقواس بما أن الأسبقيات متساوية سيبدأ
بتنفيذ العمليات من اليسار إلى اليمين فيقارن هل 6 أكبر من 5 نعم
نتيجة المقارنة هي (True) أي واحد

1.

$$1 \&\& (6 > 3)$$

سيتحقق من الشرط التالي هل الستة أكبر من ثلاثة كلا فتكون نتيجة
هي (True) أي واحد

2.

$$1 \&\& 1 = 1$$

عملية (and) بين واحد و واحد هي واحد فالنتيجة هي (True)

مثال ٤: بين بعد تتبع هذا البرنامج ما هي قيم (a, b) التي ستطبع على شاشة التنفيذ ؟

البرمجة بلغة ++C	البرمجة بلغة C
<pre>#include<iostream.h> main() { 1.int a=5; 2.int b=7; If (b %2==1) { 3.a=a+6; 4.b=b+4 } 5.a=a+3; 6.b=b+2 7.cout<<"a="<<a<<"\t b="<<b;}</pre>	<pre>#include<stdio.h> main() { 1.int a=5; 2.int b=7; If (b %2==1) { 3.a=a+6; 4.b=b+4 } 5.a=a+3; 6.b=b+2 7.printf("a=%d \t b=%d",a, b);}</pre>

تتبع خطوات البرنامج:

١. خطوة رقم (١) أصبح قيمة (a=5)
٢. خطوة رقم (٢) أصبح قيمة (b=7) وقيمة (a) بقيت ثابتة لم تتغير محتفظة بقيمتها في الخطوة السابقة (a=5)
٣. قبل خطوة رقم (٣) يوجد شرط لتنفيذ ما بين قوسي العبارة الشرطية وهو أن يكون باقي قسمة قيمة المتغير b على (٢) تساوي واحد أن يكون رقم فردي وفعلا باقي قسمته (٧) على (٢) هو واحد لذلك سينفذ الخطوة رقم (٣ و٤) وتكون المقارنة بشكل التالي

$$(b\%2==1) \rightarrow (7\%2==1) \rightarrow (1==1) \rightarrow \text{True}$$

- في الخطوة رقم (٣) أصبح قيمة (a) هي (a=5+6=11) وقيمة (b) بقيت ثابتة لم تتغير (b=7)
- خطوة رقم (٤) أصبح قيمة (b) هي (b=7+4=11) وقيمة (a) بقيت ثابتة لم تتغير (a=11)
- ٤. خطوة رقم (٥) أصبح قيمة (a) هي (a=11+3=14) وقيمة (b) بقيت ثابتة لم تتغير (b=11)
- ٥. خطوة رقم (٦) أصبح قيمة (b) هي (b=11+2=13) وقيمة (a) بقيت ثابتة لم تتغير (a=14)

والنتيجة على شاشة التنفيذ هي

a=14 b=13

مثال ٥: نفس المثال السابق فقط غير قيمة (b) في الخطوة (٢) إلى رقم (٤) إي (b=4) ولنتتبع البرنامج الجديد ؟

البرمجة بلغة	c	البرمجة بلغة	c++
	<pre>#include<stdio.h> main() { 1.int a=5; 2.int b=4; If (b %2==1) { 3.a=a+6; 4.b=b+4 } 5.a=a+3; 6.b=b+2 7.printf("a=%d \t b=%d",a, b);}</pre>		<pre>#include<iostream.h> main() { 1.int a=5; 2.int b=4; If (b %2==1) { 3.a=a+6; 4.b=b+4 } 5.a=a+3; 6.b=b+2 7.cout<<"a="<<a<<"\t b="<<b;}</pre>

١. خطوة رقم (١) أصبح قيمة (a=5)

٢. خطوة رقم (٢) أصبح قيمة (b=4) وقيمة (a) بقيت ثابتة لم تتغير محتفظة بقيمتها في الخطوة السابقة (a=5)

٣. قبل الخطوة رقم (٣) يوجد شرط لتنفيذ ما بين قوسي العبارة الشرطية وهو أن يكون باقي قسمة قيمة المتغير b على (٢) تساوي واحد أن يكون رقم فردي وان باقي قسمته على (٢) هو صفر لذلك سوف لا ينفذ الخطوة رقم (٣ و٤) وينتقل إلى خطوة رقم (٥) وتكون المقارنة بشكل التالي

$(b\%2==1) \rightarrow (4\%2==1) \rightarrow (0==1) \rightarrow \text{False}$

٤. خطوة رقم (٥) أصبح قيمة (a) هي (a=5+3=8) وقيمة (b) بقيت ثابتة لم تتغير (b=4)

٥. خطوة رقم (٦) أصبح قيمة (b) هي (b=4+2=6) وقيمة (a) بقيت ثابتة لم تتغير (a=8)

والنتيجة على شاشة التنفيذ هي

a=8 b=6

في لغة (C++) يمكنك من تعريف متغيرات أينما تشاء في البرنامج فلو عرفنا متغير بين قوسي العبارة الشرطية يكون غير معرف بالنسبة لمن هم خارج قوسي العبارة الشرطية فقط معرف بالداخل

الكود

If (True>3)

{int x=5;}

x=3; // سوف يعترض المترجم على هذا السطر ويعتبر (x) غير معرف

إي أن المتغير ننهي حياته عند الخروج من العبارة الشرطية

٢. عبارة (if--else) الشرطية:

هي عبارة شرطية مكونة من جزئيين من الاكواد البرمجية الذي نريده أن ينفذ بتحقق شرط (condition) نضعه داخل قوسي (if) والذي نريده أن ينفذ في حالة عدم تحقق الشرط نضعه بين قوسي (else)

شكل عبارة if--else الشرطية

```
If ( condition )
{
Statement1
}
else
{
Statement2
}
```

- ✓ إي بتوضيح أكثر إذا تحقق (condition) ينفذ Statement1 ، وإذا لم يتحقق الشرط سينفذ تلقائياً Statement2
- ✓ إي إما ينفذ الأسطر البرمجية بين قوسي (if) أو ينفذ الأسطر البرمجية بين قوسي (else)

للتوضيح إذا كان الشرط هو

شكل عبارة if--else الشرطية

```
If ( input_Numner%2==0 )
{
Statement1
}
else
{
Statement2
}
```

- ومن الشرط الموجود ((If (input_Numner%2==0) إي متى ما كان (input_Numner) باقي قسمته على (٢) هو صفر سوف ينفذ Statement1
- وخلافه إي إذا لم يتحقق الشرط إي إذا كان (input_Numner) باقي قسمته على (٢) لا يساوي صفر سوف ينفذ Statement2
- ✓ (Statement1 ، Statement2) قد تكون سطر برمجي واحد أو أكثر من سطر فتوضع بين قوسين

مثال توضيحي عن خطوات سير البرنامج في وجود عبارة شرطية (if—else). وهو برنامج تدخل رقم من شاشة التنفيذ ويبين لك هل الرقم موجب أم سالب (إي هل هو اكبر من الصفر أو اصغر منه)

البرمجة بلغة c++	البرمجة بلغة c
<pre>#include<iostream.h> main() {1.int a; 2.cin>>a ; If (a>0) 3. printf("is greater than 1 is positive"); else 4. printf("is less than 1 is negative"); 5. a=a+6; 6.}</pre>	<pre>#include<stdio.h> main() {1.int a; 2 scanf("%d",&a); If (a>0) 3. printf(" is greater than 1 is positive "); else 4. printf(" is less than 1 is negative "); 5. a=a+6; 6.}</pre>

تتبع خطوات البرنامج:-

١. خطوة رقم (١) تم تعريف متغير (a) من نوع integer

٢. خطوة رقم (٢) يطلب من المستخدم إدخال قيمة للمتغير (a) من شاشة التنفيذ

٣. الآن نأتي إلى شرط التحقق

مهم

✓ إذا ادخل المستخدم إي رقم اكبر من صفر سوف يحقق شرط (if) و ينفذ السطر (٣) ويطبع هذه الرسالة

is greater than 1 is positive

ويطفر السطر (٤) (لأنه يتحقق إذا لم ينفذ الشرط) تم ينفذ السطر (٥) تم السطر (٦) إي يكون تسلسل تنفيذ خطوات البرنامج الكلية

1 → 2 → 3 → 5 → 6

✓ إذا ادخل المستخدم إي رقم اصغر من صفر سوف لا يحقق شرط (if) ولذلك ينتقل إلى (else) و ينفذ السطر (٤) ويطبع هذه الرسالة في شاشة التنفيذ

is less than 1 is negative

تم ينفذ السطر (٥) تم السطر (٦) إي يكون تسلسل تنفيذ خطوات البرنامج الكلية

1 → 2 → 4 → 5 → 6



العبارات الشرطية المتداخلة:-

بعد أن تعرفنا عن (if-else) وعبارة (if) الشرطية. في بعض البرامج قد نحتاج إلى النوعان معا بشكل متداخل أو منفصل حسب الحاجة ويبقى لكل عبارة طريقتها نفسها في المعالجة ولكنها تدخل ضمن عبارة أخرى (أي لا تنفذ إلا بتنفيذ العبارة إلام) كهذا المثال حيث وقعت عبارة (if-else) ضمن عبارة (if) ووضعناها بين أقواس لكي تبين أنها تابعة إلى (if) إلام.

```

الكود
if (input_Try > 0)
{
if (input_Try > 5)
1.cout<<"the number is greater than 5";
else
2.out<<" the number is less than 5";
}
3.

```

- وان عبارة (if-else) لا تنفذ مطلقا ولا يمر عليها المترجم إلا بتحقيق شرط العبارة الشرطية إلام وهي (if (input_Try > 0) إي أن يكون الرقم المدخل (input_Try) اكبر من صفر
 - ✓ إذا كان الرقم المدخل فعلا اكبر من صفر سوف يمر على عبارة (if-else) الداخلية وإما ينفذ الخطوة رقم (1) أو الخطوة رقم (2)
 - ✓ وإذا لم يكن اكبر من صفر لا يمر أصلا على عبارة (if-else) الداخلية ويتجه نحو الخطوة (3) لينفذها
- لان عبارة (if-else) الداخلية واقعة ضمن عبارة الشرطية (if (input_Try > 0) وبما أن شرطها لم يتحقق لذلك سوف لا ينفذ ما هو موجود داخل قوسيتها



• إذا جاء لدينا أكثر من (if) وأكثر من (else) فكيف نعرف أن هذه (if-else) تنفي العبارة الشرطية (if) الأولى أم الثانية أم غيرهما..؟

بكل سهولة كل (else) تنفي اقرب عبارة (if) الشرطية عليها.

مثال: مقارنة بين مثالين احدهما يستخدم العبارة المتداخلة وآخر البوابات لتتحقق شرط أن يكون الرقم يقبل القسمة على ثلاثة ولا يقبل القسمة على تسعة وشرط آخر أن يقبل القسمة على ثلاثة ويقبل القسمة على تسعة .؟

الكود بلغة ++C بدون العبارات الشرطية المتداخلة	الكود بلغة ++C باستخدام العبارات الشرطية المتداخلة
<pre>#include<iostream.h> main() { int Number=80; If ((Number % 3==0)&&(Number % 9 !=0)) cout<<"the number is donot accept mod to 9"; If ((Number % 3==0)&&(Number % 9 ==0)) cout<<" the number is accept mod to 9"; }</pre>	<pre>#include<iostream.h> main() { int Number=80; If ((Number % 3==0) { if(Number % 9 !=0) cout<<"the number is donot accept mod to 9"; else cout<<" the number is accept mod to 9"; } }</pre>

الكود بلغة C بدون العبارات الشرطية المتداخلة	الكود بلغة C باستخدام العبارات الشرطية المتداخلة
<pre>#include<stdio.h> main() { int Number=80; If ((Number % 3==0)&&(Number % 9 !=0)) printf("the number is donot accept mod to 9"); If ((Number % 3==0)&&(Number % 9 ==0)) printf(" the number is accept mod to 9"); }</pre>	<pre>#include<stdio.h> main() { int Number=80; If ((Number % 3==0) { if(Number % 9 !=0) printf("the number is donot accept mod to 9"); else printf(" the number is accept mod to 9"); } }</pre>

لو تلاحظ أن المثالان يعطيان نفس النتيجة لكن باستخدام العبارات المتداخلة تكون الاكواد البرمجية أكثر سهولة ووضوح للمبرمج. وفي بعض الأحيان مع بعض الأسئلة لا نستطيع استخدام الطريقة الأولى لأنك تجد نفسك محكوما باستخدام الطريقة الثانية لان بها مرونة أكثر



وإذا وقع عبارتي (if) متتاليتين ويليهما عبارتي ل (else) متتاليتين فستنفي ال (else) الأولى اقرب عبارة (if) عليها إي عبارة (if) الثانية وستنفي ال (else) الثانية اقرب عبارة (if) عليها وبما أن عبارة (if) الثانية نفتها (else) الأولى فسيكون عبارة (if) الأولى هي تخص ال (else) الثانية شاهد المثال لفهم الشرح.

مثال: تتبع خطوات الحل في البرنامج التالي إذا كان الرقم المدخل من شاشة التنفيذ (input_Try=4 , input_Try=7 , input_Try=-3) وطلب المثال هو طباعة هل الأعداد فردية أو زوجية لكن فقط الأرقام الموجبة

```

C++ الكود بلغة
#include<iostream.h>
Main()
{
1.int input_Try;
2.cin>> input_Try;
if (input_Try > 0) ← نلاحظ أن (else) الأخيرة تكون مخالفة لل (if) الأولى
if (input_Try %2==0)
3.cout<<"the number is positive even";
else
4.cout<<" the number is positive Odd";
else ←
5.cout<<"the number is less than zero";
}

```

```

C الكود بلغة
#include<stdio.h>
Main()
{
1.int input_Try;
2 scanf("%d", input_Try);
if (input_Try>0) ← نلاحظ أن (else) الأخيرة تكون مخالفة لل (if) الأولى
if (input_Try %2==0)
3.printf(" the number is positive even ");
else
4. printf(" the number is positive Odd ");
else ←
5. printf("the number is less than zero")
}

```

توضيح الخطوات:

١. خطوة رقم (١) تم تعريف متغير اسمه input_Try
٢. خطوة رقم (٢) تم طلب إدخال قيمة للمتغير من شاشة التنفيذ ثم يتحقق من الشروط التالية
- ✓ إذا كان الرقم المدخل من شاشة التنفيذ input_Try= -3

سيتم التحقق من الشرط (if (input_Try>0) وان قيمة المتغير input_Try هي (-3) أي اقل من صفر أي لم يحقق الشرط لذلك سيتجه إلى عبارة (else) الثانية وينفذ الخطوة رقم (٥) ويطبع في شاشة التنفيذ

the number is less than zero

لماذا لم ينفذ خطوة رقم (٣) وخطوة رقم (٤) ؟..

كما قلنا سابقا كل عبارة شرطية (if) يتبعها سطر واحد إذا لم نضع أقواس وإذا أكثر من سطر يتبع العبارة الشرطية التي نضع أقواس حول الذي يتبعها وبما انه هنا لا يوجد أقواس معناه الذي سيتبع العبارة الشرطية `if (input_Try>0)` هو فقط السطر الذي يليه وهو `if (input_Try %2==0)` وهذا السطر أيضا يتبعه سطر واحد ويوجد عبارة `else` له إي (كأننا نقول إذا تحقق الشرط الأول وكان الرقم اكبر من صفر توجد عبارة شرطية تتحقق منه إذا كان الرقم فردي أم زوجي لكن الموجب فقط يمر على عبارة (if—else) لأنها قيدت بالشرط الذي قبلها)

✓ إذا كان الرقم المدخل من شاشة التنفيذ `input_Try=7`

سيتم التحقق من الشرط `if (input_Try>0)` وان قيمة المتغير `input_Try` هو (٧) إي اكبر من صفر إي حقق الشرط لذلك سيتجه إلى عبارة `if (input_Try %2==0)` التي تلي شرط التحقق وهذه العبارة تتحقق في ما إذا كان الرقم زوجي وفعلا الرقم ٧ هو زوجي لذلك وينفذ الخطوة رقم (٣) ويطبع في شاشة التنفيذ

the number is positive even

ولا ينفذ الخطوة (٤) لان الشرط `if (input_Try %2==0)` تحقق وأيضا لا ينفذ الخطوة رقم (٥) لان شرط `if (input_Try>0)` تحقق

✓ إذا كان الرقم المدخل من شاشة التنفيذ `input_Try=4`

سيتم التحقق من الشرط `if (input_Try>0)` وان قيمة المتغير `input_Try` هي (٤) أي اكبر من صفر أي حقق الشرط لذلك سيتجه إلى عبارة `if (input_Try %2==0)` التي تلي شرط التحقق وهذه العبارة تتحقق في ما إذا كان الرقم زوجي وان الرقم ٤ هو رقم فردي لذلك لا يتحقق الشرط وينفذ عبارة `else` وينفذ الخطوة رقم (٤) ويطبع في شاشة التنفيذ

the number is positive Odd

لا ينفذ الخطوة رقم (٥) لان شرط `if (input_Try>0)` تحقق

٢. عبارة (if—else if) الشرطية:

هي مجموعة عبارات شرطية متخالفة في شروطها . أي تكون واحدة مخالفة إلى الأخرى في شرطها . ويتم التحقق من الشروط وقت التنفيذ إذا لم يتحقق الشرط الأول ينتقل إلى (else if) الثانية وإذا لم تتحقق ينتقل إلى الثالثة حتى أخيرا يصل إلى الشرط الذي يتحقق وإذا تحقق واحد من (if) سوف ينفذ ما في داخلها ويهمل البقية .

الشكل العام

شكل عبارة if—else if الشرطية

```
If ( condition1 )
{
Statement1
}
else if( condition2   عدد   else if   يكون غير محدد يحددها المستخدم حسب حاجته
)//
{
Statement2
}
Else if( condition3 )
{
Statement3
}
else // نستطيع أن نضع else أو نحذفها إذا لم نحتاج إليها
{
Statement4
}
خطوات برمجية أخرى // خطوة جديدة
```

مهم

- ✓ إذا تحقق condition1 سوف ينفذ Statement1 وينتقل بعدها إلى "خطوة جديدة"
- ✓ إذا لم يتحقق condition1 سوف ينتقل إلى condition2 وإذا تحقق الشرط سوف ينفذ Statement2 بعدها إلى "خطوة جديدة"
- ✓ إذا لم يتحقق condition1 سوف ينتقل إلى condition2 وإذا تحقق الشرط سوف ينفذ Statement2 بعدها إلى "خطوة جديدة"
- ✓ إذا لم يتحقق condition1 سوف ينتقل إلى condition2 وإذا لم يتحقق condition2 سوف ينتقل إلى condition3 وإذا تحقق الشرط سينفذ Statement3 بعدها إلى "خطوة جديدة"
- ✓ وإذا لم يتحقق إي من conditions سوف ينفذ ما موجود في else وبعدها ينتقل إلى "خطوة جديدة"

مثال: برنامج تدخل رقم وبيبين لكل هل الرقم (يقبل القسمة على ٣ أم على ٥ أم على ٧ أم غير ذلك) ويطبع رسالة في كل حالة) في حال إذا قبل القسمة على ٧ يجمع مع الرقم المدخل قيمة ٢ ويطبعه

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> main() { 1.int number_enter; 2.cin>> number_enter; if (number_enter % 3==0) 3.cout<<"Acept Devide to 3" ; else if (number_enter % 5==0) 4. cout<<" Acept Devide to 5" ; else if (number_enter % 7==0) { 5.number_enter= number_enter+2; 6. cout<<" Acept Devide to 7=" << number_enter; } else 7. cout<<"Donot True any of conditions" ; }</pre>	<pre>#include<stdio.h> main() { 1.int number_enter; 2 scanf("%d",& number_enter); if (number_enter % 3==0) 3.printf("Acept Devide to 3"); else if (number_enter % 5==0) 4.printf(" Acept Devide to 5"); else if (number_enter % 7==0) { 5.number_enter= number_enter+2; 6.printf(" Acept Devide to 7=%d", number_enter); } else 7.printf("Donot True any of conditions"); }</pre>

تتبع خطوات البرنامج مع التوضيح:

١. خطوة رقم (١) تنفذ بشكل اعتيادي

٢. خطوة رقم (٢) تنفذ بشكل اعتيادي ويطلب من المستخدم إدخال قيمة للمتغير (number_enter) من شاشة التنفيذ لنفرض إننا أدخلنا (number_enter) مرة (٦ و٤ و١٥ و٢٠ و١٩)

لنتتبع القيم المدخلة

✓ إذا كان number_enter=6

سوف ينفذ الخطوة رقم (٣) وينتهي البرنامج (حسب شرح عن الدالة (if—else if) إذا تحقق احد الشروط في احد العبارات تنفذ ما في داخله وتتجه إلى ما بعد else النهائية لتنفذه وبما انه لا يوجد شي بعد else يوجد فقط قوس نهاية البرنامج لذلك ستنهي البرنامج) طابعا للرسالة التالية تم وينتهي البرنامج

Accept Devide to 3

✓ إذا كان number_enter=14

سوف ينفذ الخطوة رقم (5) وخطوة رقم (6) طابعا للرسالة التالية تم وينتهي البرنامج

Accept Devide to 7=9



number_enter=15 إذا كان ✓

سوف ينفذ الخطوة رقم (3) ولا ينفذ خطوة رقم (٤) رغم قبوله القسمة على (٥) لأنه برمجيا عندما يبدأ البرنامج بالتحقق من الشروط يبدأ بالتحقق بالتسلسل خطوة بخطوة حسب الترتيب ونحن واضعين شرط قبول القسمة على ثلاثة قبل شرط قبول القسمة على خمسة وعندما يمر البرنامج على الشرط الأول وحققه لا يمر على الشرط الثاني بل يتجه إلى ما بعد (else). لو كنا واضعين شرط قبول القسمة على خمسة قبل شرط قبول القسمة على ثلاثة لتنفيذ خطوة رقم (٤). لذلك سيطبع البرنامج الآن هذه الرسالة

Accept Devide to 3

number_enter=20 إذا كان ✓

سوف ينفذ الخطوة رقم (٤) طابعا للرسالة التالية تم وينتهي البرنامج

Accept Devide to 5

number_enter=19 إذا كان ✓

سوف لن يحقق أي من الشروط السابقة لذلك سيتجه إلى (else) ينفذ الخطوة رقم (٧) طابعا للرسالة التالية تم وينتهي البرنامج

Donot True any of conditions

✘ خطوة رقم (٥ و ٦) وضعت بين قوسين للدلالة على أنهما تابعتان لعبارة (else)

مثال: برنامج يقوم بطباعة أيام الأسبوع بحيث إذا ضغطنا (١) يظهر يوم الأحد في شاشة التنفيذ والبقية بالتسلسل إلى (٧) يمثل السبت ؟.

تحليل المتطلبات: بما انه يوجد لدينا سبعة أيام فوضع كل يوم على حدة بعبارة شرطية يعقد البرنامج شيئاً ما لئلا سنستخدم العبارة الشرطية (if—else if) لتسهيل الأمر قليلاً شاهد المثال بسيط جداً

C++	الكود بلغة
<pre>#include<iostream.h> main() {int Day_Number; cin>> Day_Number ; if (Day_Number==1) cout<<"sunday" ; else if (Day_Number==2) cout<<"monday" ; else if (Day_Number==3) cout<<"Tuesday" ; else if (Day_Number==4) cout<<"wednesday" ; else if (Day_Number==5) cout<<"thursday" ; else if (Day_Number==6) cout<<"friday" ; else if (Day_Number==7) cout<<"saturday" ; else cout<<"error" ; }</pre>	

C	الكود بلغة
<pre>#include<stdio.h> main() {int Day_Number; scanf("%d",& Day_Number); if (Day_Number==1) printf("sunday"); else if (Day_Number==2) printf("monday"); else if (Day_Number==3) printf("Tuesday"); else if (Day_Number==4) printf("wednesday"); else if (Day_Number==5) printf("thursday"); else if (Day_Number==6) printf("friday"); else if (Day_Number==7) printf("saturday"); else printf("error"); }</pre>	

شاهد شاشة التنفيذ عندما أدخلنا الرقم (٥) ظهر اليوم
المقابل له وهو (Thursday)

عبارة (Switch—Case) الشرطية:

هي مجموعة عبارات شرطية (Case) ويقارن القيمة عند كل (Case) مع المتغير في (switch) . ويتم التحقق من الشروط وقت التنفيذ فإذا لم يتحقق الشرط الأول ينتقل إلى (Case) الثانية فإذا لم يتحقق الشرط الثاني ينتقل إلى (Case) الثالثة حتى أخيرا يصل إلى الشرط الذي يتحقق وإذا تحقق واحد من (Case) سوف ينفذ ما في داخلها وبهمل البقية وإذا لم ينفذ إي واحد منهن سوف يتجه لينفذ ما في داخل (default). وتكون مشابه كثيرا جدا (if else if)

الشكل العام

شكل عبارة if--elseif الشرطية

```
Switch(Truth of Case )
{ Case condition1: //حاحته
{
Statement1
} Break;
Case condition2
{
Statement2
} Break;
Case condition3:
{
Statement3
} Break;
default: // نستطيع أن نضع default أو نحذفها إذا لم نحتاجها
{ Statement4 }
}
// خطوات برمجية أخرى // خطوة جديدة
```

مهم

(Truth of Case): هو المتغير أو عملية رياضية الذي ينتج عنها قيمة معينة تتم مقارنه هذه القيمة مع كل condition موجودة في كل Case وأيها يحقق الشرط ينفذ البرنامج (Statement) الخاص به. شاهد هذا التوضيح في المثال لتوضيح الصورة إذا كان لدينا متغير اسمه Number وتريد لينفذ احد الشروط أن يكون قيمته (٥١) نكتب هكذا في جمل شرطية

الكود

```
if ( Day_Number==51)
//do some thing
```

في حالة Switch--Case نكتبها هكذا

الكود

```
Switch(Day_Number)
Case 51: //do some thing
Break;
```

✓ حال (Switch—Case) كحال الجمل الشرطية إي إذا كنا نقارن مع حرف نضعه هكذا

الكود

```
Switch(Capatat_symbol)
Case 'a': //do some thing
Break;
```

حيث أن (do some thing) هي أي عدد من الاكواد البرمجية يمكن كتابتها وحسب الحاجة وتذكر إذا أردنا أن نكتب أكثر من سطر برمجي داخل (Case) يجب أن نضعها داخل قوسين وبعدها نكتب (Break).

- إي أن إذا تحقق condition1 سوف ينفذ Statement1 وينتقل بعدها إلى "خطوة جديدة"
- إذا لم يتحقق condition1 سوف ينتقل إلى condition2 وإذا تحقق الشرط سوف ينفذ Statement2 بعدها إلى "خطوة جديدة"
- وإذا لم يتحقق إي من conditions سوف ينفذ ما موجود في default وبعدها ينتقل إلى "خطوة جديدة"
- نلاحظ وجود (Break) في نهاية كل (case) هذا شيء ثابت في هيكلية هذه الدالة لكي يخرج من (switch) بعد تحقق احد الشروط

مثال: لدينا المعومات التالية عن هؤلاء الأشخاص نريد مجرد كتابة أول حرف من اسم الشخص يعطيك المعلومات الكاملة عنه

1.Ali: his names Ali kammel,20 Year old, third stage

1.Salem: his names Salem kammel,18 Year old, third stage

1.Hussien: his names Hussien Ahmmed Taleb,21 Year old, third stage eng.Computer

C++

الكود بلغة

```
#include<iostream.h>
main()
{1.char index_of_Name;
2.Cout<<"Enter First Chat of Student name: ";
3.Cin>> index_of_Name ;
4.switch( index_of_Name ) {
5.case 'A':
6.Cout<<" his names Ali kammel,20 Year old, third stage " ;
7.break;
8.case 'S':
9.Cout<<" his names Salem kammel,18 Year old, third stage " ;
10.break;
11.case 'H':
12.Cout<<" his names Hussien Ahmmed Taleb,21 Year old, third stage eng.Computer " ;
13.break;
14.default:
15.Cout<<"You Not have saved names in this index" ; }}
```

```

#include<stdio.h>
main()
{1.char index_of_Name;
2.printf("Enter First Chat of Student name: ");
3 scanf("%c", index_of_Name) ;
4.switch( index_of_Name ) {
5.case 'A':
6. printf(" his names Ali kammel,20 Year old, third stage ") ;
7.break;
8.case 'S':
9. printf(" his names Salem kammel,18 Year old, third stage ") ;
10.break;
11.case 'H':
12. printf(" his names Hussien Ahmmed Taleb,21 Year old, third stage eng.Computer ") ;
13.break;
14.default:
15. printf("You Not have saved names in this index") ; }}

```

١. خطوة رقم (١) عرفنا متغير اسمه (index_of_Name) من نوع حرفي

٢. خطوة رقم (٢) طبع رسالة للمستخدم تطلب منه إدخال أول حرف من اسم الشخص هذه الرسائل مهمة جدا لواجه برنامجك حتى يعلم المستخدم كيف يتعامل مع برنامجك ما هو المطلوب منه وماذا يدخل فدائما حاول أن يكون برنامجك واضح للمستخدم بهذه الرسائل

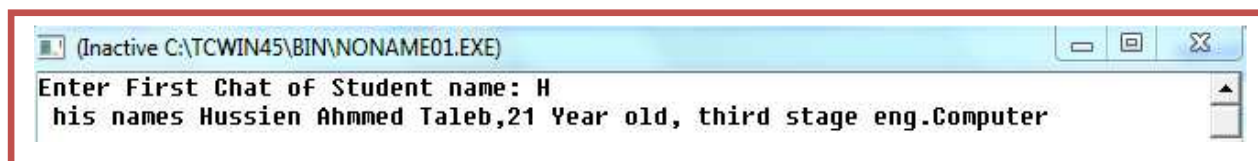
٣. خطوة رقم (٣) قمنا بقراءة حرف من شاشة التنفيذ ، خطوة (٤) أدخلنا الحرف الذي ادخله المستخدم في قائمة الخيارات وهي (switch)

٤. خطوة رقم (٥) تحوي حالة شرطية لأول حرف من اسم الشخص (Ail) وتلاحظ الأحرف في المقارنة توضع بين علامة تنصيصية واحدة من الجانبين هكذا ('A')

٥. خطوة رقم (٦) تطبع سجل هذا الشخص إذا كان هو المطلوب ، وخطوة رقم (٧) توقف ال case لهذه الحالة

وبقية الخطوات نفس الشيء

لاحظ عندما أدخلنا حرف (H) ماذا ظهر في شاشة التنفيذ



```

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
Enter First Chat of Student name: H
his names Hussien Ahmmed Taleb,21 Year old, third stage eng.Computer

```

مثال: نفس المثال السابق في موضوع (if –else if) الخاص بطباعة أيام الأسبوع حسب الرقم نحله باستخدام (switch–case) قارنه بالمثال السابق وشاهد الفرق

C++

الكود بلغة

```
#include<iostream.h>
main()
{int Day_Number ;
Cin>> Day_Number ;
switch( Day_Number ) {
case 1:cout<<"sunday" ;
break;
case 2: cout<<"monday" ;
break;
case 3: cout<<"Tuesday" ;
break;
case 4: cout<<"wednesday" ;
break;
case 5: cout<<"thursday" ;
break;
case 6: cout<<"fridaay" ;
break;
case 7: cout<<"saturday" ;
break;
default: cout<<"error" ; }
}
```

C

الكود بلغة

```
#include<stdio.h>
main()
{int Day_Number ;
scanf("%d",& Day_Number );
switch( Day_Number ) {
case 1: printf("sunday");
break;
case 2:printf("monday");
break;
case 3:printf("Tuesday");
break;
case 4:printf("wednesday");
break;
case 5:printf("thursday");
break;
case 6:printf("fridaay");
break;
case 7:printf("saturday");
break;
default: printf("error"); }
}
```

شاهد شاشة التنفيذ عندما أدخلنا الرقم (٥) ظهر اليوم المقابل له وهو(Thursday)





وضع عبارات شرطية داخل جمل Switch !..

يمكن وضع عبارات (if) الشرطية بمختلف أنواعها داخل كل case حالها كحال إي كود برمجي آخر على سبيل المثال: برنامج تدخل رقم من شاشة التنفيذ ثم يطلب من المستخدم إدخال رقم العملية التي يريد أن يؤديها على هذه الرقم إذا ادخل المستخدم رقم واحد سوف يبين له هل الرقم فردي أم زوجي وإذا ادخل رقم اثنان يبين له هل الرقم اكبر من صفر أو اصغر منه

C++

البرمجة بلغة

```
#include<iostream.h>
main()
{int    Number,Chose_check ;
Cout<<"enter your number: ";
Cin>> Number ;
Cout<<"enter Your Selected check (1) or 2:" ;
Cin>> Chose_check ;
switch( Chose_check ) {
case 1:{
if ( Number % 2== 0)
cout<<"the number is even";
else
cout<<"the number is odd";}
break;
case 2:{
if ( Number >0)
cout<<"the number is more than zero";
else
cout<<"the number is less than zero";}
break;
default: cout<<"Error Choice";}}
```

C

البرمجة بلغة

```
#include<stdio.h>
main()
{int    Number,Chose_check ;
Printf("enter your number: ");
Scanf("%d",&Number) ;
Printf("enter Your Selected check (1) or 2:") ;
Scanf("%d", &Chose_check) ;
switch( Chose_check ) {
case 1:{
if ( Number % 2== 0)
Printf("the number is even");
else
Printf("the number is odd");}
break;
case 2:{
if ( Number >0)
Printf("the number is more than zero");
else
Printf("the number is less than zero");}
break;
default: Printf("Error Choice");}}
```

شاهد شاشة التنفيذ

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter your number: 33
enter Your Selected check (1) or 2:1
the number is odd
```

الفصل الثالث

عبارات أو جمل التكرار (loop)

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصل الأول والثاني وفهما كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على دوال التكرار وطريقة استخدامها

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب : أربع ساعات

١. عبارة (For--Loop) التكرارية الاعتيادية:-

في الفصلين السابقين علمنا أن البرنامج يبدأ بالتنفيذ خطوة خطوة دون تكرار إي خطوة إلى أن يصل إلى نهاية البرنامج و عملنا في حال وجود جمل (if) الشرطية في البرنامج قد يكون بسببها هناك استثناء بعض الخطوات من التنفيذ إذا لم يتحقق الشرط. والسؤال هنا ماذا لو أردنا تكرار خطوة أو أكثر من خطوة لأكثر من مرة لسبب ما كيف نعمل ذلك وهذا ما تؤديه الدوال التكرارية ومنها عبارة (For) هي عبارة تكرارية تستخدم لتكرار خطوة في حالة لم نضع أقواس أو مجموعة خطوات في حال حصرها بين قوسي لعدد معين من المرات يحددها المبرمج. وشكلها العام

هيكلية For--Loop

```
for (Initializing; Boolean_Expression; Update)
{
statement
}
```

- (Initializing): هي القيمة البداية المعطاة للمتغير التي منها سيبدأ العد. (مثلا $i=0$)
- (Boolean_Expression): هو شرط التوقف الذي عندما تصبح قيمة المتغير غير محققة لشرط التوقف سوف يخرج من عبارة (for) بمعنى آخر مادام نتيجة اختبار ال (Boolean_Expression) هي (True) العبارة التكرارية تستمر بالتكرار ومتى أصبح (False) يخرج من العبارة التكرارية. مثلا ($i < 5$) هو شرط التوقف ويبدأ العد من الواحد سيتوقف إذا أصبح ($i=5$) لأنه نتيجة المقارنة (False)
- (Update): هي مقدار الزيادة أو النقصان في قيمة المتغير في كل دورة (loop). مثلا ($i++$) أو ($i--$) ولا يختلف إذا كان الشرط ($i++$ or $++i$, $--i$ or $i--$) لأنه في كل الحالات لا تزداد قيمة العداد إلا بعد تنفيذ الجمل بين قوسي العبارة التكرارية مؤثرا على شرط التكرار في الدورة الجديدة التي تليه
- (statement): هي الخطوات البرمجية التي ستنفذ عدد من المرات. إذا كانت عبارة عن خطوة برمجية واحدة فليس بحاجة لوضعها داخل أقواس وإذا كانت أكثر من خطوة يجب وضعها داخل أقواس.

مهم

كيف تعمل عبارة (For) التكرارية.....؟

أن المتغير يبدأ بقيمة بدائية (Initializing) ويستمر بالزيادة أو النقصان حسب (Update) إي قد يزداد أو ينقص بمقدار واحد أو أكثر من واحد ومتى ما أصبح قيمة المتغير غير محققة للشرط (Boolean_Expression) يخرج من العبارة التكرارية إلى الخطوات البرمجية التي تليه. وإذا كانت محققة للشرط ينفذ الخطوات البرمجية التي داخل العبارة التكرارية

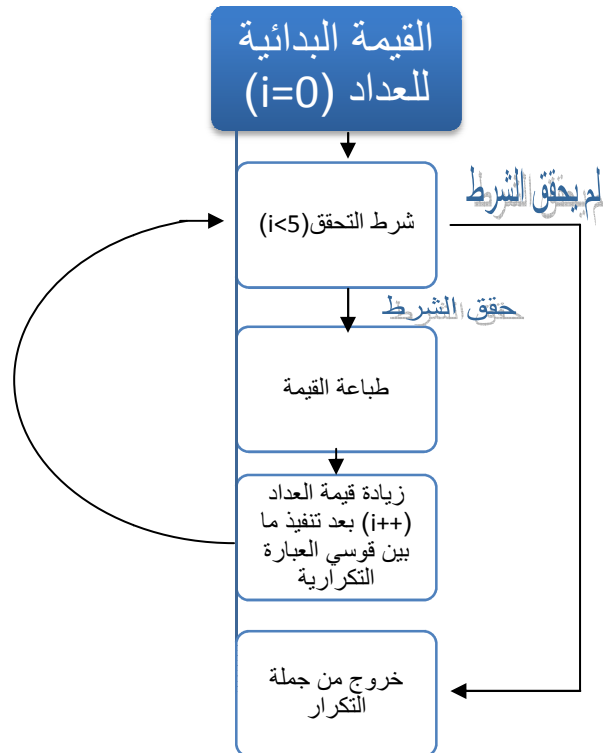
✓ لإدخال (for) في دارة لانهاية نكتب for(;;) فقط .

مثال: لو أردنا عداد يعد من (0) إلى (4) إي يطبع على شاشة التنفيذ من (0 إلى 4) ؟.

البرمجة بلغة c	البرمجة بلغة c++
<pre>#Include<stdio.h> main() {1.int i; 2.for (i=0; i<5; i++) 3.printf("%d\t",i); }</pre>	<pre>#include<iostream.h> main() {1.int i; 2.for (i=0; i<5; i++) 3.cout<<i<<"\t"; }</pre>

توضيح الخطوات :

- خطوة رقم (١) عرفنا متغير (i) من نوع integer
- خطوة رقم (٢) هي عبارة تكرارية بما إننا نريد العد من الصفر فأعطينا القيم البدائية صفر .والعد يصل إلى ٤ معناه شرط التوقف أن يكون أقل من خمسة بما إننا نريد أن يعد خمس عدات بين صفر والأربعة لذلك يجب أن يكون مقدار الزيادة واحد ليعد (0,1,2,3,4) .لان لو جعلنا مقدار الزيادة ٢ مثلا فسيعد العداد ثلاث مرات وبشكل التالي(0,2,4) لذلك يجب التركيز في هذه المواضيع جيدا



لو تلاحظ المخطط الخاص بالبرنامج عندما يحقق الشرط يتجه لخطوة الطباعة وإذا لم يحقق يخرج من جملة تكرار

- خطوة رقم (٣) هي طباعة قيمة المتغير عند كل (loop) وبما أن الذي يتبع العبارة التكرارية سطر واحد فليس بحاجة لوضعه بين قوسين تضمين ({}) .

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
0      1      2      3      4
```

سيطبع في شاشة التنفيذ

عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١) مرة واحدة فقط. وتنفذ خطوة رقم (٢) وخطوة رقم (٣) خمس مرات بقدر عدات العداد

مثال: لو أردنا عداد يعد من (4) إلى (0) إي يطبع على شاشة التنفيذ (0 1 2 3 4) .؟

تحليل: حلها نفس حل السؤال السابق فقط نقلب العداد إي نجعل قيمته البدائية هي ٤ ؛ وشرط التوقف اكبر أو يساوي صفر ويتناقص بمقدار واحد كل عدة

C++

البرمجة بلغة

```
#include<iostream.h>
main()
{1.int i;
2.for (i=4; i>=0; i--)
3.cout<<i<<"\t";
}
```

C

البرمجة بلغة

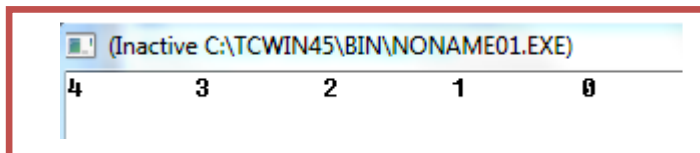
```
#Include<stdio.h>
main()
{1.int i;
2. for (i=4; i>=0; i--)
3.printf("%d\t",i);
}
```

توضيح الخطوات :

١.خطوة رقم (١) عرفنا متغير (i) من نوع integer

٢.خطوة رقم (٢) هي عبارة تكرارية بما إننا نريد العد من الأربعة فأعطينا القيم البدائية أربعة .والعد يصل إلى الصفر معناه شرط التوقف أن يكون اكبر أو يساوي صفر بما إننا نريد أن يعد خمس عدات بين صفر والأربعة لذلك يجب أن يكون مقدار نقصان واحد ليعد (4,3,2,1,0) .لان لو جعلنا مقدار النقصان ٢ مثلا فسيعد العداد ثلاث مرات مرة(4,2,0) لذلك يجب التركيز في هذه المواضيع جيدا

سيطبع في شاشة التنفيذ



عدد مرات تنفيذ الخطوات:

تنفذ خطوة رقم (١) مرة واحدة فقط. وتنفذ خطوة رقم (٢) وخطوة رقم (٣) خمس مرات بقدر عدات العداد

مثال: برنامج لجمع الأعداد الفردية بين (0-100)

تحليل: نرى انه يريد الإعداد الفردية فقط لذلك يجب أن نتجاوز الإعداد الزوجية. نكون عداد يعد من الواحد و شرط التوقف عند المائة ومقدار الزيادة في (٢) حتى نجمع فقط الإعداد الفردية (العداد أولاً يعد الرقم واحد وإذا أضفنا إليه (٢) يعد الرقم ثلاثة وإذا أضفنا (٢) يعد خمسة وإذا أضفنا (٢) يعد الرقم سبعة ويستمر إلى (٩٩))

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int n;
2.int sum=0;
3.for (n=1; n<100; n=n+2)
4. sum += n ;
5.Cout<<"sum="<<sum;
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int n;
2.int sum=0;
3.for (n=1; n<100; n=n+2)
4. sum += n ;
5.printf("sum=%d",sum);
}
```



توضيح الخطوات:

1. خطوة رقم (١) عرفنا متغير (n) من نوع (integer) ليكون عداد للعبارة التكرارية
2. خطوة رقم (٢) عرفنا متغير (sum) من نوع (integer) وأعطيناه قيمة بدائية وهي صفر لأننا سنجمع والنضير الجمعي هو صفر حتى عندما نجمعه مع أول قيمة وهي واحد سيجمع واحد مع الصفر التباس: قد يسأل سائل لما لا نجمعه مباشر إي لا نصفر قيمة (sum) هذا شيء خاطئ لأنه كما نعلم أن المتغيرات عند تعريفها تحجز مكان في الذاكرة لكن لا يخزن في ذلك المكان قيمة يبقى فارغ لذلك لو جمعناه بدون تصفير لجمع قيمة أول عدد فردي مع عنوان الموقع وليس قيمة المخزنة في الموقع لان الموقع ليس فيه إي قيمة لذلك يجب وضع قيمة في الموقع وهي صفر
3. خطوة رقم (٣) كونا عداد يعد من الواحد إلى ٩٩ (لان عندما يصبح ١٠١ يكون أعلى من ١٠٠ فلا ينفذه)
4. خطوة رقم (٤) جمعنا قيمة المتغير (sum) مع قيمة العداد عند كل عدة. إي عندما يكون (n=1) سيجمع (sum=0+1) وعندما يصبح (n=3) سيجمع قيمة (sum) السابقة وهي أصبحت واحد مع ثلاثة (sum=1+3) وعندما يصبح (n=5) سيجمع قيمة (sum) السابقة وهي أصبحت أربعة مع ثلاثة (sum=4+5) ويستمر. ونلاحظ إننا لم نحيط خطوة رابعة بين قوسي لان عبارة التكرار تتبعها خطوة واحد فليس بالحاجة لوضع أقواس
5. خطوة رقم (٥) طباعة الناتج الجمع في شاشة التنفيذ

Sum=2500

عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١ و٢) مرة واحدة فقط. وتنفذ خطوة رقم (٣) وخطوة رقم (٤) خمسين مرة لان الإعداد الفردية بين (0-100) هي خمسين رقم وتنفذ خطوة رقم (٥) مرة واحدة

مثال: برنامج يدخل المستخدم درجات ١٠ مواد وتقوم بحساب المعدل له

تحليل: بما انه يريد حساب المعدل من عشر درجات فيجب جمع هذه الدرجات وقسمتها على عشرة للحصول على المعدل وبما أن يدخلها المستخدم وبما إنهن عشر درجات فيصعب إدخالها كل واحد على حدة لذلك نستخدم عبارة (For) التكرارية تتكرر عشر مرات ونضع تحتها جملة القراءة (حيث عند كل عدة للعبارة التكرارية يطلب منك إدخال درجة ويجمع الدرجات المدخلة عند كل إدخال وناتج الجمع يقسم بعد الإدخال على عشرة

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int n,aveg,input_number;
2.int sum=0;
3.for (n=0; n<10; n++) {
4. cin>> input_number;
5. sum += input_number ;}
6. aveg=sum/10;
7.Cout<<" aveg ="<< aveg ;
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int n,aveg,input_number;
2.int sum=0;
3.for (n=0; n<10; n++) {
4. scanf("%d", &input_number);
5. sum += input_number ;}
6. aveg=sum/10;
7. printf(" aveg =%d", aveg );
}
```

تحليل:

- خطوة رقم (١) عرفنا متغير (n) ليكون عداد للعبارة التكرارية ومتغير حساب المعدل ومتغير إدخال الدرجات
- خطوة رقم (٢) عرفنا متغير (sum) من نوع (integer) وأعطيناه قيمة بدائية وهي صفر لأننا سنجمع والنصير الجمعي هو صفر
- خطوة رقم (٣) هي جملة تكرارية مطلوب منها تكرر الخطوة رقم (٤) وخطوة رقم (٥) عشر مرات وبما انه مطلوب أن يكرر أكثر من سطر برمجي يجب وضعهما بين قوسي العبارة التكرارية
- خطوة رقم (٤) هي دالة إدخال عند كل عدة للجملة التكرارية يطلب من المستخدم إدخال درجة جديدة
- خطوة رقم (٥) يقوم بجمع الدرجات المدخلة عند كل إدخال يجمع الدرجة الجديدة مع ناتج جمع الدرجات السابقة
- خطوة رقم (٦) يقوم بحساب المعدل حيث يقسم ناتج جمع الدرجات المدخلة على عشرة
- خطوة رقم (٧) يقوم بطباعة المعدل

عدد مرات تنفيذ الخطوات:تنفذ خطوة رقم (١و٢) مرة واحدة فقط.وتنفذ خطوة رقم(٣و٤و٥) عشر مرات وتنفذ خطوة رقم(٦ و٧) مرة واحدة

مثال: لمعرفة هل العدد الذي أدخلته عدد أولي أم لا

تحليل: العدد الأولي هو العدد الذي يقبل القسمة على نفسه وعلى واحد فقط (إذا قبل الرقم القسمة على غير هذان الرقمان فهو عدد غير أولي). ولحل هذا السؤال نكون عداد يعد من الاثنان إلى اقل من الرقم المدخل بواحد (مثلاً إذا كان الرقم المدخل ٢١ نكون عدد يعد من ٢ إلى ٢٠) وإذا قبل الرقم المدخل القسمة على أي من أرقام العداد التي سيعدها وهي المحصورة بين اثنان واقل من الرقم بواحد فيكون عدد غير أولي وإذا لم يقبل القسمة على أي من هذه الإعدادات فهو عدد أولي .

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i,input_number;
2.int booleanx=0;
3.cin>> input_number;
4.for(i=2;i< input_number ;i++)
5.if( input_number%i==0)
6.booleanx=1;
7.if (booleanx==1)
8.cout<<"The Number is no prime" ;
9.else
10.cout<<" The Number is prime" ; }
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i,input_number;
2.int booleanx=0;
3 scanf("%d",& input_number);
4.for(i=2;i< input_number ;i++)
5.if( input_number%i==0)
6.booleanx=1;
7.if (booleanx==1)
8.printf("The Number is no prime" ) ;
9.else
10.printf(" The Number is prime" ); }
```

توضيح الخطوات:

- خطوة رقم (١) تم تعريف متغير للعداد وآخر للرقم المدخل
- خطوة رقم (٢) تم تعريف متغير واعتبر كمتغير منطقي وأعطية قيمة صفر إذا كان الرقم المدخل أولي يبقى صفر حتى نهاية البرنامج وإذا كان غير أولي يتغير في خطوة رقم (٦) إلى واحد للدلالة على أن الرقم الذي تم إدخاله رقم غير أولي حتى في نهاية البرنامج وبالتحديد في خطوة رقم (٧) نتأكد من قيمة هذا المتغير إذا بقي صفر فكان الرقم المدخل أولي وإذا تغير إلى واحد كان الرقم المدخل غير أولي
- خطوة رقم (٣) يطلب من المستخدم إدخال الرقم من شاشة التنفيذ للتحقق منه
- خطوة رقم (٤) عداد يعد من الاثنان إلى اقل من الرقم المدخل بواحد ويزداد بمقدار واحد

٥. خطوة رقم (٥) نتحقق هل يقبل الرقم المدخل القسمة على أي رقم من الأرقام التي سيعدها العداد إذا قبل القسمة نغير قيمة للمتغير المنطقي إلى واحد في خطوة رقم (٦)
 ٦. خطوة رقم (٧) بعد أن ينتهي العداد من العد سيتم البرنامج إلى هذه الخطوة ونتأكد من قيمة المتغير المنطقي إذا تغيرت إلى واحد نطبع رسالة أن عدد غير أولي وإذا بقيت صفر نطبع رسالة أن الرقم الذي تم إدخاله هو عدد أولي
 بعد أن نفذنا البرنامج وأدخلنا الرقم ١٣ انظر ماذا ظهر في شاشة التنفيذ

```
(Inactive C:\TCWIN45\B' Hussien
13
is prime
```

عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١ و ٢ و ٣) مرة واحدة فقط.

فإذا كان الرقم المدخل مثلا ٧ ستنفذ خطوة رقم (٤) ثلاث مرات لأنها ستعد (2, 3, 4) وسوف لا تنفذ خطوة رقم (٥ و ٦) لأن إذا كان الرقم المدخل ٧ فهو لا يقبل القسمة على (٢ أو ٣ أو ٤) وبما انه لم يقبل القسمة على أي من هذه الأرقام ستبقى قيمة (booleanx=0) ثابتة على قيمتها الأولية لذلك خطوة رقم (٧ و ٨) لا تنفذ لأن شرطها لم يتحقق لذلك ستنفذ خطوة رقم (٩ و ١٠)

مثال: برنامج لإيجاد مفكوك الإعداد.؟

تحليل: المفكوك هو عملية ضرب العدد بالإعداد التي هي أقل منه وصولا إلى الواحد (مثلا مفكوك ٦ هو $6! = 6 * 5 * 4 * 3 * 2 * 1$). أي أن $(n! = n * (n-1))$ وبرمجيا لحل هذا السؤال نكون عدد يبدأ من الواحد وينتهي بالرقم المدخل ونضرب قيم العداد واحدة بالأخرى إلى النهاية نحصل على المفكوك

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i,number;
2.int fact=1;
3.cin>> number ;
4.for(i=1;i<= number ;i++)
5.fact = fact *i;
6.cout<<"factorial= "<< fact ;}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i,number;
2.int fact=1;
3 scanf("%d",& number );
4.for(i=1;i<= number ;i++)
5.fact = fact *i;
6.printf("factorial=%d", fact );}
```

توضيح خطوات :

١.خطوة رقم (١) تم تعريف متغير (i) كعداد لعبارة التكرار ومتغير آخر يحمل قيمة الرقم المراد إيجاد مفكوكه

٢.خطوة رقم (٢) تم تعريف متغير (fact) كنضير ضرب في أجزاء الرقم المراد إيجاد مفكوكه

٣.خطوة رقم (٣) يطلب من المستخدم إدخال الرقم المراد إيجاد مفكوكه

٤.خطوة رقم (٤) عداد يعد من الواحد وحتى الرقم الذي تم إدخاله كان يكون إننا أدخلنا رقم ٥ سيعد (1,2,3,4,5)

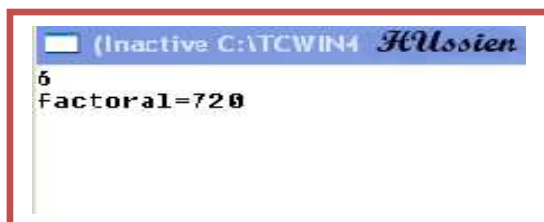
٥.خطوة رقم (٥) هنا نضرب كل قيمة جديدة يعدها العداد بالقيم السابقة كان مثلا إننا أدخلنا رقم ٣ نريد إيجاد مفكوكه سيضرب أولا واحد في قيمة (fact) لتصبح قيمته (fact=1*1=1) ثم سيضرب ٢ في قيمة (fact) السابقة لتصبح قيمته (fact=1*2=2) ثم سيضرب ٣ في قيمة (fact) السابقة لتصبح قيمته (fact=2*3=6)

توضيح خطوات المفكوك الخطوة (٤ و ٥) إذا كان الرقم المدخل ٣

```
i=1
Fact=1*i
Fact=1*1=1
i=2
Fact=1*i
Fact=1*2=2
i=3
Fact=2*i
Fact=2*3=6
```

٥.خطوة رقم (٦) سيطبع المفكوك في شاشة التنفيذ

شاشة التنفيذ إذا أدخلنا الرقم ٦



عدد مرات تنفيذ الخطوات:تنفذ خطوة رقم (١ و ٢ و ٣) مرة واحدة فقط.

إذا كان الرقم المراد إيجاد مفكوكه على سبيل المثال هو ٣ ستنفذ خطوة رقم (٤ و ٥) ثلاث مرات وتنفذ خطوة رقم (٦) مرة واحدة

مثال: برنامج لطباعة المتسلسلة التالية (1234567654321)

تحليل: نرى من السلسلة أنها تصل إلى (٧) وتعود بالتناقص فلحل هذا المثال نكون عداد يعد من الواحد إلى الستة وإذا تجاوز الستة يخرج من الجملة التكرارية ونأخذ قيمته الجديدة ونسدها لعداد آخر يبدأ منها وينتهي بالواحد

```
C++ البرمجة بلغة
#include <iostream.h>
main()
{1.int i,j;
2.for(i=1;i<7;i++)
3.cout<<i;
4.for(j=i;j>0;j--)
5.cout<<j; }
```

```
C البرمجة بلغة
#include <stdio.h>
main()
{1.int i,j;
2.for(i=1;i<7;i++)
3.printf("%d",i);
4.for(j=i;j>0;j--)
5.printf("%d",j);}
```

تحليل الخطوات:

١. خطوة رقم (١) عرفنا متغيرين لنستخدمها كعدادات

٢. خطوة رقم (٢) عبارة تكرارية تعد من (١) إلى (٦) وتطبع قيم العداد في خطوة رقم (٣) التي تكون تابعة للعبارة التكرارية

٣. خطوة رقم (٤) عداد جديد يعد من آخر قيمة وصل إليها العداد الأول ونحن نعلم أن شرط العداد الأول بالاستمرار أن يكون قيمة (i) اقل من سبعة وعندما أصبح قيمته سبعة خرج من الجملة التكرارية إي أن قيمة (i=7) في الخطوة رقم (٤) لذلك سيعد هذا العداد من السبعة ويتناقص إلى الواحد لان شرط توقف أن يكون اكبر من صفر (أي عندما تصبح قيمته صفر يخرج من الجملة التكرارية). ويطبع قيم العداد في خطوة رقم (٥) لأنها تابعة للخطوة رقم (٤)

عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١) مرة واحدة فقط. ستنفذ خطوة رقم (٢ و٣) ست مرات وتنفذ خطوة رقم (٤ و٥) سبع مرات.

٢. عبارة (For--Loop) التكرارية المتداخلة:

شرحنا سابقا على العبارة التكرارية الاعتيادية التي نحتاجها لتكرار سطر برمجي واحد أو عدة اسطر لغرض ما إما هذه العبارة التكرار سوف لا تكرر فقط اسطر برمجية إنما تكرر عبارات (for) تكرارية أخرى (أو عبارات تكرارية أخرى ك (while ,do—while) توجد في داخلها إي في كل عدة لل (for) إلام ستعد (for) الداخلية جميع عداتها. حالها كحال إي خطوة برمجية داخل عبارة تكرارية وبما أن العبارة التكرارية الداخلية يمر عليها عند كل عدة للعبارة التكرارية إلام لذلك في كل عدة للام تعد العبارة التكرارية الداخلية جميع عداتها الممكنة حسب شرطها .

هيكلية For--Loop

```
for (Initializing1; Boolean_Expression1; Update1)
{
for (Initializing2; Boolean_Expression2; Update2)
{
statement
}
}
```

على سبيل المثال لو كان لدينا هذا التداخل

مثال

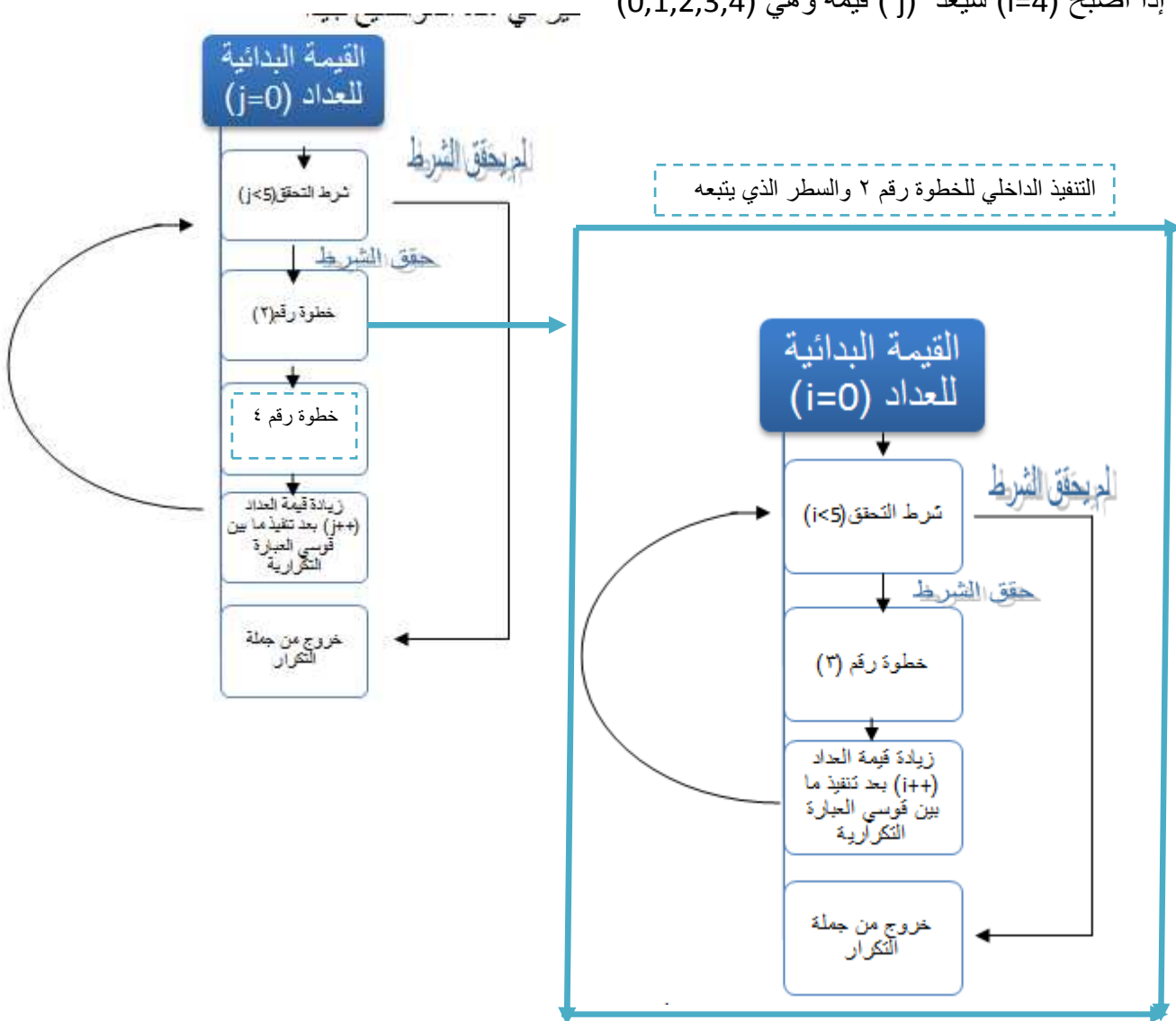
```
1.for (i=0; i<5; i++)
{
2.for (j=0; j<5; j++)
{
3.Statement2
}
4.Statement1
5.}
```



ففي كل عدة بالنسبة (for) في خطوة رقم (١) تعد (for) في خطوة رقم (٢) من الصفر إلى الأربعة وتنفذ (Statement2) خمس مرات وبعدها ينفذ (Statement1) مرة واحدة لكل عدة .
لأن عندما يبدأ العداد الخارجي بالعد (i=0) سينفذ ما موجود داخل قوسيه بالتسلسل خطوة خطوة وهي الخطوات رقم (٢ و٣ و٤). أولا سينفذ خطوة رقم (٢) وبما أن خطوة رقم (٢) أيضا عداد سيبدأ هذا العداد الداخلي بالعد من الصفر إلى خمسة وكل عدة ينفذ خطوة رقم (٣) وهي (Statement2) لأنها واقعة ضمن قوسين خطوة رقم (٢) وعندما ينتهي العداد الداخلي من عداته سينتقل لينفذ ما بعد قوسي العداد في خطوة (٢) وهي الخطوة رقم (٤) سينفذ (Statement1) مرة واحدة تم يصل في خطوة رقم (٥) إلى نهاية قوس (قوس الإغلاق) العبارة التكرارية الخارجية في خطوة رقم (١) وبما انه وصل إلى نهاية قوس العبارة التكرارية الخارجية سيعود الى الخطوة

رقم (١) ويزيد قيمة العداد (i) بمقدار واحد لينفذ ما هو موجود بين قوسي هذه العبارة التكرارية من جديد ما دام شرط توقفها لم يتحقق بعد

- ✓ إي أن إذا كان (i=0) سيعد (z) قيمه وهي (0,1,2,3,4)
- ✓ إذا أصبح (i=1) سيعد (z) قيمه وهي (0,1,2,3,4)
- ✓ إذا أصبح (i=2) سيعد (z) قيمه وهي (0,1,2,3,4)
- ✓ إذا أصبح (i=3) سيعد (z) قيمه وهي (0,1,2,3,4)
- ✓ إذا أصبح (i=4) سيعد (z) قيمه وهي (0,1,2,3,4)



مخطط سير العمليات المثال

**تبقى المبادئ ثابتة كل عبارة تكرارية يتبعها سطر برمجي واحد إذا لم نستخدم أقواس وإذا وضعنا أقواس كل الذي داخل الأقواس هو تابع للعبارة التكرارية.

٤. خطوة رقم (٤) هي عبارة تكرارية داخلية وفي كل عدة للعبارة التكرارية في الخطوة رقم (٢) تعمل هذه العبارة بالعد من (١) إلى (١٠) لكي نضرب كل رقم من الأرقام التي تعدها الخطوة رقم (٢) بالأرقام من (١) إلى (١٠) وهذه العبارة التكرارية يتبعها سطر واحد فقط لعدم وجود أقواس

٥. خطوة رقم (٥) هي عملية طباعة كل قيمة من التي تعدها العبارة التكرارية في الخطوة رقم (٢) بالأرقام من (١) إلى (١٠) التي تعدها العبارة التكرارية في خطوة رقم (٤) وهذه جملة الطباعة هي تابعة لهذه العبارة التكرارية وما تنتجه هذه جملة الطباعة شاهده بالرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب " جملة الطباعة الخطوة (٥)"

٦. خطوة رقم (٦) هي عملية طباعة سطر جديد بعد كل جدول ضرب ولاحظها هي تابعة للعبارة التكرارية في خطوة رقم (٢) وما تنتجه هذه جملة الطباعة شاهده بالرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب " جملة الطباعة الخطوة (٦)"

```

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
Multiply tabel for(1).
-----
1 * 1=1
2 * 1=2
3 * 1=3
4 * 1=4
5 * 1=5
6 * 1=6
7 * 1=7
8 * 1=8
9 * 1=9
10 * 1=10
Multiply tabel for(2).
-----
1 * 2=2
2 * 2=4
3 * 2=6
4 * 2=8
5 * 2=10
6 * 2=12
7 * 2=14
8 * 2=16
9 * 2=18
10 * 2=20

```

Annotations in the image:

- Green box: جملة الطباعة الخطوة (٣) - points to the first line of the first loop.
- Green box: جملة الطباعة الخطوة (٥) - points to the 10th line of the first loop.
- Green box: جملة الطباعة الخطوة (٦) - points to the first line of the second loop.



كيف نعرف أن الخطوة رقم (٦) هي تابعة للخطوة رقم (٢) وليس للخطوة رقم (٤)؟..

كما تلاحظ أن خطوة رقم (٢) تحصر بين قوسيه خطوات رقم (٣ و٤ و٥ و٦) ونلاحظ أن العبارة التكرارية في الخطوة رقم (٤) تتبعها فقط سطر واحد لأننا لم نضع أقواس خلفها أي إننا نقصد فقط السطر الذي يليها هو تابع لها فتبقى الخطوة رقم (٦) وبما أنها داخل قوسين الخطوة رقم (٢) فهي تابعة لها.

مثال: برنامج يطبع الشكل التالي في شاشة التنفيذ .؟

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
1
12
123
1234
12345
```

تحليل: من شاشة التنفيذ نرى انه يعد (1) ثم (12) ثم (123) ويستمر نرى أن تكوين رسم بهذا الشكل يستحيل دون استخدام عبارات التكرار المتداخلة. فما نحتاجه هنا هو عبارة تكرارية خارجية تعد من 2 إلى 6 (أي تعد خمس عدات بقدر عدد الأسطر تعد (2,3,4,5,6)) وعبارة تكرارية داخلية تبدأ بالعد من الواحد وشرط توقفها اقل من قيمة العداد الخارجي لكل عدة مثلاً في العدة الأولى تكون قيمة المتغير (i=2) في العداد الخارجي سيعد العداد الداخلي من واحد إلى اقل من (2) أي سيعد (1) وفي العدة الثانية تكون قيمة المتغير (i=3) في العداد الخارجي سيعد العداد الداخلي من واحد إلى اقل من (3) أي سيعد (12) ويستمر إلى نهاية .

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i,j;
2.for (i=2; i<7; i++)
{3.for (j=1; j<i; j++)
4.cout<<j ;
5.cout<<"\n" ;}
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i,j;
2.for (i=2; i<7; i++)
{3.for (j=1; j<i; j++)
4.printf("%d",j);
5. printf("\n");}
}
```

توضيح الخطوات:

1. خطوة رقم (1) تم تعريف متغيرين للعدادات. وفي خطوة رقم (2) عداد خارجي يعد من 2 إلى 6 وتتبعه الخطوات البرمجية رقم (3 و4 و5) لأنها داخل قوسيه ففي كل عدة له تحدث هذه الخطوات

2. خطوة رقم (3) هي عبارة تكرارية داخلية تبدأ بالعد من الواحد إلى اقل من قيمة العداد الخارجي أي إذا عد العداد الخارجي 2 تعد هذه العبارة التكرارية (1) وإذا عد العداد الخارجي 3 تعد هذه العبارة التكرارية (12) وإذا عد العداد الخارجي 4 تعد هذه العبارة التكرارية (123) وإذا عد العداد الخارجي 5 تعد هذه العبارة التكرارية (1234) ويستمر

3. خطوة رقم (4) تابعة للخطوة رقم (3) حتى تطبع قيم العداد وخطوة رقم (5) تابعة للخطوة رقم (2) حتى بعد كل طباعة ينزل المؤشر إلى سطر جديد لكي يطبع العبارة الجديدة على سطر آخر. جرب احذفها وشاهد ماذا سيحدث

```
(Inactive C:\TCWIN45)
*
 *
  *
   *
```

مثال: برنامج يطبع الشكل التالي في شاشة التنفيذ .؟

تحليل: هذا المثال سهل لو ركزت فيه انه يطبع نجمة وينزل سطر ثم يطبع فراغ ونجمة ثم ينزل سطر ثم فراغ ونجمة وينزل سطر ثم ثلاث فراغات ونجمة وينزل سطر ويستمر بزيادة عدد الفراغات. مقدا انه يحتاج إلى عدادان حتى يكونا الشكل المطلوب العداد الأول يعد من الواحد إلى الخمسة (لأنها خمس نجومات) وآخر داخلي يعد من واحد إلى اقل من العداد الخارجي ليوضع فراغات قبل النجمات

```
البرمجة بلغة C ++
#include<iostream.h>
main()
{1.int star , empty ;
2.for(star=1; star <=5; star ++ )
{3.for( empty =1; empty < star ; empty ++ )
4.cout<<" " ;
5.cout<<"*\n" ;
}
```

```
البرمجة بلغة C
#include<stdio.h>
main()
{1.int star ,empty;
2.for(star=1; star <=5; star ++ )
{3.for(empty =1; empty < star ; empty ++ )
4.printf(" " );
5.printf("*\n" );
}
```

توضيح الخطوات:

١. خطوة رقم (١) تم تعريف متغيرين للعدادات الأول اسم (star) للنجمات وآخر اسمه (empty) لطباعة الفراغات وفي خطوة رقم (٢) عداد خارجي يعد من ١ إلى ٥ وتتبعه الخطوات البرمجية رقم (٣ و٤ و٥) لأنها داخل قوسيه ففي كل عدة له تحدث هذه الخطوات

٢. خطوة رقم (٣) هي عبارة تكرارية داخلية تبدأ بالعد من الواحد إلى اقل من قيمة العداد الخارجي أي إذا عد العداد الخارجي (١) لا تعد هذه العبارة التكرارية لأنها لا يتحقق شرطها ولا تنفذ وخطوة رقم (٤) التابعة لها لا تنفذ أيضا تنفذ الخطوة رقم (٥) فقط ليطلع نجمة وينزل سطر. وإذا عد العداد الخارجي ٢ تعد هذه العبارة التكرارية (١) وتنفذ خطوة رقم (٤) مرة واحدة طابعة فراغ واحد ثم تنفذ خطوة رقم (٥) طابعة نجمة وتنزل سطر. وإذا عد العداد الخارجي ٢ تعد هذه العبارة التكرارية (1,2) وتنفذ خطوة رقم (٤) مرتان طابعة فراغان ثم تنفذ خطوة رقم (٥) طابعة نجمة وتنزل سطر. وإذا عد العداد الخارجي ٣ تعد هذه العبارة التكرارية (1,2,3) وتنفذ خطوة رقم (٤) ثلاث مرات طابعة ثلاث فراغات ثم تنفذ خطوة رقم (٥) طابعة نجمة وتنزل سطر ويستمر إلى النهاية.

مثال: برنامج يكون شكل نقاط كما في الرسم (هذه النقاط هي مواقع عناصر المصفوفة ذات بعدين ٥ * ٥)

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
(0 , 0) (0 , 1) (0 , 2) (0 , 3) (0 , 4)
(1 , 0) (1 , 1) (1 , 2) (1 , 3) (1 , 4)
(2 , 0) (2 , 1) (2 , 2) (2 , 3) (2 , 4)
(3 , 0) (3 , 1) (3 , 2) (3 , 3) (3 , 4)
(4 , 0) (4 , 1) (4 , 2) (4 , 3) (4 , 4)
```

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{ int i,j;
for (i=0; i<5; i++)
{
for (j=0; j<5; j++)
cout<<"<<i<<" , " <<j <<" ";
Cout<<"\n";}
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{ int i,j;
for (i=0; i<5; i++)
{
for (j=0; j<5; j++)
printf("( %d , %d ) ",i,j);
printf("\n");}
}
```

المثال واضح فليس بحاجة إلى توضيح ووضعنا هذه الشكل المطلوب في الرسم كمقدمة للمصفوفات وطريقة تمثيلها حيث هذه النقاط مواقع في الذاكرة حيث أول موقع هو (0,0) ويستمر.....

عبارة (Break) :-

تستخدم هذه الدالة للتوقف عن العبارات التكرارية عند شرط معين والانتقال إلى ما بعد العبارة التكرارية (لاحظ شكل السهم إلى أين ينتقل). أي انه فقط يخرج من عبارة التكرار الموجود هو فيها وينتقل لينفذ الخطوات البرمجية التي تليه

موقع عبارة (Break) في البرنامج

```
for (Initializing1; Boolean_Expression1; Update1)
{
If (condition)
Break;
Statement;
}
```

مثال: لو كان لدينا عبارة تكرارية تعد من (1—200) ونريدها عندما يصل العداد إلى (75) يخرج البرنامج من العبارة التكرارية إليك هذا المثال.

البرمجة بلغة

C ++

```
#include<iostream.h>
main()
{1.int i;
2.for (i=1; i<200; i++)
{3.If (i==75)
4.break;
5.cout<<i<<"\t" ;}
6.cout<<"\nis finished print to 75";
}
```

البرمجة بلغة

C

```
#include<stdio.h>
main()
{1.int i;
2.for (i=1; i<200; i++)
{3.If (i==75)
4.break;
5.printf ("%d\t", i);}
6. printf ("\nis finished print to 75");
}
```

توضيح الخطوات:

1. خطوة رقم (1) تم تعريف متغير ليكون عداد

٢. خطوة رقم (٢) عداد يعد من (١) إلى (٢٠٠) ويكرر في داخله الخطوات رقم (٣ و ٤ و ٥)

٣. خطوة رقم (٣) هو شرط للتوقف أي عندما تصبح قيمة العداد هي (٧٥) ينفذ الخطوة رقم (٤) والتي تسبب بالخروج من العبارة التكرارية والانتقال إلى ما بعدها وهي خطوة رقم (٦) كما تلاحظ في الرسم بالأسهم.

مهم

٤. خطوة رقم (٥) تتم طباعة قيمة العداد عند كل دورة وتلاحظ أن العداد سوف لا يصل في عده إلى (٢٠٠) إنما يصل إلى (٧٤) في الطباعة وينقطع لأنه خطوة رقم (٤) تقطع تنفيذ الجمل البرمجية التي تليها داخل العبارة التكرارية عندما يصل العداد إلى (٧٥) وشاهد شاشة التنفيذ.

٥. خطوة رقم (٦) تطبع رسالة بعد الخروج من الجملة التكرارية وشاهدها في شاشة التنفيذ بالأسفل

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
1      2      3      4      5      6      7      8      9      10
11     12     13     14     15     16     17     18     19     20
21     22     23     24     25     26     27     28     29     30
31     32     33     34     35     36     37     38     39     40
41     42     43     44     45     46     47     48     49     50
51     52     53     54     55     56     57     58     59     60
61     62     63     64     65     66     67     68     69     70
71     72     73     74
is finished print to 75
```

تلاحظ قطعت طباعة العداد عند ٧٤ وطبع بعدها رسالة الخطوة البرمجية رقم (٦)

استخدام (Break) داخل العبارات التكرارية المتداخلة..؟

موقع عبارة (Break) في البرنامج

```
for (Initializing1; Boolean_Expression1; Update1)
{
for (Initializing2; Boolean_Expression2; Update2)
{
If (condition)
Break;
Statement2;
}
Statement1;
}
```

لاحظ كلما يتحقق الشرط داخل العبارة التكرارية الداخلية ينتقل لينفذ ال (Statement1) الموجود داخل العبارة التكرارية الخارجية. أي يخرج من العبارة التكرارية الداخلية.

- تطرقنا في ما مضى على ملاحظة تقول (لإدخال (for) في دواره لانهاية نكتب (for(;;) فقط) لكن لم نبين ما هي وكيف تستخدم .العدادات التي استخدمناها حتى الآن هي محدودة تعد إلى رقم معين وتتوقف ماذا لو أردنا أن لا يتوقف العداد أبدا أو أردنا لا يتوقف إلا بشرط معين مثلا برنامج رقم سري سيضل يطلب من المستخدم إدخال الرقم السري إلى أن يدخل الرقم صحيح بعدها يخرج من العبارة التكرارية وهذا ابسط مثال على عبارة تكرارية لانهاية

مثال:برنامج إدخال الرقم السري ويستمر بطلب الرقم من المستخدم إلى أن يدخل الرقم الصحيح

```

C ++ البرمجة بلغة
#include<iostream.h>
main()
{ int password;
  for (;;)
  { cout<<"enter password\n";
    Cin>> password;
    if (password==1234)
    {cout<<"password is correct";
     break;}
    else
    cout<<"password is error try agin\n";}}

```

```

C ++ البرمجة بلغة
#include<iostream.h>
main()
{ int password;
  for (;;)
  { printf("enter password\n");
    Scanf("%d",&password);
    if (password==1234)
    { printf("password is correct");
     break;}
    else
    printf("password is error try agin\n...");}}

```

شاشة التنفيذ

```

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter password
1654
password is error try agin
enter password
456757
password is error try agin
enter password
1234
password is correct

```

عبارة (continue) :-

تستخدم هذه الدالة لإهمال الخطوات البرمجية التي تليها والانتقال إلى عبارة التكرار وعد عدة جديدة حسب شرط عبارة التكرار إذا كانت تقبل عدة جديدة (لاحظ شكل السهم إلى أين ينتقل). أي انه فقط بهمل ال (Statement) وينتقل إلى عبارة التكرار لبدء عدة جديدة اعتيادية

موقع عبارة (continue) في البرنامج

```
for (Initializing1; Boolean_Expression1; Update1)
{
  If (condition)
  continue;
  Statement;
}
```

مثال: لو كان لدينا عبارة تكرارية تعد من (1—200) ونريدها عندما يصل العداد إلى (75) بهمل الخطوات البرمجية التي تليه داخل عبارة التكرار أي لا يطبع رقم (٧٥).

C ++

البرمجة بلغة

```
#include<iostream.h>
main()
{1.int i;
2.for (i=1; i<200; i++)
{3.If (i==75)
4.continue;
5.cout<<i<<"\t" ;}
6.cout<<"\nis finished print to200  exectp 75 is not print";
}
```

C

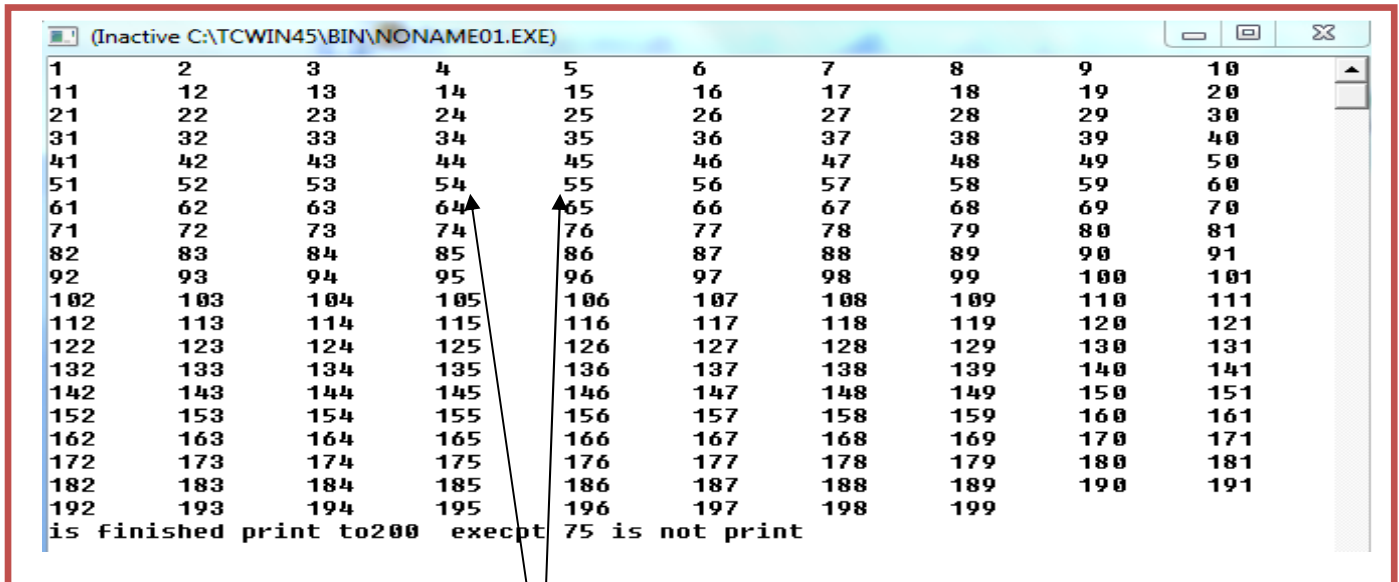
البرمجة بلغة

```
#include<stdio.h>
main()
{1.int i;
2.for (i=1; i<200; i++)
{3.If (i==75)
4.continue;
5.printf ("%d\t", i);}
6. printf ("\nis finished print to200  exectp 75 is not print");
}
```

توضيح الخطوات:

١. خطوة رقم (١) تم تعريف متغير ليكون عداد
٢. خطوة رقم (٢) عداد يعد من (١) إلى (٢٠٠) ويكرر في داخله الخطوات رقم (٣ و ٤ و ٥)
٣. خطوة رقم (٣) هو تحقق من شرط متى أصبح رقم العداد (٧٥) سينفذ خطوة رقم (٤) و سيهمل الخطوات البرمجية التي تليه سيهمل خطوة رقم (٥) أي سوف لا يطبع رقم العداد عند هذه العدة
٤. خطوة رقم (٥) نطبع قيمة العداد عند كل عدة عدا عندما تكون قيمته (٧٥) لأنه سيهمل هذه الخطوة
٥. خطوة رقم (٦) طباعة رسالة بعد الانتهاء من العبارة التكرارية .

شاهد شاشة التنفيذ



```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
1      2      3      4      5      6      7      8      9      10
11     12     13     14     15     16     17     18     19     20
21     22     23     24     25     26     27     28     29     30
31     32     33     34     35     36     37     38     39     40
41     42     43     44     45     46     47     48     49     50
51     52     53     54     55     56     57     58     59     60
61     62     63     64     65     66     67     68     69     70
71     72     73     74     75     76     77     78     79     80
81     82     83     84     85     86     87     88     89     90
91     92     93     94     95     96     97     98     99     100
101    102    103    104    105    106    107    108    109    110
111    112    113    114    115    116    117    118    119    120
121    122    123    124    125    126    127    128    129    130
131    132    133    134    135    136    137    138    139    140
141    142    143    144    145    146    147    148    149    150
151    152    153    154    155    156    157    158    159    160
161    162    163    164    165    166    167    168    169    170
171    172    173    174    175    176    177    178    179    180
181    182    183    184    185    186    187    188    189    190
191    192    193    194    195    196    197    198    199
is finished print to200 except 75 is not print
```

لم يتم طباعة رقم ٧٥ طبع رقم ٧٤ وبعدها طبع رقم ٧٦
لأنه عندما أصبح قيمة العداد ٧٥ تحقق شرط خطوة (٣)
وهمل الخطوات التي تليه بما فيه خطوة الطباعة هملت فلم
يطبع الرقم

عبارة (go to) :-

تؤدي هذه العبارة إلى انتقال تنفيذ البرنامج إلى سطر معين فقط نعنون السطر بأي اسم ونضع بعده (:) وهيكلية هذه الدالة هي.

هيكلية (go to) في البرنامج

```
Main()
{Statement1;
If (condition)
goto npoint;
Statement2;
Statement3;
Statement4;
Npoint:
Statement5;}
```

نلاحظ من الهيكلية انه إذا تحقق شرط معين (وحتى نستطيع أن نضعها بدون شرط لتنفيذ) سوف ينتقل التنفيذ البرنامج لينفذ ال (Statement5) هاملاً أو تاركاً ال (Statement2 Statement3 و Statement4)

مثال: برنامج عداد يعد من (0—14) وباستخدام تكرار الجمل البرمجية أي باستخدام إيعاز (go to)

C ++

البرمجة بلغة

```
#include<iostream.h>
main()
{1. int count1=0;
2.mpointn:
3.cout<< count1<<"\t";
4.count1=count1+1;
5.if (count1!=15)
6.goto mpointn;
}
```

C

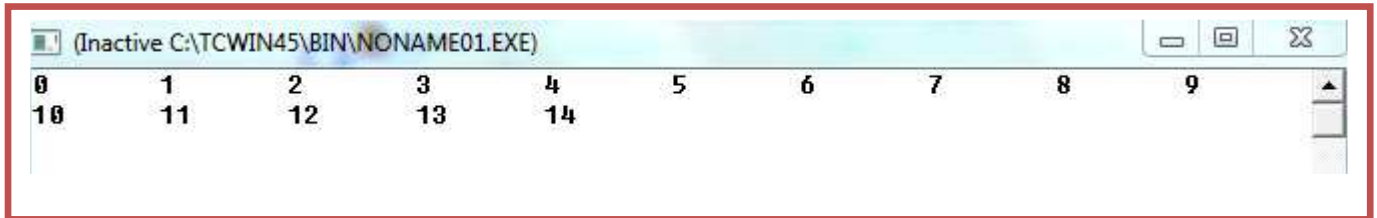
البرمجة بلغة

```
#include<iostream.h>
main()
{1. int count1=0;
2.mpointn:
3.printf("%d\t", count1);
4.count1=count1+1;
5.if (count1!=15)
6.goto mpointn;}
```

توضيح الخطوات:

1. خطوة رقم (١) تم تعريف متغير للعداد التي سيتم طباعة قيمته عند كل عدة
2. خطوة رقم (٢) هي نقطة تعرفها جملة (goto) نكتبها هكذا حتى عندما نريد الرجوع لها فقط نكتب (goto) إلى النقطة التي نود الرجوع إليها. وتسمية النقطة ممكن يكون أي اسم
3. خطوة رقم (٣) تطبع قيمة العداد
4. خطوة رقم (٤) زيادة قيمة العداد بمقدار واحد
5. خطوة رقم (٥) شرط التحقق يتحقق مادام قيمة العداد لم تصل إلى (١٥) يرجع إلى النقطة التي في الخطوة رقم (٢) وينفذ الخطوات بشكل متتالي

شاشة التنفيذ للبرنامج.

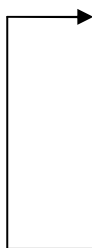


```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
0      1      2      3      4      5      6      7      8      9
10     11     12     13     14
```

ليس فقط نستطيع إهمال الخطوات بل نستطيع تكرار الخطوات أيضا

هيكلية (go to) في البرنامج

```
Main()
{Statement1;
 Npoint:
 Statement2;
 Statement3;
 Statement4;
 If (condition)
 goto npoint;
 Statement5;}
```



سوف يتم تكرار (Statement2 Statement3 و Statement4)

عبارة (While) التكرارية :-

هي عبارة تكرارية تستخدم لتكرار خطوات برمجية المحصورة بين قوسيهما لعدد من المرات تكون مشابه تماما لعبارة التكرار (For). وتستطيع المقارنة بين هيكلية هاتان العبارتان التكراريتان وسوف تجد التشابه بينهما

```
هيكلية while--Loop
Initializing
while ( Boolean_Expression)
{
Statement;
Update
}
```

- **(Initializing):** هي القيمة البداية المعطاة للمتغير التي منها سيبدأ العد. (مثلا $i=0$)
- **(Boolean_Expression):** هو شرط التوقف الذي عندما تصبح قيمة المتغير غير محققة لشرط التوقف سوف يخرج من عبارة (while) بمعنى آخر مادام نتيجة اختبار ال (Boolean_Expression) هي (True) العبارة التكرارية تستمر بتنفيذ ما بين قوسيهما ومتى ما أصبح (False) يخرج من العبارة التكرارية. مثلا ($i < 5$) هو شرط التوقف ويبدأ العد من الواحد
- **(Update):** هي مقدار الزيادة أو النقصان في قيمة المتغير في كل دورة (loop). مثلا ($i++$) أو ($i--$)
- **(statement):** هي الخطوات البرمجية التي ستنفذ عدد من المرات. إذا كانت عبارة عن خطوة برمجية واحدة فليس بحاجة لوضعها داخل أقواس وإذا كانت أكثر من خطوة يجب وضعها داخل أقواس.



كيف تعمل عبارة ال (while) التكرارية.....؟

أن المتغير يبدأ بقيمة بدائية (Initializing) ويستمر بالزيادة أو النقصان حسب (Update) أي قد يزداد أو ينقص بمقدار واحد أو أكثر من واحد ومتى ما أصبح قيمة المتغير غير محققة للشرط (Boolean_Expression) يخرج من العبارة التكرارية إلى الخطوات البرمجية التي تليه وإذا كانت محققة للشرط ينفذ الخطوات البرمجية التي داخل العبارة التكرارية .

- لإدخال (while) في دواره لانهائية نكتب (1) while فقط .
- مثال: لو أردنا عداد يعد من (0) إلى (4) أي يطبع على شاشة التنفيذ من (0 إلى 4)

```
البرمجة بلغة C++
#include<iostream.h>
main()
{1.int i=0;
2.while( i<5)
{3.cout<<i<<"\t";
4.i++;}
}
```



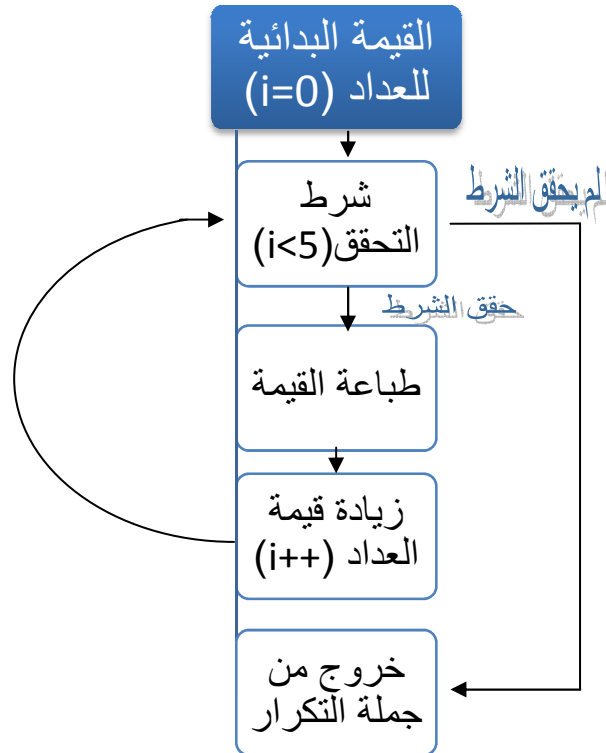
```

C
#include<stdio.h>
main()
{1.int i=0;
2. while( i<5)
{3.printf("%d\t",i);
4.i++;}
}

```

توضيح الخطوات :

١. خطوة رقم (١) عرفنا متغير (i) من نوع integer. بما إننا نريد العد من الصفر فأعطينا القيم البدائية صفر.
٢. خطوة رقم (٢) هي عبارة تكرارية بما أن العد يصل إلى ٤ معناه شرط التوقف أن يكون أقل من خمسة بما إننا نريد أن يعد خمس عدات بين صفر والأربعة لذلك يجب أن يكون مقدار الزيادة واحد ليعد (0,1,2,3,4). لأن لو جعلنا مقدار الزيادة ٢ مثلاً فسيعد العداد ثلاث مرات مرة (0,2,4) لذلك يجب التركيز في هذه المواضيع جيداً.



لو تلاحظ المخطط الخاص بالبرنامج عندما يحقق الشرط يتجه لخطوة الطباعة وإذا لم يحقق يخرج من جملة تكرار

٣. خطوة رقم (٣) هي طباعة قيمة المتغير عند كل (loop) .
٤. خطوة رقم (٤) هو عداد يزداد بمقدار واحد عند كل عدة ولا يفرق إذا كتبنا (++i أو ++i) لأن الرجوع للخطوة (٢) وتنفيذها يعتبر خطوة منفصلة عن خطوة الزيادة لذا على حدة ففي كلا الحالتين يزداد قيمة العداد قبل تنفيذ خطوة تحقق من الشرط (إذا كان أقل من خمسة).

```

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
0      1      2      3      4

```

*قارن هذا المثال بالمثال المشابه له ب(For) وشاهد الفرق

عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١) مرة واحدة فقط. وتنفذ خطوة رقم (٣) وخطوة رقم (٤) خمس مرات

مثال: لو أردنا عداد يعد من (4) إلى (0) أي يطبع على شاشة التنفيذ (0 1 2 3 4)

تحليل: حلها نفس حل السؤال السابق فقط نقلب العداد أي نجعل قيمته البدائية هي 4 و شرط التوقف اكبر أو يساوي صفر ويتناقص بمقدار واحد كل عدة

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i=4;
2.while( i>=0)
{3.cout<<i<<"\t";
4.i--;}
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i=4;
2. while( i>=0)
{3.printf("%d\t",i);
4.i--;}
}
```

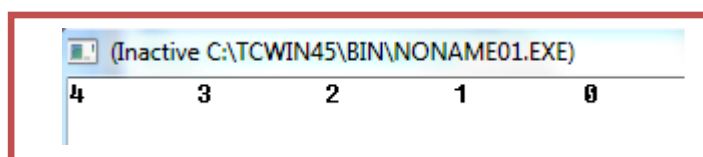
توضيح الخطوات :

١. خطوة رقم (١) عرفنا متغير (i) من نوع integer فأعطيناه القيم البدائية أربعة

٢. خطوة رقم (٢) هي عبارة تكرارية بما إننا نريد العد من الأربعة فأعطينا القيم البدائية أربعة. والعد يصل إلى الصفر معناه شرط التوقف أن يكون اكبر أو يساوي صفر (while(i>=0)) بما إننا نريد أن يعد خمس عدات بين صفر والأربعة لذلك يجب أن يكون مقدار نقصان واحد ليعد (4,3,2,1,0). لأن لو جعلنا مقدار النقصان ٢ مثلا فسيعد العداد ثلاث مرات مرة (4,2,0) لذلك يجب التركيز في هذه المواضيع جيدا

٣. خطوة رقم (٣) طباعة قيمة عداد. وخطوة رقم (٤) عداد النقصان يتناقص بمقدار واحد عند كل عدة

سيطبع في شاشة التنفيذ



```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
4      3      2      1      0
```

عدد مرات تنفيذ الخطوات:

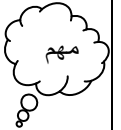
تنفذ خطوة رقم (١) مرة واحدة فقط وتنفذ خطوة رقم (٣) وخطوة رقم (٤) خمس مرات بقدر عدات العداد

مثال: برنامج لجمع الأعداد الفردية بين (0-100)

تحليل: نرى انه يريد الإعداد الفردية فقط لذلك يجب أن نطفر الإعداد الزوجية. نكون عداد يعد من الواحد و شرط التوقف عند المائة ومقدار الزيادة في (٢) حتى نجمع فقط الإعداد الفردية

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int n=1;
2.int sum=0;
3.while( n<100)
{4. sum += n ;
5. n=n+2;}
6.Cout<<"sum="<<sum;}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int n=1;
2.int sum=0;
3.while( n<100)
{4. sum += n ;
5. n=n+2;}
6.printf("sum=%d",sum);}
```



توضيح الخطوات:

١. خطوة رقم (١) عرفنا متغير (n) من نوع (integer) ليكون عداد للعبارة التكرارية وقيمتها الابتدائية هي واحد

٢. خطوة رقم (٢) عرفنا متغير (sum) من نوع (integer) وأعطيناها قيمة بدائية وهي صفر لأننا سنجمع والنضير الجمعي هو صفر حتى عندما نجمعه مع أول قيمة وهي واحد سيجمع واحد مع الصفر

٣. خطوة رقم (٣) كونا عداد يتوقف عند ٩٩ (while(n<100)) ويبدأ العد من الواحد لأنه أول عدد فردي

٤. خطوة رقم (٤) جمعنا قيمة المتغير (sum) مع قيمة العداد عند كل عدة. أي عندما يكون (n=1) سيجمع (sum=0+1) وعندما يصبح (n=3) سيجمع قيمة (sum) السابقة وهي أصبحت واحد مع ثلاثة (sum=1+3) وعندما يصبح (n=5) سيجمع قيمة (sum) السابقة وهي أصبحت أربعة مع ثلاثة (sum=4+5) ويستمر.

٥. خطوة رقم (٥) هو زيادة للعداد بمقدار اثنان حتى فقط نجمع الأعداد الفردية

٦. خطوة رقم (٦) طباعة الناتج الجمع في شاشة التنفيذ

Sum=2500

عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١ و٢) مرة واحدة فقط وتنفذ خطوة رقم (٤) وخطوة رقم (٥) خمسين مرة لان الأعداد الفردية بين (0-100) هي خمسين رقم وتنفذ خطوة رقم (٦) مرة واحدة

مثال: برنامج يدخل المستخدم درجات ١٠ مواد وتقوم بحساب المعدل له

تحليل: بما انه يريد حساب المعدل من عشر درجات فيجب جمع هذه الدرجات وقسمتها على عشرة للحصول على المعدل وبما انه يقول يدخلها المستخدم وبما انهن عشر درجات فيصعب إدخالها كل واحد على حدة لذلك نستخدم عبارة (while) التكرارية تتكرر عشر مرات ونضع تحتها جملة القراءة (حيث عند كل عدة للعبارة التكرارية يطلب منك إدخال درجة ويجمع الدرجات المدخلة عند كل إدخال وناتج الجمع يقسم بعد الإدخال على عشرة

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int aveg,input_number;
2.int sum=0,n=0;
3.while(n<10) {
4. cin>> input_number;
5. sum += input_number ;
6.n++;}
7. aveg=sum/10;
8.Cout<<" aveg ="<< aveg ;
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int aveg,input_number;
2.int sum=0,n=0;
3.while(n<10) {
4. scanf("%d", &input_number);
5. sum += input_number ;
6.n++;}
7. aveg=sum/10;
8. printf(" aveg =%d", aveg );
}
```

تحليل:

- 1.خطوة رقم (١) عرفنا متغير متغير حساب المعدل ومتغير إدخال الدرجات
- 2.خطوة رقم (٢) عرفنا متغير (sum) من نوع (integer) وأعطيناها قيمة بدائية وهي صفر لأننا سنجمع والنصير الجمعي هو صفر وعرفنا المتغير (n) ليكون عداد للعبارة التكرارية وأعطينا قيمته الابتدائية صفر
- 3.خطوة رقم (٣) هي جملة تكرارية مطلوب منها تكرر الخطوة رقم (٤ و ٥ و ٦) عشر مرات
- 4.خطوة رقم (٤) هي دالة إدخال عند كل عدة للجملة التكرارية يطلب من المستخدم إدخال درجة جديدة
- 5.خطوة رقم (٥) يقوم بجمع الدرجات المدخلة عند كل إدخال يجمع الدرجة الجديدة مع ناتج جمع الدرجات السابقة
- 6.خطوة رقم (٦) هو عداد الجملة التكرارية يزداد بمقدار واحد عند كل عدة.
- 7.خطوة رقم (٧) يقوم بحساب المعدل حيث يقسم ناتج جمع الدرجات المدخلة على عشرة
- 8.خطوة رقم (٨) يقوم بطباعة المعدل

٢. عبارة (While) التكرارية المتداخلة:

وتكون مشابهة تماما لعبارات (for) التكرارية. هذه العبارة التكرار سوف لا تكرر فقط اسطر برمجية إنما تكرر عبارات (while) تكرارية أخرى (أو عبارات تكرارية أخرى كـ **for,do—while**) توجد في داخلها أي في كل عدة لل (while) إلام ستعد (while) الداخلية جميع عداتها. حالها كحال أي خطوة برمجية داخل عبارة تكرارية وبما أن العبارة التكرارية الداخلية يمر عليها عند كل عدة للعبارة التكرارية إلام لذلك في كل عدة للام تعد العبارة التكرارية الداخلية جميع عداتها الممكنة حسب شرطها .

هيكلية while--Loop

```
Initializing1;  
While(Boolean_Expression1)  
{ Initializing2;  
While(Boolean_Expression2)  
{  
Statement;  
Update2;  
}  
Update1; }
```

على سبيل المثال لو كان لدينا هذا التداخل

مثال

```
i=0;  
while( i<5)  
{ j=0;  
While(j<5)  
{  
statement  
j++;}  
i++;}
```



ففي كل عدة بالنسبة لل (While) الخارجية تعد ال (While) الداخلية خمس مرات وتنفذ (statement) خمس مرات أي أن إذا كان (i=0) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=1) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=2) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=3) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=4) سيعد ال (j) قيمه وهي (0,1,2,3,4)

**تبقى المبادئ ثابتة كل عبارة تكرارية يتبعها سطر برمجي واحد إذا لم نستخدم أقواس وإذا وضعنا أقواس كل الذي داخل الأقواس هو تابع للعبارة التكرارية.

مثال: برنامج جدول ضرب من (١) إلى (١٠) .؟

تحليل: (المثال نفسه موجود في عبارة (for) التكرارية المتداخلة ستجد تحليل بشكل مفصل).

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i=1,j;
2.while( i<=10) {
3.cout<<"Multiply tabel for("<<i<<"\n-----\n";
4. j=1;
5.while( j<=10) {
6.cout<< j<<" * " <<i<<"="<<i*j<<"\n";
7. j++;}
8.cout<<"\n";
9. i++;}
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i=1,j;
2.while( i<=10) {
3.printf("Multiply tabel for(%d).\n-----\n", i);
4. j=1;
5.while( j<=10) {
6.printf("%d * %d=%d\n",i, j, i*j);
7. j++;}
8.printf("\n");
9. i++;}
}
```

توضيح الخطوات:

١.خطوة رقم (١) تم تعريف متغيرين المتغير (i) هو جدول ضرب الرقم (كان يكون جدول ضرب ثلاثة مثلا) والمتغير (j) هو يضرب بالمتغير الأول الأرقام من واحد إلى العشرة

٢.خطوة رقم (٢) هي عبارة تكرارية تعد من (١) إلى (١٠) وفي كل عدة لها تتكرر الخطوات رقم (٣ إلى ٩)

٣.خطوة رقم (٣) هي رسالة تظهر بداية كل جدول تبين للمستخدم انه هذا الجدول هو للرقم المعين وهي بضبط الرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب (" جملة الطباعة الخطوة (٣) ") وكما تلاحظ أن في كل عدة للعبارة التكرارية الخطوة رقم (٢) تتكرر الخطوة رقم (٣) مرة واحدة لتظهر بداية كل جدول

٤.خطوة رقم (٤) هي عملية وضع قيمة ابتدائية لل (while) الداخلية عند كل عدة لل (while) في خطوة رقم (٢)

٥. خطوة رقم (٥) هي عبارة تكرارية داخلية وفي كل عدة للعبارة التكرارية في الخطوة رقم (٢) تعمل هذه العبارة بالعدد من (١) إلى (١٠) لكي نضرب كل رقم من الأرقام التي تعدها الخطوة رقم (٢) بالأرقام من (١) إلى (١٠) وهذه العبارة التكرارية يتبعها خطوات رقم (٦ و ٧)

٦. خطوة رقم (٦) هي عملية طباعة كل قيمة من التي تعدها العبارة التكرارية في الخطوة رقم (٢) بالأرقام من (١) إلى (١٠) التي تعدها العبارة التكرارية في خطوة رقم (٥) وهذه جملة الطباعة هي تابعة لهذه العبارة التكرارية وما تنتجها هذه جملة الطباعة شاهده بالرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب " جملة الطباعة الخطوة (٥)"

٧. هو عداد خاص بالعبارة التكرارية الداخلية يعد من (١) إلى (١٠)

٨. خطوة رقم (٨) هي عملية طباعة سطر جديد بعد كل جدول ضرب ولاحظها هي تابعة للعبارة التكرارية في خطوة رقم (٢) وما تنتجها هذه جملة الطباعة شاهده بالرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب " جملة الطباعة الخطوة (٦)"

٩. هو عداد خاص بالعبارة التكرارية الخارجية يعد من (١) إلى (١٠)

```

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
Multiply tabel for(1).
-----
1 * 1=1
2 * 1=2
3 * 1=3
4 * 1=4
5 * 1=5
6 * 1=6
7 * 1=7
8 * 1=8
9 * 1=9
10 * 1=10
-----
Multiply tabel for(2).
-----
1 * 2=2
2 * 2=4
3 * 2=6
4 * 2=8
5 * 2=10
6 * 2=12
7 * 2=14
8 * 2=16
9 * 2=18
10 * 2=20
-----

```



كيف نعرف أن الخطوة رقم (٨) هي تابعة للخطوة رقم (٢) وليس للخطوة رقم (٥)؟..

كما تلاحظ أن خطوة رقم (٢) تحصر بين قوسيهما خطوات رقم (٣ إلى ٩) ونلاحظ أن العبارة التكرارية في الخطوة رقم (٥) تتبعها فقط خطوة رقم (٦ و ٧) فتبقى الخطوة رقم (٦) خارج قوس العبارة التكرارية الداخلية وبما أنها داخل قوسين الخطوة رقم (٢) فهي تابعة لها.

عبارة (Do-- While) التكرارية :-

هي عبارة تكرارية تستخدم لتكرار خطوات برمجية المحصورة بين قوسيهما لعدد من المرات تكون مشابه كثيرا لعبارة التكرار (For) و (while) . فقط الاختلاف بينها وبينهم أن عبارة (Do—While) التكرارية تنفذ ما بين قوسيهما ثم يتحقق من الشرط أما (For) و (while) يتحقق من الشرط ثم تنفذ ما بين قوسيهما.

هيكلية Do-- while—Loop

```
Initializing
do {
Statement;
Update
} while ( Boolean_Expression);
```

- (Initializing): هي القيمة البداية المعطاة للمتغير التي منها سيبدأ العد. (مثلا $i=0$)
- (Boolean_Expression): هو شرط التوقف الذي عندما تصبح قيمة المتغير غير محققة لشرط التوقف سوف يخرج من عبارة (while) بمعنى آخر مادام نتيجة اختبار ال (Boolean_Expression) هي (True) العبارة التكرارية تستمر بتنفيذ ما بين قوسيهما ومتى ما أصبح (False) يخرج من العبارة التكرارية. مثلا ($i < 5$) هو شرط التوقف ويبدأ العد من الواحد
- (Update): هي مقدار الزيادة أو النقصان في قيمة المتغير في كل دورة (loop). مثلا ($i++$) أو ($i--$)
- (statement): هي الخطوات البرمجية التي ستنفذ عدد من المرات. إذا كانت عبارة عن خطوة برمجية واحدة فليس بحاجة لوضعها داخل أقواس وإذا كانت أكثر من خطوة يجب وضعها داخل أقواس.

كيف تعمل عبارة ال (do --while) التكرارية.....؟

أن المتغير يبدأ بقيمة بدائية (Initializing) ويستمر بالزيادة أو النقصان حسب (Update) أي قد يزداد أو ينقص بمقدار واحد أو أكثر من واحد ومتى ما أصبح قيمة المتغير غير محققة للشرط (Boolean_Expression) يخرج من العبارة التكرارية إلى الخطوات البرمجية التي تليه وإذا كانت محققة للشرط ينفذ الخطوات البرمجية التي داخل العبارة التكرارية من جديد.

- لإدخال (do--while) في دواراة لانهاية نكتب (1) do -- while فقط .
مثال: لو أردنا عداد يعد من (0) إلى (4) أي يطبع على شاشة التنفيذ من (0 إلى 4)

البرمجة بلغة C++

```
#include<iostream.h>
main()
{1.int i=0;
2.do {
3.cout<<i<<"\t";
4. i++;}
5. while( i<5);
}
```



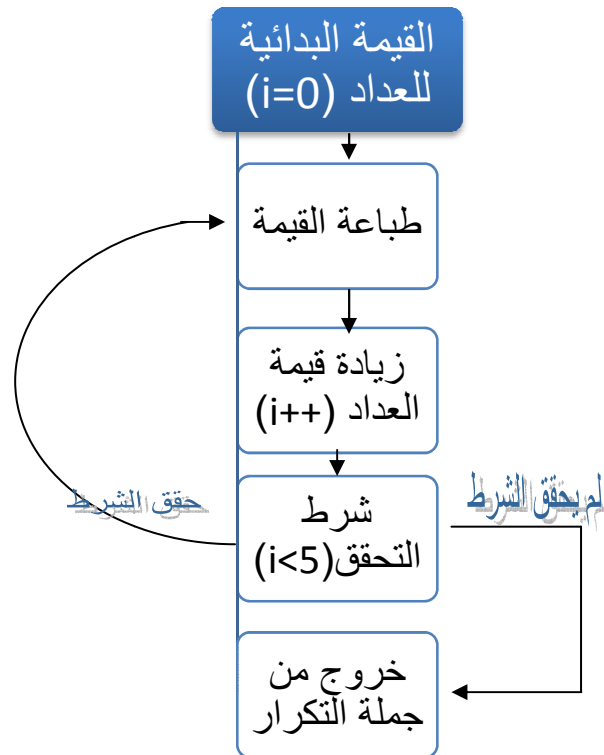
```

C
#include<stdio.h>
main()
{1.int i=0;
2.do {
3.printf("%d\t",i);
4. i++;}
5. while( i<5);
}

```

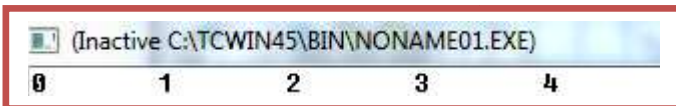
توضيح الخطوات :

١. خطوة رقم (١) عرفنا متغير (i) من نوع integer. بما إننا نريد العد من الصفر فأعطينا القيم البدائية صفر
 ٢. خطوة رقم (٢) تمتد إلى خطوة رقم (٥) هي عبارة تكرارية بما أن العد يصل إلى ٤ معناه شرط التوقف أن يكون اقل من خمسة بما إننا نريد أن يعد خمس عدات بين صفر والأربعة لذلك يجب أن يكون مقدار الزيادة واحد ليعد (0,1,2,3,4). لان لو جعلنا مقدار الزيادة ٢ مثلا فسيعد العداد ثلاث مرات مرة (0,2,4) لذلك يجب التركيز في هذه المواضيع جيدا.



لو تلاحظ المخطط الخاص بالبرنامج عندما يحقق الشرط يتجه لخطوة الطباعة وإذا لم يحقق يخرج من جملة تكرار

٣. خطوة رقم (٣) هي طباعة قيمة المتغير عند كل عدة. خطوة رقم (٤) هو عداد يزداد بمقدار واحد عند كل عدة * قارن هذا المثال بالمثل المشابه له ب(while , For) وشاهد الفرق .



عدد مرات تنفيذ الخطوات: تنفذ خطوة رقم (١) مرة واحدة فقط. وتنفذ خطوة رقم (٣) وخطوة رقم (٤) خمس مرات

مثال: لو أردنا عداد يعد من (4) إلى (0) أي يطبع على شاشة التنفيذ (0 1 2 3 4)

تحليل: حلها نفس حل السؤال السابق فقط نقلب العداد أي نجعل قيمته البدائية هي 4 و شرط التوقف اكبر أو يساوي صفر ويتناقص بمقدار واحد كل عدة

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i=4;
2.do
{3.cout<<i<<"\t";
4.i--;}
5. while( i>=0);
}
```

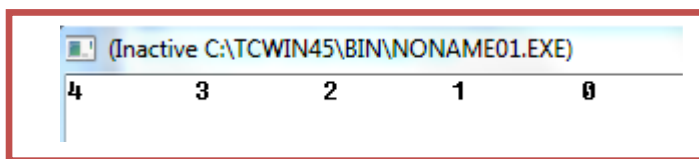
```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i=4;
2.do
3.printf("%d\t",i);
4.i--;}
5.while( i>=0);
}
```

توضيح الخطوات :

١. خطوة رقم (١) عرفنا متغير (i) من نوع integer فأعطيناه القيم البدائية أربعة

٢. خطوة رقم (٢) هي عبارة تكرارية بما إننا نريد العد من الأربعة فأعطينا القيم البدائية أربعة .والعد يصل إلى الصفر معناه شرط التوقف أن يكون اكبر أو يساوي صفر (while(i>=0)) بما إننا نريد أن يعد خمس عدات بين صفر والأربعة لذلك يجب أن يكون مقدار نقصان واحد ليعد (4,3,2,1,0) .لان لو جعلنا مقدار النقصان ٢ مثلا فسيعد العداد ثلاث مرات مرة (4,2,0) لذلك يجب التركيز في هذه المواضيع جيدا
٣. خطوة رقم (٣) هي طباعة قيمة المتغير عند كل عدة .خطوة رقم (٤) هو عداد ينقص بمقدار واحد عند كل عدة

سيطبع في شاشة التنفيذ



```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
4      3      2      1      0
```

عدد مرات تنفيذ الخطوات:

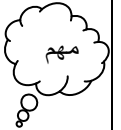
تنفذ خطوة رقم (١) مرة واحدة فقط وتنفذ خطوة رقم (٣) وخطوة رقم (٤) خمس مرات بقدر عدات العداد

مثال: برنامج لجمع الأعداد الفردية بين (0—100)

تحليل: نرى انه يريد الإعداد الفردية فقط لذلك يجب أن نطفر الإعداد الزوجية. نكون عداد يعد من الواحد و شرط التوقف عند المائة ومقدار الزيادة في (٢) حتى نجمع فقط الإعداد الفردية

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int n=1;
2.int sum=0;
3.do
{4. sum += n ;
5. n=n+2;} while( n<100);
6.Cout<<"sum="<<sum;}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int n=1;
2.int sum=0;
3.do
{4. sum += n ;
5. n=n+2;} while( n<100);
6.printf("sum=%d",sum);}
```



توضيح الخطوات:

١. خطوة رقم (١) عرفنا متغير (n) من نوع (integer) ليكون عداد للعبارة التكرارية وقيمه الابتدائية هي واحد

٢. خطوة رقم (٢) عرفنا متغير (sum) من نوع (integer) وأعطيناها قيمة بدائية وهي صفر لأننا سنجمع والنصير الجمعي هو صفر حتى عندما نجمعه مع أول قيمة وهي واحد سيجمع واحد مع الصفر

٣. خطوة رقم (٣) جملة تكرارية (do-- while(n<100)) ويبدأ العد من الواحد لأنه أول عدد فردي إلى ٩٩

٤. خطوة رقم (٤) جمعنا قيمة المتغير (sum) مع قيمة العداد عند كل عدة. أي عندما يكون (n=1) سيجمع (sum=0+1) وعندما يصبح (n=3) سيجمع قيمة (sum) السابقة وهي أصبحت واحد مع ثلاثة (sum=1+3) وعندما يصبح (n=5) سيجمع قيمة (sum) السابقة وهي أصبحت أربعة مع ثلاثة (sum=4+5) ويستمر.

٥. خطوة رقم (٥) هو زيادة للعداد بمقدار اثنان حتى فقط نجمع الأعداد الفردية وإذا تجاوز الشرط يخرج من تكرار

٦. خطوة رقم (٦) طباعة الناتج الجمع في شاشة التنفيذ

Sum=2500

عدد مرات تنفيذ الخطوات:تنفذ خطوة رقم (١ و٢) مرة واحدة فقط. وتنفذ خطوة رقم (٤) وخطوة رقم (٥) خمسين مرة لان الأعداد الفردية بين (0—100) هي خمسين رقم وتنفذ خطوة رقم (٦) مرة واحدة

٢. عبارة (do-- While) التكرارية المتداخلة:

وتكون مشابهة لعبارات (for و while) التكرارية تقريبا. هذه العبارة التكرار سوف لا تكرر فقط اسطر برمجية إنما تكرر عبارات تكرارية أخرى توجد في داخلها (while , for , do-while) . أي في كل عدة لل (do-while) إلام ستعد (do-while) الداخلية جميع عداتها. حالها كحال أي خطوة برمجية داخل عبارة تكرارية وبما أن العبارة التكرارية الداخلية يمر عليها عند كل عدة للعبارة التكرارية إلام لذلك في كل عدة للام تعد العبارة التكرارية الداخلية جميع عداتها الممكنة حسب شرطها .

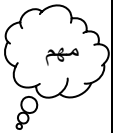
هيكلية do--while

```
Initializing1;  
do  
{ Initializing2;  
do  
{  
Statement;  
Update2;  
} While(Boolean_Expression2) ;  
Update1;  
} While(Boolean_Expression1);
```

على سبيل المثال لو كان لدينا هذا التداخل

مثال

```
i=0;  
do  
{ j=0;  
do  
{  
statement  
j++;} While(j<5)  
i++;} while( i<5);
```



ففي كل عدة بالنسبة لل (do--While) الخارجية تعد ال (do-while-) الداخلية خمس مرات وتنفذ

(statement) خمس مرات أي أن إذا كان (i=0) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=1) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=2) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=3) سيعد ال (j) قيمه وهي (0,1,2,3,4) و إذا أصبح (i=4) سيعد ال (j) قيمه وهي (0,1,2,3,4)

**تبقى المبادئ ثابتة كل عبارة تكرارية يتبعها سطر برمجي واحد إذا لم نستخدم أقواس وإذا وضعنا أقواس كل الذي داخل الأقواس هو تابع للعبارة التكرارية.

مثال: برنامج جدول ضرب من (١) إلى (١٠) .؟

تحليل: (المثال نفسه موجود في عبارة (for) التكرارية المتداخلة ستجد تحليل بشكل مفصل).

```
C++ البرمجة بلغة
#include<iostream.h>
main()
{1.int i=1,j;
2.do {
3.cout<<"Multiply tabel for("<<i<<"\n-----\n";
4. j=1;
5.do {
6.cout<<j<<" * " <<i<<"="<<i*j<<"\n";
7.j++;} while( j<=10);
8.cout<<"\n";
9.i++;} while( i<=10);
}
```

```
C البرمجة بلغة
#include<stdio.h>
main()
{1.int i=1,j;
2.do {
3.printf("Multiply tabel for(%d).\n-----\n", i);
4. j=1;
5.do {
6.printf("%d * %d=%d\n",i, j, i*j);
7.j++;} while( j<=10);
8.printf("\n");
9.i++;} while( i<=10);
}
```

توضيح الخطوات:

١.خطوة رقم (١) تم تعريف متغيرين المتغير (i) هو جدول ضرب الرقم (كان يكون جدول ضرب ثلاثة مثلا) والمتغير (j) هو يضرب بالمتغير الأول الأرقام من واحد إلى العشرة

٢.خطوة رقم (٢) هي عبارة تكرارية تعد من (١) إلى (١٠) وفي كل عدة لها تتكرر الخطوات رقم (٣ إلى ٩)

٣.خطوة رقم (٣) هي رسالة تظهر بداية كل جدول تبين للمستخدم انه هذا الجدول هو للرقم المعين وهي بضبط الرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب (" جملة الطباعة الخطوة (٣)") وكما تلاحظ أن في كل عدة للعبارة التكرارية الخطوة رقم (٢) تتكرر الخطوة رقم (٣) مرة واحدة لتظهر بداية كل جدول

٤.خطوة رقم (٤) هي عملية وضع قيمة ابتدائية لل (do--while) الداخلية عند كل عدة لل (do-- while) في خطوة رقم (٢)

٥. خطوة رقم (٥) هي عبارة تكرارية داخلية وفي كل عدة للعبارة التكرارية في الخطوة رقم (٢) تعمل هذه العبارة بالعدد من (١) إلى (١٠) لكي نضرب كل رقم من الأرقام التي تعدها الخطوة رقم (٢) بالأرقام من (١) إلى (١٠) وهذه العبارة التكرارية يتبعها خطوات رقم (٦ و ٧)

٦. خطوة رقم (٦) هي عملية طباعة كل قيمة من التي تعدها العبارة التكرارية في الخطوة رقم (٢) بالأرقام من (١) إلى (١٠) التي تعدها العبارة التكرارية في خطوة رقم (٥) وهذه جملة الطباعة هي تابعة لهذه العبارة التكرارية وما تنتجها هذه جملة الطباعة شاهده بالرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب " جملة الطباعة الخطوة (٥)"

٧. هو عداد خاص بالعبارة التكرارية الداخلية يعد من (١) إلى (١٠)

٨. خطوة رقم (٨) هي عملية طباعة سطر جديد بعد كل جدول ضرب ولاحظها هي تابعة للعبارة التكرارية في خطوة رقم (٢) وما تنتجها هذه جملة الطباعة شاهده بالرسالة الموجودة في شاشة التنفيذ في الأسفل ومؤشر عليها ب " جملة الطباعة الخطوة (٦)"

٩. هو عداد خاص بالعبارة التكرارية الخارجية يعد من (١) إلى (١٠)

```

(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
Multiply tabel for(1).
-----
1 * 1=1
2 * 1=2
3 * 1=3
4 * 1=4
5 * 1=5
6 * 1=6
7 * 1=7
8 * 1=8
9 * 1=9
10 * 1=10
-----
Multiply tabel for(2).
-----
1 * 2=2
2 * 2=4
3 * 2=6
4 * 2=8
5 * 2=10
6 * 2=12
7 * 2=14
8 * 2=16
9 * 2=18
10 * 2=20
-----

```



كيف نعرف أن الخطوة رقم (٨) هي تابعة للخطوة رقم (٢) وليس للخطوة رقم (٥)؟..

كما تلاحظ أن خطوة رقم (٢) تحصر بين قوسيهما خطوات رقم (٣ إلى ٩) ونلاحظ أن العبارة التكرارية في الخطوة رقم (٥) تتبعها فقط خطوة رقم (٦ و ٧) فتبقى الخطوة رقم (٦) خارج قوس العبارة التكرارية الداخلية وبما أنها داخل قوسين الخطوة رقم (٢) فهي تابعة لها.



ما الفرق إذن بين دوال التكرار (for,while,do—while)؟..

كما قلنا أن الاختلاف بينها وبينهم أن دالة (Do—While) التكرارية تنفذ ما بين قوسيهما ثم يتحقق من الشرط فإذا كان ناتج الشرط (True) تعيد تنفيذ ما بين قوسيهما من جديد وإذا كان (False) فنتنقل للخطوة التي تليه. أما (For) و (while) يتحقق من الشرط ثم تنفذ ما بين قوسيهما كان ناتج الشرط (True) تنفذ ما بين قوسيهما وإذا كان (False) فنتنقل للخطوة التي تليه.

لتوضيح الفكرة بهذا المثال.

مثال: برنامج يطلب منك أن تدخل رقم ويقدر الرقم يطبع عبارة (Hello!) أي لو أدخلت (٥) سيطبع هذه العبارة خمسة مرات وإذا أدخلت واحد يطبعها مرة واحدة.

تحليل: مادام عدد مرات الطباعة مرتبط بإدخال المستخدم فالحل يكون عبارة تكرارية تبدأ من الواحد إلى الرقم الذي أدخله المستخدم وتحتة جملة طباعة (أي إذا أدخل ٢ سيعد العداد (1,2) أي عدتان ويطبع الجملة مرتان).

الحل باستخدام دالة (For) التكرارية.

C++

البرمجة بلغة

```
#include<iostream.h>
main()
{
1.int i,input_numbe;
2.cin>> input_numbe ;
3.for (i=1 ;i<= input_numbe ;i++)
4.cout<<"Hello!\n";
}
```

C

البرمجة بلغة

```
#include<stdio.h>
main()
{
1.int i,input_numbe;
2 scanf("%d",& input_numbe) ;
3.for (i=1 ;i<= input_numbe ;i++)
4.printf("Hello!\n");
}
```

الحل باستخدام دالة (While) التكرارية.

C++

البرمجة بلغة

```
#include<iostream.h>
main()
{1.int i=1,input_numbe;
2.cin>> input_numbe ;
3.while(i<= input_numbe )
{ 4.cout<<"Hello!\n";
5.i++; }
}
```

C

البرمجة بلغة

```
#include<stdio.h>
main()
{1.int i=1,input_numbe;
2 scanf("%d",& input_numbe) ;
3.while(i<= input_numbe )
{4.printf("Hello!\n");
5.i++; }
}
```

الحل باستخدام دالة (Do--While) التكرارية.

C++

البرمجة بلغة

```
#include<iostream.h>
main()
{1.int i=1,input_numbe;
2.cin>> input_numbe ;
3.do
{ 4.cout<<"Hello!\n";
5.i++; } while(i<= input_numbe ) ;
}
```

C

البرمجة بلغة

```
#include<stdio.h>
main()
{1.int i=1,input_numbe;
2 scanf("%d",& input_numbe) ;
3.do
{4.printf("Hello!\n");
5.i++; } while(i<= input_numbe ) ;
}
```


الآن لنقارن نتائج الأمثلة الثلاثة

١. إذا أدخلنا الرقم (٥) من شاشة التنفيذ

سوف يطبع (for,while) الجملة خمس مرات لأنه عداد يعد من (١) إلى (٥) وإذا تجاوز الرقم (٥) لا يسمح له بطباعة الجملة من جديد ويخرج من العبارة التكرارية

سوف يطبع (do--while) الجملة خمس مرات لأنه عداد يعد من (١) إلى (٥) وإذا تجاوز الرقم (٥) لا يسمح له بطباعة الجملة من جديد ويخرج من العبارة التكرارية

```
(Inactive C:\TCWI
5
Hello!
Hello!
Hello!
Hello!
Hello!
```

وتكون شاشة التنفيذ بالنسبة للجميع هكذا

وأي رقم من واحد فما فوق تعطي الجميع نفس النتائج



١. إذا أدخلنا الرقم (0) من شاشة التنفيذ

سوف لا يطبع (for,while) جملة الطباعة ولا مرة لأنه عندما يتحقق من الشرط في خطوة رقم (٣) وهو (i<= input_numbe) وهو هكذا (1<=0=false) سوف لا يحقق الشرط العبارة الشرطية لذلك سوف لا ينفذ الجملة التابعة لعبارة التكرار. ولا يطبع جملة الطباعة التابعة لها

```
(Inactive C:\TCWIN45\BIN\N
0
```

وتكون شاشة التنفيذ هكذا

سوف يطبع (do--while) الجملة مرة واحدة لأنه أولاً ينفذ الخطوات (٣ و٤ و٥) بعد خطوة خامسة يتحقق من الشرط ويجد انه الشرط لا يتحقق ولا يعيد عبارة التكرار. ولأنه طبع الجملة في خطوة (٤) قبل التحقق من الشرط

```
(Inactive C:\TCWIN45\BI
0
Hello!
```

وتكون شاشة التنفيذ هكذا

وهذا معنى قولنا (تنفذ (do-while) ما بين قوسيهما ثم تتحقق من الشرط)

الفصل الرابع

المصفوفات وأنواعها

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصول السابقة وفاهماً كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على المصفوفات وطرق تمثيلها

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب: ١٢ ساعة

المصفوفات الأحادية الأبعاد:

قبل أن نتحدث عن المصفوفات علمنا أن المتغيرات تحجز مكان في الذاكرة تخزن فيه قيمة المتغير .
لو أخذنا هذا الجزء من الذاكرة وافترضنا انه المتغير (x=15) مخزون في موقع (18126).

مواقع خلايا الذاكرة	
الموقع	محتواه
18125	data
18126	15
18127	data

هذا هو موقع المتغير x

مهم

كما تلاحظ من الرسم أن المتغير (x) حاز مكان في الذاكرة وخازن قيمته بداخله.
فما هي المصفوفات. هي مجموعة خلايا متتالية في الذاكرة تحجز لغرض خزن معلومات معينة في داخلها كأن نخزن في داخلها أرقام أو أحرف وتبقى هذه القيم مخزنة داخل المصفوفة حتى نغلق البرنامج إذا لم نغيرها داخل البرنامج المصفوفات يجب الإعلان عن عدد المواقع التي نحتاجها في العمل في بداية البرنامج حتى يحجزها المترجم للمصفوفة ولا يخزن قيم أخرى داخل في داخلها و تبقى محجوزة فقط لعناصر المصفوفة. ويكون الإعلان عليها هكذا

هيكلية

Type arrayname[size of array]

- (**arrayname**) هو اسم المصفوفة الذي سنتعامل معه في البرنامج أي اسم ممكن أن نسمي المصفوفة
- (**size of array**) هو حجم الذي سنتشغله المصفوفة في الذاكرة وقد يكون أي رقم حسب احتياجك
- (**Type**) هو نوع المصفوفة التي سوف نعرفها قد تكون حرفية أو رقمية. لو عرفنا مصفوفة من نوع (**integer**) جميع عناصرها تكون (**integer**) ولا يجوز تخزين أحرف في داخلها

مثال: تعريف مصفوفة من نوع (**integer**) حجمها خمسة واسمها (**first_array**).

هيكلية

int first_array [5];

لنفرض أن تخزين أول عنصر في المصفوفة بدء بموقع (18126) ويكون شكل مواقع الذاكرة هكذا

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0]	18126	
first_array [1]	18127	
first_array [2]	18128	
first_array [3]	18129	
first_array [4]	18130	
	18131	data

هذه المواقع محجوزة للمصفوفة first_array



لو تلاحظ أن أول عنصر للمصفوفة وهو (`first_array [0]`) يحجز الموقع (18126) وتخزن فيه قيمة أول عنصر ويليها بقية العناصر بالتتالي أي (`first_array [1]`) يحجز الموقع (18127) وبالتتالي...؟

** عنونة عناصر المصفوفة تبدأ من الصفر كما تلاحظ أول موقع بالمصفوفة هو (`first_array [0]`)

هيكلية

```
first_array [index]
```

(index) هو عنوان الموقع الذي نريد أن نصل إلى محتوياته في داخل المصفوفة لنعدل عليها أو نطبعها.

- مثلاً أردنا وضع (79) بثالث موقع في المصفوفة نكتب هكذا

كود

```
first_array [2]=79;
```

لو تلاحظ أن (`index=2`) وليس (3) لأنه كما قلنا تسلسل عناصر المصفوفة يبدأ من الصفر وليس من الواحد أي لو كان لدينا مصفوفة حجمها خمسة عناصر فتسلسل (`index`) يكون من الصفر إلى الأربعة. لذلك بعد أن وضعنا رقم (79) بالموقع الثالث يصبح شكل المصفوفة في مواقع الذاكرة هكذا

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
<code>first_array [0]</code>	18126	
<code>first_array [1]</code>	18127	
<code>first_array [2]</code>	18128	79
<code>first_array [3]</code>	18129	
<code>first_array [4]</code>	18130	
	18131	data

محتويات الموقع الثالث

هذه المواقع محجوزة للمصفوفة `first_array`

- لو أردنا إدخال قيمة (90) في الموقع الرابع في المصفوفة فيكون الكود

كود

```
first_array [3]=90;
```

- لو أردنا طباعة محتويات الموقع الثالث

C++	الكود بلغة C	الكود بلغة C++
<code>cout<< first_array [2];</code>	<code>Printf("%d", first_array [2]);</code>	

لو أردنا تعريف مصفوفة من نوع float نفس طريقة مصفوفة السابقة وكذلك بقية الأنواع تعرف بنفس الطريقة

هيكلية

```
float first_array [5];
```

نستطيع إعطاء عناصر المصفوفة قيم معينة كقيم أولية وقت التعريف عن المصفوفة هكذا

كود

```
int first_array [5]={34,26,43,23,54};
```

فيكون شكل المصفوفة في مقطع الذاكرة وهي مخزن فيها العناصر هكذا

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0]	18126	34
first_array [1]	18127	26
first_array [2]	18128	43
first_array [3]	18129	23
first_array [4]	18130	54
	18131	data

هذه المواقع محجوزة
للمصفوفة
first_array
ومحتوياتها

إي أن العنصر الأول في المصفوفة تكون قيمته (first_array [0]=34) وبقية المواقع بالتسلسل كما في الشكل .

مثال: برنامج ندخل مصفوفة أحادية الأبعاد مكونة من ستة عناصر من شاشة التنفيذ ونطبعها!؟

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> main() { 1.int i,first_arr[6]; 2.for (i=0;i<6;i++) 3.cin>> first_array[i] ; 4.cout<<"the Content of array is .\n"; 5.for (i=0;i<6;i++) 6.cout<< first_array[i]<<"\t";}</pre>	<pre>#include<stdio.h> main() {1. int i,first_array[6]; 2.for (i=0;i<6;i++) 3 scanf("%d",& first_array[i]) ; 4.printf("the Content of array is .\n"); 5.for (i=0;i<6;i++) 6.printf("%d\t", first_array[i]);}</pre>

توضيح الخطوات:

١. خطوة رقم (١) تم تعريف متغير للعداد وتم تعريف مصفوفة أرقام حجمها ستة

٢. خطوة رقم (٢) هو اعداد يعد من صفر إلى خمسة وكل عدة يتم طلب في خطوة (٣) إدخال عنصر جديد من المصفوفة كما هو مبين في الشكل ، عندما يكون (i=0) فان (first_array[i]) معناه (first_array[0]) أي يطلب من المستخدم إدخال العنصر الأول ويستمر بطلب الإدخال إلى آخر عنصر وهو (i=5) (كما هو بالشكل بالأسفل)

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
first_array[0] first_array[1] first_array[2] first_array[3] first_array[4]
first_array[5]
```

٣.خطوة رقم (٣) هناك طريقتين للقراءة أم ضغط مفتاح (enter) وإدخال عنصر جديد من عناصر المصفوفة أو ضغط المسطرة وإدخال عنصر جديد . خطوة رقم (٤) هي رسالة تقول انه سوف يتم طباعة محتويات المصفوفة

٥.خطوة رقم (٦) عداد يعد من صفر إلى الخمسة ليطلع جميع عناصر المصفوفة في الخطوة رقم (٦). أي عندما يعد العداد رقم صفر ستنفذ خطوة رقم (٦) طباعة (first_arr[0]) تم عندما يعد العداد رقم واحد ستنفذ خطوة رقم (٦) طباعة (first_arr[1]) تم عندما يعد العداد رقم اثنان ستنفذ خطوة رقم (٦) طباعة (first_arr[2]) ويستمر بطباعة جميع عناصر المصفوفة من محتويات أول موقع إلى محتويات آخر موقع .

وتكون شاشة التنفيذ هكذا

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
8 45 34 2 9 5 3
the Content of array is .
8      45      34      2      9      5
```



لو غيرنا عداد خطوة رقم(٥) وجعلناه يتناقص من خمسة إلى الصفر سوف يطبع أولاً (first_arr[5]) ثم (first_arr[4]) ويستمر إلى الصفر طابعا المصفوفة بالمقلوب (من آخر عنصر إلى أول عنصر)

أي يصبح عداد خطوة رقم (٥) هكذا

كود

```
5.for (i=5;i>=0;i--)
```

ستكون شاشة التنفيذ بعد الإدخال هكذا

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
43 54 23 7 4 34
the Content of array is .
34      4      7      23      54      43
```



أبدال بين قيم موقعين: لإبدال بين قيمة موقعين في المصفوفة نحتاج إلى متغير ثالث من نفس نوع المصفوفة حتى نخزن به نتيجة احد الموقعين ثم نبدل لان في حال عدم وجود متغير ثالث لا نستطيع الإبدال وستضيع احد القيمتين

كود الإبدال بين قيمة المتغير (first_arr[5]) والمتغير (first_arr[2])

```
int item; // هو المتغير المؤقت الذي سنخزن به احد قيم الموقعين
Item= first_arr[5];
first_arr[5]= first_arr[2];
first_arr[2]=item;
```

مثال: برنامج ندخل مصفوفة أحادية الأبعاد مكونة من خمسة عناصر ونجمعها .؟

تحليل: خطوات الإدخال هي اعتيادية وتقريبا ثابتة في المصفوفات فمصفوفة حجمها خمسة نحتاج عداد يعد من الصفر إلى الأربعة ثم بعد الإدخال نجمع جميع عناصر المصفوفة مع بعضها

C++	البرمجة بلغة c
<pre>#include<iostream.h> main() { 1.int i,array1[5],sum=0; 2.for (i=0;i<5;i++) 3.cin>> array1[i] ; 4.for (i=0;i<5;i++) 5.sum=sum+array1[i]; 6.cout<< "sum of array item="<<sum;}</pre>	<pre>#include<stdio.h> main() { 1.int i,array1[5],sum=0; 2.for (i=0;i<5;i++) 3 scanf("%d",& array1[i]); 4.for (i=0;i<5;i++) 5.sum=sum+array1[i]; 6.printf("sum of array item=%d",sum);}</pre>

توضيح الخطوات:

- خطوة رقم (١) تم تعريف متغير للعداد وتم تعريف مصفوفة أرقام حجمها خمسة ومتغير للجمع وأعطينا صفر كقيمة ابتدائية كما قلنا سابقا يجب تصفير المتغير الجمع قبل الجمع
- خطوة رقم (٢ و ٣) إدخال المصفوفة كما تلاحظ العداد يعد من الصفر إلى الأربعة لان حجم المصفوفة خمسة
- خطوة رقم (٤) عداد يمر على عناصر المصفوفة عنصر لكي يجمعها مع بعضها أي عندما يكون قيمة العداد هي (i=0) سيجمع (sum=sum+array1[0]) ويستمر بالجمع مع القيم السابقة كما هي هذا الشكل هذا الشكل هو تتبع للعمليات التي تحدث في خطوة رقم (٤) وخطوة رقم (٥)

توضيح عملية جمع في المصفوفة إذا فرضنا عناصر المدخلة هي (11,43,23,56,45) كما في شاشة التنفيذ

```
When i=0
sum=sum+array1[0]; → sum=0+11 →sum=11
When i=1
sum=sum+array1[1]; → sum=11+43 →sum=54
When i=2
sum=sum+array1[2]; → sum=54+23 →sum=77
When i=3
sum=sum+array1[3]; → sum=77+56 →sum=133
When i=4
sum=sum+array1[4]; → sum=133+45 →sum= 178
```

لو تلاحظ كيف يجمع نتائج الجمع السابقة مع العناصر الجديدة.....!

٤.خطوة رقم (٦) يطبع نتيجة الجمع

```
(Inactive C:\TCWIN45\BIN\NON)
11
43
23
56
45
sum of array item=178
```

مثال: برنامج ندخل مصفوفة أحادية الأبعاد مكونة من سبعة عناصر جد اكبر رقم.

تحليل: خطوات الإدخال هي اعتيادية وتقريبا ثابتة في المصفوفات فمصفوفة حجمها سبعة نحتاج عداد يعد من الصفر إلى ستة. ثم بعد الإدخال نجد اكبر رقم بين عناصر المصفوفة كيف؟ نسند قيمة أول عنصر في المصفوفة لمتغير و ثم نقارن قيمة هذا المتغير مع بقية العناصر أي عنصر في المصفوفة نجد أن قيمته اكبر من قيمة متغيرنا نبدل قيمة متغيرنا بقيمة عنصر المصفوفة ونستمر حتى النهاية وكل ما وجد متغيرنا قيمة أعلى منه أخذها وإذا اقل منه لاتهمه ويستمر إلى النهاية ويكون في التالي حاملا أعلى قيمة.

البرمجة بلغة	c	البرمجة بلغة	C++
#include<stdio.h> main() { 1.int i,array1[7],max; 2.for (i=0;i<7;i++) 3 scanf("%d",& array1[i]); 4.max=array1[0]; 5.for (i=0;i<7;i++) 6.if (array1[i] > max) 7.max=array1[i]; 8.printf("max number in array1 is=%d",max);}	#include<iostream.h> main() { 1.int i,array1[7],max; 2.for (i=0;i<7;i++) 3.cin>> array1[i] ; 4.max=array1[0]; 5.for (i=0;i<7;i++) 6.if (array1[i] > max) 7.max=array1[i]; 8.cout<< "max number in array1 is="<<max;}		

وهذا توضيح لما سيحصل في خطوات رقم (٥ و٦ و٧)

جعلت الخطوات التي تنفذ ذو لون عميق والتي لا تنفذ ذات لون رفيع

توضيح (٥ و٦ و٧) إذا فرضنا العناصر المدخلة هي (11,32,22,32,43,31,23) كما في شاشة التنفيذ

4. max=array1[0]; → max=11

5. When i=0

6. هذه الخطوة لا تحقق الشرط لان قيمة المتغير (max=11) ليست اصغر من (array1[0]=11)

7. لا تنفذ هذه الخطوة إذا لم يتحقق الشرط في خطوة (٦)

5. When i=1

6. هذه الخطوة تحقق الشرط لان قيمة المتغير (max=11) اصغر من (array1[1]=32)

7. max =array1[1] → max=32 // أخذ المتغير قيمة جديدة اكبر منه لتصبح قيمته ٣٢

5. When i=2

6. هذه الخطوة لا تحقق الشرط لان قيمة المتغير (max=32) ليست اصغر من (array1[2]=22)

7. لا تنفذ هذه الخطوة إذا لم يتحقق الشرط في خطوة (٦)

5. When i=3

6. هذه الخطوة لا تحقق الشرط لان قيمة المتغير (max=32) ليست اصغر من (array1[3]=32)

7. لا تنفذ هذه الخطوة إذا لم يتحقق الشرط في خطوة (٦)

5. When i=4

6. هذه الخطوة تحقق الشرط لان قيمة المتغير (max=32) اصغر من (array1[4]=43)

7. max =array1[4] → max=43 // أخذ المتغير قيمة جديدة اكبر منه لتصبح قيمته ٤٣

5. When i=5

6. هذه الخطوة لا تحقق الشرط لان قيمة المتغير (max=43) ليست اصغر من (array1[5]=31)

7. لا تنفذ هذه الخطوة إذا لم يتحقق الشرط في خطوة (٦)

5. When i=6

6. هذه الخطوة لا تحقق الشرط لان قيمة المتغير (max=43) ليست اصغر من (array1[6]=23)

7. لا تنفذ هذه الخطوة إذا لم يتحقق الشرط في خطوة (٦)

توضيح الخطوات:

١. خطوة رقم (١) تم تعريف متغير للعداد وتم تعريف مصفوفة أرقام حجمها سبعة ومتغير لخرن اكبر رقم وخطوة رقم (٢ و ٣) هي إدخال للمصفوفة

٢. خطوة رقم (٤) سندنا قيمة أول موقع إلى متغير (max) حتى نقارنه مع بقية المواقع حتى أي قيمة تكون اكبر منه في المقارنة نبدلها بقيمة (max) حتى نحصل على اكبر قيمة.

٣. خطوة رقم (٥) عداد يمر على جميع عناصر المصفوفة لكي نقارن في خطوة رقم (٦) عناصر المصفوفة بقيمة المتغير ومتى ما كان احد عناصر المصفوفة قيمته اكبر نبدل قيمة (max) في خطوة رقم (٧) بالقيمة الأكبر منه

لو تلاحظ أن قيمة الموقع الخامس هي اكبر قيمة من بين القيم (شاهد التوضيح وكيفية تغير قيمة (max))

Array1[0]	Array1[1]	Array1[2]	Array1[3]	Array1[4]	Array1[5]	Array1[6]	تغير قيمة المتغير عند (max) مقارنته
11	32	22	32	43	31	23	
Max=11	Max=32	Max=32	Max=32	Max=43	Max=43	Max=43	

*

```
(Inactive C:\TCWIN45\BIN\NONAME0
11
32
22
32
43
31
23
max number in array1 is=43
```

٤. خطوة رقم (٨) يطبع اكبر رقم ←



لو أردنا إيجاد اصغر رقم في المصفوفة فقط نبدل الشرط في الخطوة رقم (٦) إلى اصغر هكذا

```
كود
6.if (array1[i] < max )
```

وطريقة التتبع هنا كلما يجد رقم في المصفوفة اصغر من قيمة المتغير يسند المتغير للقيمة الأصغر



لو أردنا ضرب كل عنصر من عناصر المصفوفة في (٢)

```
كود
array1[i]=2* array1[i];
```

كأنما نقول قيمة العنصر الجديد تساوي قيمته القديمة مضروبة في (٢)

وكذلك بقية العمليات الرياضية تعامل نفس الطريقة كالقسمة. مثلا نقسم كل عنصر على ٤

كود

```
array1[i]= array1[i] /4;
```

ولا توضع فقط هذه الجملة وحدها يجب وضع قبلها عداد يمر على جميع عناصر المصفوفة لكي يقسم جميع العناصر عنصر عنصر لو تلاحظ خطوة رقم (٢) وخطوة رقم (٣) في المثال التالي.

مثال: مصفوفة من خمسة عناصر إضافة قيمة خمسة لكل عنصر في المصفوفة

البرمجة بلغة	c	البرمجة بلغة
C++		
<pre>#include<iostream.h> main() { 1.int i,array1[5]={10,15,30,32,21}; 2.for (i=0;i<5;i++) {3. array1[i]= array1[i]+5; 4.cout<< array1[i]<<"\t";} }</pre>		<pre>#include<stdio.h> main() { 1.int i,array1[5]={10,15,30,32,21}; 2.for (i=0;i<5;i++) {3. array1[i]= array1[i]+5; 4.printf("%d\t", array1[i]);} }</pre>

توضيح الخطوات:

١.خطوة رقم (٣) نضيف فيها قيمة خمسة لكل عنصر من عناصر المصفوفة

لو كان قيمة العداد هي (i=3) فيكون تنفيذ الخطوة رقم (٣) هكذا

```
array1[3]= array1[3]+5; → array1[3]=32+5 → array1[3]=37
```

٢.وضعنا خطوة رقم (٤) داخل قوس العبارة التكرارية حتى يطبع مباشرة قيمة كل موقع بعد الإضافة .

ترتيب عناصر المصفوفات:

ترتيب الأرقام أو الأحرف تصاعدياً أو تنازلياً يكون بمقارنة كل عنصر من عناصر المصفوفة مع العناصر الذي تليه في المصفوفة فعلى سبيل المثال إذا أردنا ترتيب العناصر تصاعدياً فنقارن كل عنصر في المصفوفة بالعناصر الذي تليه فإذا وجدنا رقم أقل من قيمة العنصر نبدل بين العنصرين (مثلاً إذا كان $array[0]=50, array[3]=2$) فعند البدء بالمقارنة نجد القيمة في الموقع الرابع من المصفوفة هو (2) وفي الموقع الأول هو (50) وقيمة الموقع الرابع أقل لذلك سنبدل قيمة الموقع الرابع بقيمة الموقع الأول ونقارن بقية المواقع بقيمة الموقع الأول الجديدة وهو أصبح (50) ونستمر بالإبدال إلى أن نصل في المقارنة إلى آخر رقم بالمصفوفة ثم نأخذ ثاني عنصر بالمصفوفة ونقارنه بقية العناصر ثم الثالث إلى أن نصل إلى آخر عنصر نجد المصفوفة مرتبة وشاهد هذه خطوات تفصيلية للترتيب لنفرض أن لدينا مصفوفة من خمسة عناصر ونريد ترتيبها تصاعدياً وكانت القيم المخزنة بالمصفوفة كما في الشكل

array[0]	array[1]	array[2]	array[3]	array[4]
45	42	85	6	64



عند الترتيب التصاعدي نضع الرقم الأقل إلى الامام والاعلى الى الخلف في المقارنة والتنازلي بالعكس

للترتيب كما قلنا نقارن كل موقع بجميع المواقع التي تليه هكذا..؟

١. مقارنة الموقع الأول ببقية المواقع التي تليه:

١. مقارنة من الموقع الأول مع الموقع الثاني

array[0]	array[1]	array[2]	array[3]	array[4]
45	42	85	6	64

وجدنا إن قيمة الموقع الثاني أقل من قيمة الموقع الأول لذلك سنبدل بينهما

٢. مقارنة الموقع الأول بالموقع الثالث

array[0]	array[1]	array[2]	array[3]	array[4]
42	45	85	6	64

وجدنا إن قيمة الموقع الثالث أعلى من قيمة الموقع الأول لذلك لا نبدل بينهما

٣. مقارنة الموقع الأول بالموقع الرابع

array[0]	array[1]	array[2]	array[3]	array[4]
42	45	85	6	64

وجدنا أن قيمة الموقع الرابع أقل من قيمة الموقع الأول لذلك سنبدل بينهما

٤. مقارنة الموقع الأول بالموقع الخامس

array[0]	array[1]	array[2]	array[3]	array[4]
6	45	85	42	64

وجدنا أن قيمة الموقع الخامس أعلى من قيمة الموقع الأول لذلك لا نبدل بينهما

٢. مقارنة الموقع الثاني ببقية المواقع التي تليه:

١. مقارنة بين الموقع الثاني مع الموقع الثالث

array[0]	array[1]	array[2]	array[3]	array[4]
6	45	85	42	64

وجدنا أن قيمة الموقع الثالث أعلى من الثاني لذلك سوف لا يبدل بينهما.

٢. مقارنة الموقع الثاني مع الموقع الرابع

array[0]	array[1]	array[2]	array[3]	array[4]
6	45	85	42	64

وجدنا أن قيمة الموقع الرابع أقل من الموقع الثاني لذلك سنبدل بينهما

٣. مقارنة الموقع الثاني بالموقع الخامس

array[0]	array[1]	array[2]	array[3]	array[4]
6	42	85	45	64

وجدنا أن قيمة الموقع الخامس أعلى من قيمة الموقع الثاني لذلك سوف لا يبدل بينهما

٣. مقارنة الموقع الثالث ببقية المواقع التي تليه:

١. مقارنة الموقع الثالث بالموقع الرابع

array[0]	array[1]	array[2]	array[3]	array[4]
6	42	85	45	64

وجدنا قيمة الموقع الرابع أقل من قيمة الموقع الثالث لذلك سنبدل بينهما

٢. مقارنة الموقع الثالث بالموقع الخامس

array[0]	array[1]	array[2]	array[3]	array[4]
6	42	45	85	64

وجدنا قيمة الموقع الخامس أقل من قيمة الموقع الرابع لذلك سنبدل بينهما

٣. مقارنة الموقع الرابع ببقية المواقع التي تليه: يقارن فقط بالموقع الخامس لأنه آخر موقع وسنجد أن قيمة

الموقع الخامس أقل من قيمة الموقع الرابع لذلك سنبدل بينهما

array[0]	array[1]	array[2]	array[3]	array[4]
6	42	45	85	64

قيمة الموقع الخامس أقل من قيمة الموقع الرابع لذلك سنبدل بينهما وستصبح المصفوفة مرتبة بشكل التالي

array[0]	array[1]	array[2]	array[3]	array[4]
6	42	45	64	85

الآن لنحول الكلام الذي كتبناه والمخططات إلى مثال

مثال: برنامج لترتيب عناصر المصفوفة يدخلها المستخدم تصاعديا..؟

تحليل: كيف نرتب برمجيا؟ علمنا انه بالترتيب نقارن كل موقع بجميع المواقع التي تليه لذلك سنحتاج إلى عدادان الأول خاص بالمرور على كل موقع مرة واحدة وعداد داخلي يقارن كل عنصر مرة عليه العداد الخارجي بجميع العناصر التي تليه في المصفوفة وأيضا وجد رقم اقل منه يبدل بينهما.

C++	البرمجة بلغة	C	البرمجة بلغة
<pre>#include<iostream.h> int main() {1.int array[5],i,j; 2. int item=0; 3.cout<<"Here is the Array befor sorted enter it\n"; 4. for (i=0;i<5;i++) 5.cin>>array[i] ; 6. for (i=0;i<5-1;i++) 7.for (j=i; j<5;j++) 8.if (array[j] <array[i]){ 9.item =array[j]; 10.array[j]=array[i]; 11. array[i]= item ;} 12.cout<<"Here is the Array after sorted\n"; 13.for (i=0;i<5;i++) 14.cout<<array[i]<<"\n";}</pre>		<pre>#include<stdio.h> int main() {1.int array[5],i,j; 2. int item=0; 3.printf("Here is the Array befor sorted enter it\n") ; 4. for (i=0;i<5;i++) 5.scnaf("%d",&array[i]) ; 6. for (i=0;i<5-1;i++) 7.for (j=i; j<5;j++) 8.if (array[j] <array[i]){ 9.item =array[j]; 10.array[j]=array[i]; 11. array[i]= item ;} 12. printf("Here is the Array after sorted\n"); 13.for (i=0;i<5;i++) 14.printf("%d\n",array[i]);}</pre>	

توضيح الخطوات:

١. خطوة رقم (١) عرفنا مصفوفة حجمها خمسة ومتغير للعداد الخارجي (i) ومتغير آخر للعداد الداخلي (j)

٢. خطوة رقم (٢) عرفنا متغير لتخزين قيمة احد المتغيرين عند الإبدال لكي لا تضيع قيمة احدهما

٣. خطوة رقم (٣) هي رسالة طباعيه تطلب من المستخدم إدخال المصفوفة وخطوة رقم (٤ و ٥) قراءة المصفوفة

٤. خطوة رقم (٦) هو اعداد خارجي يبدأ بالعد من الصفر إلى الموقع القبل الأخير لكي يرتبهم ولا يعد الموقع الأخير لان الموقع الأخير مرتب أصلا من قبل الخطوة التي تسبقه لذلك نرى شرط التوقف في العداد هو (i<5-1).

أي في كل عدة لهذا العداد تتكرر الخطوات رقم (٧ و ٨ و ٩ و ١٠ و ١١)

٥. خطوة رقم (٧) هو اعداد يبدأ بالعد من قيمة العداد الخارجي إلى أربعة أي إذا عد العداد الخارجي صفر سيعد العداد الداخلي من صفر إلى أربعة لكي يقارن الموقع الأول بجميع المواقع التي تليه. وإذا عد العداد الخارجي واحد سيعد العداد الداخلي من واحد إلى أربعة لكي يقارن المواقع الثاني بجميع المواقع التي تليه ويستمر إلى موقع الرابع

٦. خطوة رقم (٨) هو شرط إذا كان قيمة العنصر اقل من قيمة احد المواقع التي تليه يبدل بينهم في خطوة رقم (٩ و ١٠ و ١١)

٧. خطوات رقم (١٢ و١٣ و١٤) هوا عملية طباعة للمصفوفة بعد الترتيب هكذا

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
Here is the Array befor sorted enter it
45
42
85
6
64
Here is the Array after sorted
6
42
45
64
85
```

القيمة الوسطية بعد الترتيب تقع منتصف عناصر المصفوفة وبما إن حجم مصفوفتنا خمسة فالقيمة الوسطية هي الموقع الثالث أي (array[2])



لو أردنا ترتيب المصفوفة تنازليا نفس الطريقة فقط نبدل الشرط في خطوة رقم (٨) إلى

```
كود
8.if (array[j] >array[i])
```



لو أردنا ترتيب فقط المواقع الثلاث الأولى ولاتهنما البقية فقط نغير شرط التوقف في خطوة رقم (٦ و٧) نبدل رقم خمسة ب ثلاثة هكذا (وضعنا ثلاثة عند شرط التوقف لكي يرتب فقط الموقع الثلاثة الأولى)

```
كود
6. for ( i=0;i<3-1;i++)
7.for ( j=i; j<3;j++)
```



لو أردنا ترتيب المواقع الزوجية في المصفوفة ترتيبا تصاعديا فقط نغير هاتان الخطوتان

```
كود
6. for ( i=0;i<5-1;i=i+2)
7.for ( j=i; j<5;j=j+2)
```



مثال: برنامج لمعرفة هل الرقم متناظر (أي يقرأ من الجهتين) ويكون عدد الأرقام خمسة مثلا ١٢٣٢١

تحليل: لكي نعرف هل الرقم متناظر هنا في مصفوفة خماسية يجب أن يساوي محتويات الموقع الأول محتويات الموقع الخامس والثاني يساوي الرابع (المثال التالي ليس يتحقق فقط من تناظر الأرقام بل الحروف أيضا)

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> #include<conio.h> main() {1.int i, bol=0, a[5]; 2.for(i=0;i<5;i++) 3.a[i]=getche(); 4.for(i=0;i<5;i++) 5.if(a[i]!= a[5-i-1]) 6.bol=1; 7. if(bol==0) 8. cout<<"\nthe data is symmetric"; 9.else 10.cout<<"\nthe data is no symmetric";}</pre>	<pre>#include<stdio.h> #include<conio.h> main() {1.int i, bol=0, a[5]; 2.for(i=0;i<5;i++) 3.a[i]=getche(); 4.for(i=0;i<5;i++) 5.if(a[i]!= a[5-i-1]) 6.bol=1; 7. if(bol==0) 8. printf("\nthe data is symmetric"); 9.else 10.printf("\nthe data is no symmetric");}</pre>

توضيح الخطوات:

- خطوة رقم (٣) هي دالة لإدخال حرف وليس رقم ورغم أنهما ستدخل بشكل حرف ومداخلتنا قد تكون أرقام فلا يهم ففي المقارنة فبدلاً ما يقارن الرقم (١) يقارن الاسكي كود له أي (٥٠) فلا يضر لكن للجمالية استخدمنا هذه الدالة
- خطوة رقم (٥) يقارن العناصر الأول بالأخر والثاني بالقبل الأخير وهكذا ماداماً متساويين يكون المتغير (bol=0) فإذا كانا غير متساويين ويتحقق شرط خطوة رقم (٥) وينفذ خطوة (٦) ستتحول قيمته إلى واحد
- خطوة رقم (٧) يتحقق من قيمة المتغير (bol) إذا بقيت ثابتة حسب تعريفها طبع رسالة إن الرقم متناظر وإذا تغيرت إلى واحد يطبع رسالة أن الرقم غير متناظر

```
(Inactive C:\TCWIN45\BIN\NONAME01.
18581
the data is symmetric
```

```
(Inactive C:\TCWIN45\BIN\NON
arhra
the data is symmetric
```

مثال: مصفوفة حجمها (٥) اسمها (a) ضع العناصر الزوجية لها في مصفوفة (b) والأعداد الفردية في (c)

تحليل: الموضوع بسيط جدا لديك ثلاث مصفوفات نفس الحجم أو بالأحرى نفس حجم المصفوفة (a) لأنها قد تكون جميع الإعدادات فردية أو زوجية. وبعد أن ندخل المصفوفة (a) نكون شرط إذا كان عددها فردي نضعه في (c) وإذا كلن زوجي نضعها في (b) ولكل مصفوفة عداد خاص بها.

C++	البرمجة بلغة c	البرمجة بلغة
<pre>#include<iostream.h> main() {1.int count_a, count_b=0, count_c=0,a[5],b[5], c[5]; 2.for(count_a=0; count_a<5; count_a++) 3.cin>>a[count_a]; 4.for(count_a=0; count_a<5; count_a++) 5.if (a[count_a] %2==0) {6.b[count_b]=a[count_a]; 7.count_b= count_b+1;} 8.else {9.c[count_c]=a[count_a]; 10.count_c= count_c+1;} 11.cout<<"items in array b is\n"; 12.for(count_a=0; count_a< count_b; count_a++) 13.cout<< b[count_a]<<"\t"; 14.cout<<"\nitems in array c is\n"; 15.for(count_a=0; count_a< count_c; count_a++) 16.cout<< c[count_a]<<"\t";}</pre>	<pre>#include<stdio.h> main() {1.int count_a, count_b=0, count_c=0,a[5],b[5], c[5]; 2.for(count_a=0; count_a<5; count_a++) 3.scanf("%d",&a[count_a]); 4.for(count_a=0; count_a<5; count_a++) 5.if (a[count_a] %2==0) {6.b[count_b]=a[count_a]; 7.count_b= count_b+1;} 8.else {9.c[count_c]=a[count_a]; 10.count_c= count_c+1;} 11.printf("items in array b is\n"); 12.for(count_a=0; count_a< count_b; count_a++) 13.printf("%d\t", b[count_a]); 14. printf("\nitems in array c is\n"); 15.for(count_a=0; count_a< count_c; count_a++) 16. printf("%d\t", c[count_a]);}</pre>	

توضيح الخطوات:

١. خطوة رقم (١) عرفنا عداد للمصفوفة (a) وهو (count_a) و عداد للمصفوفة (b) وهو (count_b) و عداد للمصفوفة (c) وهو (count_c) و عرفنا المصفوفات بنفس الأحجام

٢. خطوة رقم (٢ و ٣) هي قراءة للمصفوفة (a) من شاشة التنفيذ

٣. خطوة رقم (٤) هو عداد لاختبار عناصر المصفوفة (a) تتبعه الخطوات رقم (5—11) كلها داخله التكرار

٤. خطوة رقم (٥) هو شرط التحقق إذا كان رقم المصفوفة زوجية سيوضع في مصفوفة (b) في خطوة رقم (٦) ونزود عداد هذه المصفوفة في خطوة رقم (٧) (لكل مصفوفة عدادها لكي نميز بينهم بالإضافة حتى تكون الإضافة صحيحة)

٥. خطوة رقم (٨) هو شرط التحقق إذا لم يكون رقم المصفوفة زوجي سيوضع في مصفوفة (c) في خطوة رقم (9) ونزيد عداد هذه المصفوفة في خطوة رقم (10) (لكل مصفوفة عدادها لكي نميز بينهم بالإضافة حتى تكون الإضافة صحيحة)

٦. خطوة رقم (١٢) هو عداد يبدأ بالصفري وينتهي بأقل من عداد المصفوفة (b) لكي نطبع فقط المواقع التي تحتوي على عناصر في هذه المصفوفة ولانطبع المواقع الفارغة اعتمادا على عدادها الذي يزداد بعد كل إضافة في خطوة رقم (7)

٧. خطوة رقم (١٥) هو عداد يبدأ بالصفري وينتهي بأقل من عداد المصفوفة (c) لكي نطبع فقط المواقع التي تحتوي على عناصر في هذه المصفوفة ولانطبع المواقع الفارغة اعتمادا على عدادها الذي يزداد بعد كل إضافة في خطوة رقم (١٠)

هذا ما سيظهر في شاشة التنفيذ

```
(Inactive C:\TCWIN45\BIN\NONAM
4 5 3 2 4 5
items in array b is
4      2      4
items in array c is
5      3
```

من الآن فصاعدا متى ما قال لك قسم عناصر مصفوفة إلى مصفوفتين استخدم نفس هذا الأسلوب فقط الشرط في خطوة رقم خمسة يتغير أو تغير آخر بسيط لكن تبقى الهيكلية تقريبا نفسها.....!

المصفوفات ثنائية الأبعاد:

لا تختلف المصفوفات الثنائية كثيرا عن المصفوفة أحادية الأبعاد. تختلف فقط في العنونة فبدلا من (index) واحد نستخدم اثنان واحد للصفوف وآخر للأعمدة (مثلا مصفوفة (3*2) أي مكونة من ثلاث صفوف وعموديين)

شكل مصفوفة (3*2)	عمود 0	عمود 1
صف 0 →	(0,0)	(0,1)
صف 1 →	(1,0)	(1,1)
صف 2 →	(2,0)	(2,1)

مهم

العنونة تكون (رقم العمود، رقم الصف) وان العنونة الأعمدة والصفوف تبدأ من الصفر وليس من الواحد أي لو أردنا الوصول إلى عنصر في الصف الثاني العمود الأول تكون عنونه (1,0).

والمصفوفة الثنائية أيضا مجموعة خلايا متتالية في الذاكرة تحجز لغرض تخزين معلومات معينة في داخلها كأن نخزن في داخلها أرقام أو أحرف وتبقى هذه القيم مخزنة داخل المصفوفة حتى نغلق البرنامج إذا لم نغيرها داخل البرنامج. المصفوفات الثنائية: يجب الإعلان عن عدد المواقع التي نحتاجها في العمل في بداية البرنامج حتى يحجزها المترجم للمصفوفة ولا يخزن قيم أخرى داخل هذه المواقع تبقى محجوزة فقط لعناصر المصفوفة ويكون الإعلان عليها هكذا

هيكلية

Type of array `arrayname[row,columns]`

- `(arrayname)` هو اسم المصفوفة التي سنتعامل معه في البرنامج أي اسم ممكن
- `(row)` هو عدد صفوف المصفوفة ، `(columns)` هو عدد الأعمدة لأننا نتعامل مع مصفوفة ثنائية الأبعاد
- `(array of Type)` هو نوع المصفوفة التي سوف نعرفها قد تكون حرفية أو رقمية . لو عرفنا مصفوفة من نوع `(integer)` جميع عناصرها تكون `(integer)` ولا يجوز تخزين أحرف في داخلها

مثال: تعريف مصفوفة من نوع `(integer)` حجمها (3*2) واسمها `(first_array)`.

هيكلية

`int first_array [3][2];`

لنرسم المصفوفة هذه ذات حجم ثلاث صفوف وعمودان

شكل مصفوفة (3*2)	عمود 0	عمود 1
صف 0 →	<code>first_array [0][0]</code>	<code>first_array [0][1]</code>
صف 1 →	<code>first_array [1][0]</code>	<code>first_array [1][1]</code>
صف 2 →	<code>first_array [2][0]</code>	<code>first_array [2][1]</code>

مهم

كما تلاحظ في العنونة كيف تكون: عمود رقم (0) وهو العمود الأول وعمود رقم (1) وهو العمود الثاني وصف رقم (0) وهو الصف الأول و صف رقم (1) وهو الصف الثاني و صف رقم (2) وهو الصف الثالث.

وتخزين هذه العناصر في الذاكرة يكون نفس طريقة تخزين المصفوفة الأحادية لكن هنا يخزن صف وبعده صف آخر بالتسلسل إلى أن تنتهي الصفوف.

لنفرض أن تخزين أول عنصر في المصفوفة بدء بموقع (18126) ويكون شكل مواقع الذاكرة هكذا



مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0][0]	18126	
first_array [0][1]	18127	
first_array [1][0]	18128	
first_array [1][1]	18129	
first_array [2][0]	18130	
first_array [2][1]	18131	
	18132	data



لو تلاحظ أن أول عنصر للمصفوفة وهو (**first_array [0][0]**) يحجز الموقع (18126) وتخزن فيه قيمة أول عنصر ويليه بقية العناصر بالتتالي أي (**first_array [0][1]**) يحجز الموقع (18127) وبالتتالي....؟

عنونة عناصر المصفوفة تبدأ من الصفر للصف والعمود كما تلاحظ أول موقع بالمصفوفة هو (**first_array [0][0]**)

هيكلية

first_array [index_row][index_columns]

- (**index_row**) هو عنوان الصف الذي نريد أن نصل إليه ويمثل دائما بالمتغير (i).
- (**index_columns**) هو عنوان العمود الذي نريد أن نصل إليه ويمثل دائما بالمتغير (j).

مثال: لو أردنا وضع (79) في المصفوفة في الصف الثاني العمود الأول نكتب هكذا

هيكلية

first_array [1][0]=79;

لو تلاحظ أن (**index_row=1**) وليس (2) لأنه كما قلنا تسلسل صفوف المصفوفة تبدأ من الصفر وليس من الواحد أي الصف الثاني في الحقيقة هو صف رقم واحد في البرمجة ولاحظ المخطط في الصفحة السابقة

لو تلاحظ أن (**index_columns=0**) وليس (1) لأنه كما قلنا تسلسل صفوف المصفوفة تبدأ من الصفر وليس من الواحد أي العمود الأول في الحقيقة هو عمود رقم صفر في البرمجة ولاحظ المخطط في الصفحة السابقة

لذلك بعد أن وضعنا رقم (79) بالموقع الثالث يصبح شكل المصفوفة في مواقع الذاكرة هكذا

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0][0]	18126	
first_array [0][1]	18127	
first_array [1][0]	18128	79
first_array [1][1]	18129	
first_array [2][0]	18130	
first_array [2][1]	18131	
	18132	data

محتويات
الموقع بعد
الإضافة

هذه المواقع محجوزة
للمصفوفة
first_array

لو أردنا طباعة محتويات الصف الثالث العمود الثاني

```
C++ الكود بلغة
Cout<< first_array [2][1];
```

```
C الكود بلغة
Printf("%d", first_array [2][1]);
```

لو أردنا تعريف مصفوفة من نوع float نفس طريقة مصفوفة السابقة وكذلك بقية الأنواع

```
هيكلية
float first_array [5][3];
```

نستطيع إعطاء عناصر المصفوفة قيم وقت التعريف هكذا

```
هيكلية
int first_array [2][2]={{34,26},{43,23}};
```

فيكون شكل المصفوفة في مقطع الذاكرة وهي مخزن فيها العناصر هكذا

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0][0]	18126	34
first_array [0][1]	18127	26
first_array [1][0]	18128	43
first_array [1][1]	18129	23

إعطاء قيم لعناصر
المصفوفة وقت التعريف
يكون بوضع عناصر كل
صف بين قوسين ووضع
فارزة بين كل صفيين

هذه المواقع
محجوزة للمصفوفة
first_array
ومحتوياتها



لو أردنا وضع رقم معين في جميع مواقع عناصر المصفوفة وقت التعريف نكتب هكذا
مثلا إذا أردنا وضع رقم صفر في جميع مواقع مصفوفة حجمها (2*2)

هيكلية

```
int first_array [2][2]={0};
```



تعريف وإدخال المصفوفة الثنائية: هذه الخطوات ثابتة في تعريف وإدخال أي مصفوفة ثنائية لا تتغير ابد
مثال: مصفوفة حجمها (4*3) تعريفها وإدخالها

الكود باللغة C++

```
1.int first_arr[4][3];
2.for (i=0;i<4;i++)
3.for (j=0;j<3;j++)
4.cin>> first_array[i][j];
```

الكود باللغة C

```
1.int first_arr[4][3];
2.for (i=0;i<4;i++)
3.for (j=0;j<3;j++)
4 scanf("%d",& first_array[i][j]);
```

الذي يتغير بين سؤال وآخر فقط المكتوب بخط عريض .

1. في خطوة رقم (1) عرفنا نوع المصفوفة ممكن أن يتبدل نوعها حسب الطلب وحجمها حسب السؤال قال في السؤال (4*3) فكونا مصفوفة عدد صفوفها أربعة وعدد الأعمدة ثلاثة
2. خطوة رقم (2) فقط شرط عدد الصفوف يتبدل هنا أربع صفوف كتبنا اصغر من أربعة وإذا خمسة نكتب اصغر من خمسة وكذلك بقية الأرقام
3. خطوة رقم (2) فقط شرط عدد الأعمدة يتبدل هنا ثلاث أعمدة كتبنا اصغر من ثلاثة وإذا أربعة نكتب اصغر من أربعة وكذلك بقية الأرقام
4. خطوة الإدخال أيضا تبقى ثابتة يدخل صف صف بالتسلسل



لحل أي سؤال يتعلق بالمصفوفات الثنائية نرسمها أولا قبل الحل فلو جاء على سبيل المثال في السؤال مصفوفة حجمها (5*5) نرسم هذه المصفوفة أولا هكذا

القطر الرئيسي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

الآن نستطيع حل جميع الأسئلة عن المصفوفة هذه فعلى سبيل المثال لو قال جد عناصر القطر الرئيسي وهو الذي نحن واضعون خط عليه وهو منتصف المصفوفة ما الذي يمكننا من تمييزه عن غيره من الصفوف؟؟
الذي يميز عناصر القطر الرئيسي عن غيره من العناصر في المصفوفة هو قيمة الصف تساوي قيمة العمود لاحظ القيم (4,4), (3,3), (2,2), (1,1), (0,0) فيكون الشرط هو ثابت في كل مصفوفة نريد إيجاد قطر الرئيسي.

البرمجة

```
if(i==j)
```

```
// print the diagonal هنا نقوم بطباعة العناصر المصفوفة في قطر الرئيسي
```

قيمة (i) هي عنوان الصف وقيمة (j) هي عنوان العمود

** الحصول على عناصر القطر الثانوي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

القطر الثانوي

الذي يميز عناصر القطر الثانوي عن غيرها من العناصر في المصفوفة أن قيمة الصف عند جمعها بقيمة العمود تساوي أربعة لاحظ $(4+0=4), (3+1=4), (2+2=3), (1+3=4), (0+4=4)$ فعلا حاصل جمع رقم الصف مع رقم العمود في القطر الثانوي يساوي أربعة لكن يختلف من مصفوفة إلى أخرى هذه القيمة الجمع حسب إبعاد المصفوفة أنت ارسم المصفوفة حسب حجمها وستجد العلاقة كما وجدناها نحن الآن. فيكون الشرط هكذا وهو غير ثابتة قيمة الأربعة

البرمجة

```
if(i+j=4)
```

** الحصول على العناصر فوق القطر الرئيسي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

العناصر فوق القطر الرئيسي

الذي يميز عناصر فوق القطر الرئيسي عن غيرها من العناصر في المصفوفة أن قيمة العمود أكبر من قيمة الصف لاحظ التالي $(0,1), (0,2), (0,3), (0,4), (1,2), (1,3), (1,4), (3,4)$ فعلا قيم الأعمدة جميعها أعلى من قيم المصفوفة في القطر الثانوي فيكون الشرط هكذا وهو ثابت

البرمجة

```
if(i<j)
```

**الحصول على العناصر تحت القطر الرئيسي

العناصر
تحت القطر
الرئيسي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

الذي يميز عناصر تحت القطر الرئيسي عن غيرها من العناصر في المصفوفة أن قيمة كل العمود اقل من قيمة كل الصف لاحظ التالي ((1,0),(2,0),(3,0),(4,0),(2,1),(3,1),(4,1),(4,3)) فعلا قيم الأعمدة جميعها أعلى من قيم المصفوفة في القطر الثانوي فيكون الشرط هكذا وهو ثابت

البرمجة

if(i>j)

**الحصول على العناصر فوق القطر الثانوي

العناصر
فوق القطر
الثانوي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

الذي يميز عناصر فوق القطر الثانوي عن غيرها من العناصر في المصفوفة أن قيمة العمود عند جمعها مع قيمة الصف لا يتجاوز ناتج الجمع ثلاثة بينما بقية العناصر تتجاوز ثلاثة لاحظ التالي ((0,0),(0,1),(0,2),(0,3),(1,0),(1,1),(1,2),(2,0),(2,1),(3,0)) مصفوفة إلى أخرى حسب حجمها

البرمجة

if(i+j<4)

**الحصول على العناصر تحت القطر الثانوي

العناصر
تحت القطر
الثانوي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

الذي يميز عناصر تحت القطر الثانوي عن غيرها من العناصر في المصفوفة أن قيمة العمود عند جمعها مع قيمة الصف يتجاوز ناتج الجمع فوق الأربعة بينما بقية العناصر لا تتجاوز الأربعة لاحظ التالي
(1,4),(2,4),(2,3),(3,4),(3,3),(3,3),(4,4),(4,3),(4,2),(4,1) فيكون الشرط هكذا وهو غير ثابت من مصفوفة إلى أخرى حسب حجمها

```
البرمجة
if(i+j>4)
```

**الحصول على العناصر العمود الأول والرابع

عناصر العمود الأول والرابع	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
	(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

ما يميز عناصر العمود الأول عن غيره من الأعمدة أن قيمة (j=0) ما يميز عناصر العمود الرابع عن غيره من الأعمدة أن قيمة (j=3) فشرط الحصول عليهما وهو ثابت

```
البرمجة
if((j==0) || (j==3))
```

الحصول على عناصر الصف الأول و الثالث والرابع

عناصر الصف الأول والثالث والرابع	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
	(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

ما يميز عناصر الصف الأول عن غيره من الصفوف أن قيمة (i=0) وما يميز عناصر الصف الثالث عن غيره من الصفوف أن قيمة (i=2) و ما يميز عناصر الصف الرابع عن غيره من الصفوف أن قيمة (i=3) فشرط الحصول عليهما وهو ثابت

```
البرمجة
if((i==0) || (i==2) || (i==3))
```


مثال: مصفوفة مربعة (5*5) أطلع فقط عناصر القطر الرئيسي ؟

تحليل: نعلم أن شرط الحصول على عناصر القطر الرئيسي في أي مصفوفة هو ثابت وهو $(i==j)$ وللكتير من المعلومات راجع المخطط السابق عن الحصول على القطر الرئيسي

C++	البرمجة بلغة c
<pre>#include<iostream.h> main() {1.int i; 2.int j; 3.int a[5][5]; 4.for(i=0;i<5;i++) 5.for(j=0;j<5;j++) 6.cin>>a[i][j]; 7.for(i=0;i<5;i++) 8.for(j=0;j<5;j++) 9.if(i==j) 10.cout<<a[i][j]<<"\n";}</pre>	<pre>#include<stdio.h> main() {1.int i; 2.int j; 3.int a[5][5]; 4.for(i=0;i<5;i++) 5.for(j=0;j<5;j++) 6.scanf("%d",&a[i][j]); 7.for(i=0;i<5;i++) 8.for(j=0;j<5;j++) 9.if(i==j) 10.printf("%d \n ",a[i][j]);}</pre>

توضيح الخطوات:

١. خطوة رقم (٣ و٤ و٥ و٦) هي تعريف وإدخال المصفوفة وهي ثابتة في كل برنامج ندخل فيه مصفوفة حسب حجم المصفوفة فقط تتغير قيم توقف العدادان كما قلنا سابقا

٢. خطوة رقم (٧ و٨) عدادان يمران على جميع عناصر المصفوفة بالتسلسل صف صف لكي نتحقق من الشرط فأي رقم يقع في القطر الرئيسي سيكون عندها قيمة العدادان $(i==j)$ لذلك سيحقق الشرط في الخطوة رقم (٩) ويطبع الرقم في الخطوة رقم (١٠)

```
(Inactive C:\TCWIN45\BIN\NO
1 2 3 4 5
4 5 4 3 4
4 5 6 7 5
5 6 7 6 5
5 6 7 8 6)
```

كما في الشكل



* جميع الأمثلة التي حليناها في المصفوفات الأحادية نستطيع استخدامها في المصفوفات الثنائية فقط نبذل البعد الواحد إلى بعدين والعداد إلى عدادين

* لو أردنا ضرب جميع عناصر المصفوفة برقم معين مثلا (٢) نضرب هكذا

```
البرمجة
a[i][j]=2*a[i][j];
```

وكذلك بقية العمليات جمع عنصر مع كل عناصر المصفوفة أو ضربه

لا تنسى أن تضع عداد للصفوف وعداد لأعمدة المصفوفة قبل هذه الخطوة ليضرب عنصر عنصر

مثال: مصفوفة مربعة (5*5) اجمع العناصر فوق القطر الرئيسي وجمع العناصر تحته وجمع العناصر فوق القطر الثانوي وتحت كل على حدة

تحليل: ارجع إلى خطوات التحليل السابقة وستفهم الشروط المراد إيجادها وطريقة تحليلها ورسم المصفوفة .

C++	البرمجة بلغة C
<pre>#include<iostream.h> main() {1.int i,j,sum,sum1,sum2,sum3; 2.sum=sum1=sum2=sum3=0; 3.int a[5][5]; 4.for(i=0;i<5;i++) 5.for(j=0;j<5;j++) 6.cin>>a[i][j]; 7.for(i=0;i<5;i++) 8.for(j=0;j<5;j++) { 9.if(i<j) 10.sum+=a[i][j]; 11.if(i>j) 12.sum1+=a[i][j]; 13. if((i+j)<4) 14sum2+=a[i][j]; 15if((i+j)>4) 16.sum3+=a[i][j];} 17.cout<<"\n sum above secondary diagonal= "<<sum2; 18.cout<<"\n sum above main diagonal= "<<sum; 19.cout<<"\n sum under main diagonal= "<<sum1; 20.cout<<"\n sum under secondary diagonal= "<<sum3;}</pre>	<pre>#include<stdio.h> main() {1.int i,j,sum,sum1,sum2,sum3; 2.sum=sum1=sum2=sum3=0; 3.int a[5][5]; 4.for(i=0;i<5;i++) 5.for(j=0;j<5;j++) 6.scanf("%d",&a[i][j]); 7.for(i=0;i<5;i++) 8.for(j=0;j<5;j++) { 9.if(i<j) 10.sum+=a[i][j]; 11.if(i>j) 12.sum1+=a[i][j]; 13. if((i+j)<4) 14sum2+=a[i][j]; 15if((i+j)>4) 16.sum3+=a[i][j];} 17.printf("\n sum above secondary diagonal= %d ",sum2); 18. printf("\n sum above main diagonal=%d ",sum); 19.printf("\n sum under main diagonal= %d ",sum1); 20.printf("\n sum under secondary diagonal=%d ",sum3);}</pre>

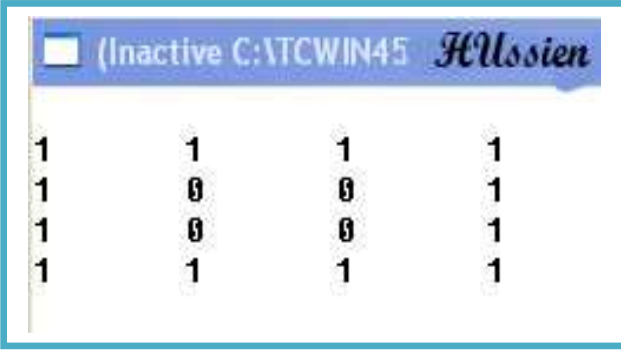
توضيح الخطوات:

١. خطوة رقم (٣ و ٤ و ٥ و ٦) هي تعريف وإدخال المصفوفة وهي ثابتة في كل برنامج ندخل فيه مصفوفة حسب الحجم
٢. خطوة رقم (٧ و ٨) عدادان يمران على جميع عناصر المصفوفة بالتسلسل صف صف لكي نتحقق من الشروط
٣. خطوة رقم (٩) هو شرط لجمع العناصر فوق القطر الرئيسي أي إذا جاء أي عنصر ضمن عناصر فوق القطر الرئيسي سينفذ الخطوة رقم (١٠) لكي يجمعه ببقية عناصر فوق القطر الرئيسي
٤. خطوة رقم (١١) هو شرط لجمع العناصر تحت القطر الرئيسي أي إذا جاء أي عنصر ضمن عناصر تحت القطر الرئيسي سينفذ الخطوة رقم (١٢) لكي يجمعه ببقية عناصر تحت القطر الرئيسي
٥. خطوة رقم (١٣) هو شرط لجمع العناصر فوق القطر الثانوي أي إذا جاء أي عنصر ضمن عناصر فوق القطر الثانوي سينفذ الخطوة رقم (١٤) لكي يجمعه ببقية عناصر فوق القطر الثانوي
٦. خطوة رقم (١٥) هو شرط لجمع العناصر تحت القطر الثانوي أي إذا جاء أي عنصر ضمن عناصر تحت القطر الثانوي سينفذ الخطوة رقم (١٦) لكي يجمعه ببقية عناصر تحت القطر الثانوي

مثال: تكوين مصفوفة (4*4) كما في الشكل

تحليل: كما قلنا سابقا نرسم المصفوفة في البداية

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)



كما نلاحظ كن الرسم أننا نريد أن نضع قيمة واحد في الصف الأول والصف الرابع والعمود الأول والعمود الرابع والمواقع خلافهما نضع فيها صفر. وان ما يميز الصف الأول عن باقي الصفوف أن قيمة (i=0) وما يميز الصف الرابع عن بقية الصفوف أن قيمة (i=3) وان ما يميز العمود الأول عن باقي الأعمدة أن قيمة (j=0) وما يميز العمود الرابع عن بقية الأعمدة أن قيمة (j=3) وسيكون لذلك الشرط يجمع بين هذه الشروط الأربعة كما في البرنامج

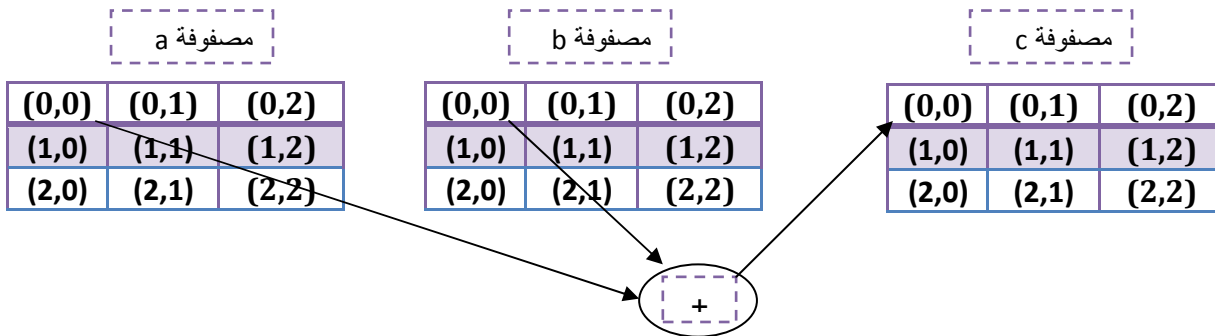
C++	البرمجة بلغة C
<pre>#include<iostream.h> main() { 1.int i,j; 2.int a[4][4]={0}; 3.for(i=0;i<4;i++){ 4.for(j=0;j<4;j++){ 5.if((j==0) (i==3) (j==3) (i==0)) 6.a[i][j]=1; 7.cout<< a[i][j]<<"\t"; 8.cout<<"\n"; } }</pre>	<pre>#include<stdio.h> main() { 1.int i,j; 2.int a[4][4]={0}; 3.for(i=0;i<4;i++){ 4.for(j=0;j<4;j++){ 5.if((j==0) (i==3) (j==3) (i==0)) 6.a[i][j]=1; 7.printf("%d\t", a[i][j]); 8. printf("\n"); } }</pre>

توضيح الخطوات:

- خطوة رقم (٢) عرفنا مصفوفة حجمها (4*4) وخرنا صفر في جميع مواقعها
- خطوة رقم (٣ و ٤) عداد يمر على جميع عناصر المصفوفة صف صف
- خطوة رقم (٥) هو شرط اختيار إذا كان الصف الأول أو الصف الرابع أو العمود الأول أو العمود الرابع سوف ينفذ خطوة رقم (٦) ليضع واحد بد الصفر في الموقع
- خطوة رقم (٧) طبع محتويات المصفوفة
- خطوة رقم (٨) وهي خطوة النزول إلى سطر جديد بعد طباعة صف كامل حتى يكون شكل المصفوفة المطبوعة بالشكل المطلوب ومطابقة لهيكلية المصفوفات الثنائية ولكي نتأكد من موقعها قبل أن تغلق قوس العبارة التكرارية الخاصة بالعداد (i) ضعها في كل برنامج

مثال: برنامج لجمع مصفوفتين حجمهما (3*3)

تحليل: لجمع مصفوفتين نجمع العنصر الأول بالمصفوفة الأولى مع العنصر الأول في المصفوفة الثانية وكذلك البقية



هذا الشكل يمثل جمع المواقع الأول والمواقع البقية نفس الشيء الثاني مع الثاني وبالتتابع

C++	البرمجة بلغة c	البرمجة بلغة
<pre>#include<iostream.h> main() { 1.int i,j; 2.int a[3][3]; 3.int b[3][3]; 4. int c[3][3]; 5.cout<<"enter first matrixs\n"; 6. for(i=0;i<3;i++) 7.for(j=0;j<3;j++) 8.cin>>a[i][j]; 9. cout<<"enter second matrixs\n"; 10.for(i=0;i<3;i++) 11.for(j=0;j<3;j++) 12. cin>>b[i][j]; 13.for(i=0;i<3;i++){ 14. for(j=0;j<3;j++){ 15.c[i][j]=a[i][j]+b[i][j]; 16.cout<<c[i][j]<<"\t"; 17.cout<<"\n";}}</pre>	<pre>#include<stdio.h> main() { 1.int i,j; 2.int a[3][3]; 3.int b[3][3]; 4. int c[3][3]; 5.printf("enter first matrixs\n"); 6. for(i=0;i<3;i++) 7.for(j=0;j<3;j++) 8 scanf("%d",&a[i][j]); 9. printf("enter second matrixs\n"); 10.for(i=0;i<3;i++) 11.for(j=0;j<3;j++) 12. scanf("%d",&b[i][j]); 13.for(i=0;i<3;i++){ 14. for(j=0;j<3;j++){ 15.c[i][j]=a[i][j]+b[i][j]; 16. printf("%d\t",c[i][j]); 17. printf("\n");}}</pre>	

توضيح الخطوات:

١. خطوة رقم (٦ و ٧ و ٨) هي عملية إدخال للمصفوفة الأولى. خطوة رقم (١٠ و ١١ و ١٢) إدخال المصفوفة الثانية

٢. خطوة رقم (١٣ و ١٤) هو اعدادا يمر على جميع عناصر المصفوفة صف ليجمع في خطوة رقم (١٥) العنصر الأول في المصفوفة الأولى مع العنصر الأول بالمصفوفة الثانية والثاني بالتتابع

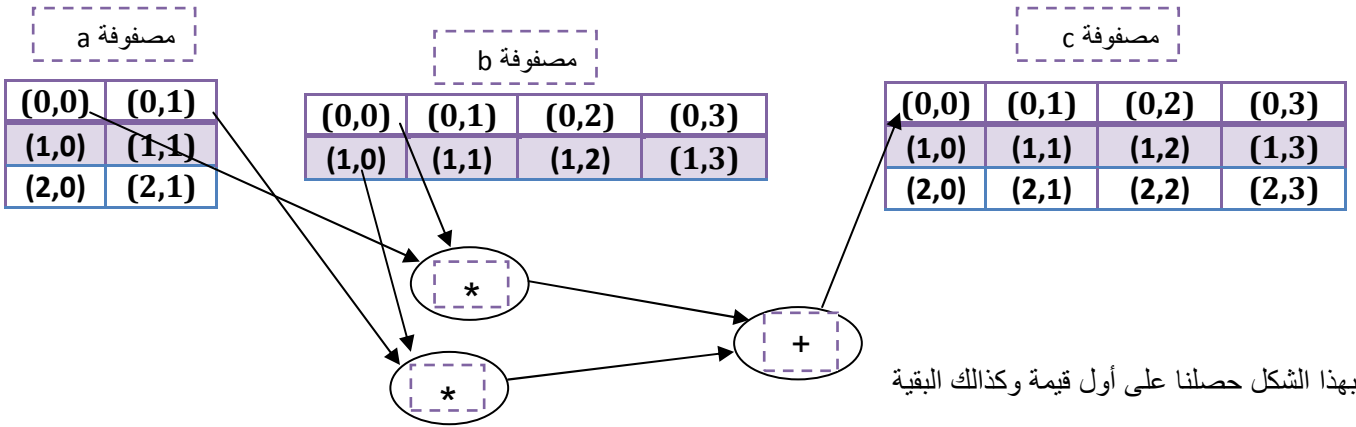
٣. خطوة رقم (١٦) هو عملية طباعة لعناصر المصفوفة

عملية طرح مصفوفتين نفس الخطوات السابقة فقط نبدل خطوة (١٥) بالتالي

```
البرمجة
15.c[i][j]=a[i][j]-b[i][j];
```

مثال: برنامج لضرب مصفوفتين $(3*2) * (2*4)$ ؟.

تحليل: لضرب مصفوفتين $(3*2) * (2*4)$ يجب أن يكون الناتج مصفوفة بحجم $(3*4)$ حسب قواعد الضرب



C++	البرمجة بلغة C
<pre>#include<iostream.h> main() {1.int i,j,k; 2. int a[3][2]; 3.int b[2][4]; 4. int c[3][4]={0}; //put zero in every location 5.cout<<"enter first matrixs\n" ; 6. for(i=0;i<3;i++) 7.for(j=0;j<2;j++) 8.cin>>a[i][j]; 9.cout<<"enter second matrixs\n" ; 10. for(i=0;i<2;i++) 11.for(j=0;j<4;j++) 12.cin>>b[i][j]; 13.for(i=0;i<3;i++){ 14.for(j=0;j<4;j++){ 15.for(k=0;k<2;k++) 16.c[i][j]+=a[i][k]*b[k][j]; 17. cout<<c[i][j]<<"\t" ;} 18.cout<<"\n";} }</pre>	<pre>#include<stdio.h> main() {1.int i,j,k; 2. int a[3][2]; 3.int b[2][4]; 4. int c[3][4]={0}; //put zero in every location 5. printf("enter first matrixs\n"); 6. for(i=0;i<3;i++) 7.for(j=0;j<2;j++) 8. scanf("%d",&a[i][j]); 9.printf("enter second matrixs\n"); 10. for(i=0;i<2;i++) 11.for(j=0;j<4;j++) 12. scanf("%d",&b[i][j]); 13.for(i=0;i<3;i++){ 14.for(j=0;j<4;j++){ 15.for(k=0;k<2;k++) 16.c[i][j]+=a[i][k]*b[k][j]; 17. printf("%d\t",c[i][j]);} 18.printf("\n");} }</pre>

توضيح الخطوات:

- خطوة رقم (٤) عرفنا مصفوفة ووضعنا صفر في جميع مواقعها لأننا سنخزن فيها نتيجة الضرب وكما ترى في المخطط يضرب ثم يجمع أي توجد عملية جمع أكثر من مرة لذلك يجب تفسير المواقع حتى لا تؤثر على الجمع
- خطوة رقم (١٣ و ١٤) هو التحرك ببعد المصفوفة الجديدة وهي بحجم $(3*4)$ وبعدها خطوة رقم (١٥) هو البعد المفقود الذي سيفقد في عملية الضرب وخطوة رقم (١٦) هو كما موضح في المخطط أي إننا متى ما جاءتنا عملية ضرب مصفوفتين نأخذ عدادان بأبعاد مصفوفة ناتجة وعداد بالبعد المفقود وبعدها خطوة رقم (١٦) أي لو ضربنا المصفوفات التالية $(6*3) * (4*6)$ وناتج يكون مصفوفة $(4*3)$ والبعد المفقود (6)

```
البرمجة
13.for(i=0;i<4;j++){
14.for(j=0;j<3;j++){
15.for(k=0;k<6;k++)
16.c[i][j]+=a[i][k]*b[k][j]
```



ترتيب المصفوفات ثنائية الأبعاد:

لا يختلف ترتيب المصفوفات الثنائية عن ترتيب المصفوفات الأحادية في شيء نفس الصيغة نأخذ أول عنصر في المصفوفة ونقارنه مع بقية العناصر والثاني وبالتالي لذلك راجع ترتيب المصفوفات الأحادية أولاً ...

مثال: ترتيب مصفوفة ثنائية الأبعاد (5*5) ترتيب تنازلياً

C++	البرمجة بلغة C
<pre>#include<iostream.h> int main() {1.int const row=5; 2.int const col=5; 3. int array[row][col]; 4.int i,j,k,x,a ; 5.cout<<"Here is the Array befor sorted\n" ; 6. for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.cin>>array[i][j] ; 9.for(k=0;k<row;k++){ 10. for(l=0;l<col;l++){ 11. for(i=0;i<row;i++){ 12. for (j=0;j<col;j++){ 13. if (array[i][j] < array[k][l]){ 14.x=array[k][l]; 15.array[k][l]=array[i][j]; 16.array[i][j]=x; 17.}} } } } 18. cout<<"Here is the Array after sorted\n" ; 19. for (i=0;i<row;i++){ 20. for (j=0;j<row;j++) 21. cout<<array[i][j]<<"\t"; 22. cout<<"\n" ;} }</pre>	<pre>#include<stdio.h> int main() {1.int const row=5; 2.int const col=5; 3. int array[row][col]; 4.int i,j,k,x,a ; 5.printf("Here is the Array befor sorted\n"); 6. for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.scanf("%d",&array[i][j]); 9.for(k=0;k<row;k++){ 10. for(l=0;l<col;l++){ 11. for(i=0;i<row;i++){ 12. for (j=0;j<col;j++){ 13. if (array[i][j] < array[k][l]){ 14.x=array[k][l]; 15.array[k][l]=array[i][j]; 16.array[i][j]=x; 17.}} } } } 18.printf("Here is the Array after sorted\n"); 19. for (i=0;i<row;i++){ 20. for (j=0;j<row;j++) 21.printf("%d\t",array[i][j]); 22.printf("\n");} }</pre>

توضيح الخطوات:



١. خطوة رقم (١ و ٢) هو الإعلان عن أبعاد المصفوفة وهذه الصيغة مهمة جداً للمطور ففي البرامج السابقة لو أردت فقط نفس المثال فقط تغيير أبعاد المصفوفة لكنت بحاجة لتغيير بعد المصفوفة وشروط توقف العدادات في كل الخطوات أما هنا إذا أردنا أن نغير بعد المصفوفة مثلاً قال رتب مصفوفة (4*4) فقط نغير قيمة (row=4) و (col=4) ويتحول السؤال كله كما نريد بدون أي خطأ

٢. خطوة رقم (٩ و ١٠) عدادان يمران على جميع عناصر المصفوفة لكي يقارن كل عنصر بجميع العناصر التي تليه بواسطة العدادان في الخطوة (١١ و ١٢)

٣. خطوة رقم (١٣ و ١٤ و ١٥ و ١٦) هي عملية أبدال بين موقعين شرحت سابقاً في المصفوفة الأحادية

٤. خطوة رقم (١٩ و ٢٠ و ٢١ و ٢٢) طباعة للمصفوفة بعد الترتيب

مثال: ترتيب صفوف مصفوفة ثنائية الأبعاد (5*5) ترتيب تنازليا

يقارن عناصر الصف الواحد ليرتبها ١.الأول مع الجميع

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

تحليل: لترتيب المصفوفة كل صف على حدة نحن بحاجة لعداد خارجي يقف على عناصر المصفوفة صف صف وليكن اسمه (k) ويبدأ بترتيب عناصر الصف الواحد بمقارنة العنصر الأول مع جميع العناصر التي تليه في الصف الذي عليه المؤشر العداد ثم بعد أن يرتبه ينتقل إلى الصف التالي.

عندما يكون الترتيب على الصف الأول (k=0)

C++	البرمجة بلغة C	البرمجة بلغة C
<pre>#include<iostream.h> int main() {1.int const row=5; 2. int const col=5; 3.int array[row][col]; 4.int i,j,k,x ; 5.cout<<"Here is the Array befor sorted\n" ; 6. for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.cin>>array[i][j] ; 9.for(k=0;k<row;k++) 10.for(i=0;i<row;i++) 11.for (j=0;j<col;j++){ 12. if (array[k][j] <array[k][i]){ 13. x=array[k][j]; 14.array[k][j]=array[k][i]; 15.array[k][i]=x;}} 16. cout<<"Here is the Array after sorted\n" ; 17.for (i=0;i<row;i++){ 18. for (j=0;j<row;j++) 19. cout<<array[i][j]<<"\t"; 20.cout<<"\n" ;} }</pre>	<pre>#include<stdio.h> int main() {1.int const row=5; 2. int const col=5; 3.int array[row][col]; 4.int i,j,k,x ; 5.printf("Here is the Array befor sorted\n"); 6. for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.scanf("%d",&array[i][j]); 9.for(k=0;k<row;k++) 10.for(i=0;i<row;i++) 11.for (j=0;j<col;j++){ 12. if (array[k][j] <array[k][i]){ 13. x=array[k][j]; 14.array[k][j]=array[k][i]; 15.array[k][i]=x;}} 16.printf("Here is the Array after sorted\n"); 17.for (i=0;i<row;i++){ 18. for (j=0;j<row;j++) 19.printf("%d\t",array[i][j]); 20.printf("\n");} }</pre>	<pre>#include<stdio.h> int main() {1.int const row=5; 2. int const col=5; 3.int array[row][col]; 4.int i,j,k,x ; 5.printf("Here is the Array befor sorted\n"); 6. for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.scanf("%d",&array[i][j]); 9.for(k=0;k<row;k++) 10.for(i=0;i<row;i++) 11.for (j=0;j<col;j++){ 12. if (array[k][j] <array[k][i]){ 13. x=array[k][j]; 14.array[k][j]=array[k][i]; 15.array[k][i]=x;}} 16.printf("Here is the Array after sorted\n"); 17.for (i=0;i<row;i++){ 18. for (j=0;j<row;j++) 19.printf("%d\t",array[i][j]); 20.printf("\n");} }</pre>

مهم

توضيح الخطوات:

ماذا سيحصل في خطوة رقم (9—15)؟...

كيف رتبنا صفوف المصفوفة؟ نجد انه يقوم بمقارنة عناصر الصف الواحد ببعضها وترتيبها حيث أن المتغير (k) ينتقل في كل لوب إلى صف جديد بعد أن ينتهي من ترتيب الصف الذي يسبقه وفي المقارنة خطوة رقم (١٢) نثبت الصف بين المصدر والمسار الترتيبي وننقل العداد الخارجي (i) في كل لوب إلى عمود جديد ضمن الصف الواحد ويعمل العداد الداخلي (j) على مقارنة هذا العنصر الذي يؤشر عليه (i) في الصف الواحد بكل العناصر في الأعمدة التي تليه في نفس الصف فإذا وجد فيها اصغر منه يبدله

أتمنى أن تكون قد استوعبت الفكرة (ابومشاري)

مثال: ترتيب أعمدة مصفوفة ثنائية الأبعاد (5*5) ترتيب تنازليا

يقارن عناصر العمود الواحد ليرتبها ١.الأول مع الجميع

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

تحليل: لترتيب المصفوفة كل عمود على حدة نحن بحاجة لعداد خارجي يقف على عناصر المصفوفة عمود عمود وليكن اسمه (k) ويبدأ بترتيب عناصر العمود الواحد بمقارنة العنصر الأول مع جميع العناصر التي تليه في العمود الذي عليه المؤشر العداد ثم بعد إن يرتبه ينتقل إلى العمود التالي.

عندما يكون الترتيب على العمود الأول (k=0)

C++	البرمجة بلغة C	البرمجة بلغة C
<pre>#include<iostream.h> int main() {1.int const row=5; 2. int const col=5; 3.int array[row][col]; 4.int i,j,k,x,l ; 5.cout<<"Here is the Array befor sorted\n" ; 6.for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.cin>>array[i][j] ; 9. for(k=0;k<row;k++) 10.for(i=0;i<row;i++) 11. for (j=0;j<col;j++) { 12. if (array[j][k] <array[i][k]){ 13. x=array[j][k]; 14.array[j][k]=array[i][k]; 15.array[i][k]=x;}} 16. cout<<"Here is the Array after sorted\n" ; 17.for (i=0;i<row;i++){ 18. for (j=0;j<row;j++) 19. cout<<array[i][j]<<"\t"; 20. cout<<"\n" ;}}</pre>	<pre>#include<stdio.h> int main() {1.int const row=5; 2. int const col=5; 3.int array[row][col]; 4.int i,j,k,x,l ; 5.printf("Here is the Array befor sorted\n"); 6.for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.scanf("%d",&array[i][j]); 9. for(k=0;k<row;k++) 10.for(i=0;i<row;i++) 11. for (j=0;j<col;j++) { 12. if (array[j][k] <array[i][k]){ 13. x=array[j][k]; 14.array[j][k]=array[i][k]; 15.array[i][k]=x;}} 16.printf("Here is the Array after sorted\n"); 17.for (i=0;i<row;i++){ 18. for (j=0;j<row;j++) 19.printf("%d\t",array[i][j]); 20.printf("\n");}}</pre>	<pre>#include<stdio.h> int main() {1.int const row=5; 2. int const col=5; 3.int array[row][col]; 4.int i,j,k,x,l ; 5.printf("Here is the Array befor sorted\n"); 6.for (i=0;i<row;i++) 7. for (j=0;j<col;j++) 8.scanf("%d",&array[i][j]); 9. for(k=0;k<row;k++) 10.for(i=0;i<row;i++) 11. for (j=0;j<col;j++) { 12. if (array[j][k] <array[i][k]){ 13. x=array[j][k]; 14.array[j][k]=array[i][k]; 15.array[i][k]=x;}} 16.printf("Here is the Array after sorted\n"); 17.for (i=0;i<row;i++){ 18. for (j=0;j<row;j++) 19.printf("%d\t",array[i][j]); 20.printf("\n");}}</pre>



توضيح الخطوات: . . . مهم

ماذا سيحصل في خطوة رقم (15—9)...

كيف رتبنا أعمدة المصفوفة؟ نجد انه يقوم بمقارنة عناصر العمود الواحد ببعضها وترتيبها حيث أن المتغير (k) ينتقل في كل لوب إلى عمود جديد بعد أن ينتهي من ترتيب العمود الذي يسبقه. وفي المقارنة خطوة رقم (١٢) نثبت العمود بين المصدر والمسار الترتيبي وننقل العداد الخارجي (i) في كل لوب إلى صف جديد ضمن العمود الواحد ويعمل العداد الداخلي (j) على مقارنة هذا العنصر الذي يؤثر عليه (i) في العمود الواحد بكل العناصر في الصفوف التي تليه في نفس العمود الذي عليه المؤشر فإذا وجد فيها اصغر منه يبدله

* لو أردنا ترتيب تصاعدي فقط نبدل علامة الأصغر في شرط المقارنة إلى اكبر

النصوص (string):

١. تمثيلها بالمصفوفات الأحادية الأبعاد:

هي مصفوفات مكونه من سلسلة من الحروف وتخزن بنفس طريقة المصفوفة الاعتيادية ويخزن بعد آخر موقع نخزن فيه المصفوفة الحرف ('\\0') للدلالة على أن السلسلة انتهت.
لو أردنا تكوين مصفوفة حرفيه اسمها (*name*) نخزن فيها اسم شخص وليكن (hussien)

هيكلية تعريف

```
Char name[8]="hussien";
```

أو تكتب هكذا لكن هنا يجب وضع ('\\0') في نهاية السلسلة

هيكلية تعريف

```
Char name[8]={'h','u','s','s','i','e','n','\\0'};
```

نلاحظ إننا حجزنا ثمانية مواقع ورغم أن الاسم مكون من سبعة أحرف لأنه كما قلنا يضيف ('\\0') إلى نهاية السلسلة بالإضافة إلى الأحرف التي أدخلناها (أي دائما نحجز حجم المصفوفة اكبر من ما نحتاجه بواحد)

أي سلسلة تخزن في الذاكرة بالتسلسل أي أن مصفوفة name التي تحوي على "hussien" تخزن هكذا إذا افترضنا انه أول موقع يخزن فيه أول حرف هوا (200) كما لاحظت أن بقية الأحرف يخزنهم بالتتالي.

مواقع خلايا الذاكرة	عنوان المصفوفة	name[0]	name[1]	name[2]	name[3]	name[4]	name[5]	name[6]	
	عنوان الموقع	200	201	202	203	204	205	206	207
	محتواه	h	u	s	s	i	e	n	\\0

*نعلم أن ترقيم المصفوفة في هذه اللغة يبدأ من الصفر أي أن حرف (h) يخزن في موقع صفر وحرف (i) يخزن في موقع ٤

لو أردنا أن نضع حرف (o) بدل حرف (u) وبما انه ثاني موقع في المصفوفة نخزنه هكذا

البرمجة

```
name[1]='o';
```

مواقع خلايا الذاكرة	عنوان المصفوفة	name[0]	name[1]	name[2]	name[3]	name[4]	name[5]	name[6]	
	عنوان الموقع	200	201	202	203	204	205	206	207
	محتواه	h	o	s	s	i	e	n	\\0

هذا الحرف الذي تم أبداله

**المكتبة التي نستخدمها مع (string) هي <string.h>

دوال الإدخال والإخراج لل (string):

في حال إدخال (string) من قبل المستخدم . عبارات الإدخال الاعتيادية تكون غير كافية لإدخالها أو غير ملائمة لدرجة 100%. على سبيل المثال دالة الإدخال الاعتيادية ونحاول إدخال (string) بها ماذا سيحدث

دالة الإدخال بلغة ++C

```
char name[30];
scanf("%s",name);
```

دالة الإدخال بلغة C

```
char name[30];
cin>>name ;
```

الفراغ الأول

سوف يأخذ الجملة المدخلة لكن ليس كلها يأخذها حتى أول فراغ بالإدخال أي لو أدخلنا

Hussien Ahmmed Taleb

لذلك سوف يأخذ المدخلات فقط كلمة (Hussien) وبهمل من الفراغ الأول إلى النهاية



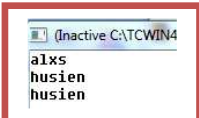
إذا أدخلنا عدة كلمات فما الذي سوف يحدث للذي مهلته دالة الإدخال الأولى..؟

الذي مهلته دالة الإدخال الأولى لن يذهب سدى إنهما يبقى ينتظر عملية إدخال ثانية ليكون هو كمدخل لها) أي في عملية الإدخال الثانية سوف لا يطلب من المستخدم إدخال هو يعتبر ما تبقى من دالة الإدخال الأولى كمدخل لدالة الإدخال الثانية) أي لو رجعنا إلى المثال السابق ووضعنا دالة إدخال أخرى وأدخلنا نفس الجملة كما في المثال

C++	البرمجة بلغة C	البرمجة بلغة ++C
<pre>#include<iostream.h> int main() {1.char string1[20],string2[20]; 2.cin>>string1 ; 3 . cin>>string2 ; 4.cout<<string2 ; }</pre>	<pre>#include<stdio.h> int main() {1.char string1[20],string2[20]; 2.scanf("%s",&string1); 3 .scanf("%s",&string2); 4.printf("%s",string2); }</pre>	

في هذا البرنامج المفروض تنفذ خطوة رقم (٢) يدخل المستخدم (string1) ثم تنفذ خطوة رقم (٣) ويدخل المستخدم

(string2) هذه الكلام صحيح إذا لم يتجاوز المدخلات أكثر من كلمة هكذا



لو تلاحظ شاشة التنفيذ نفذ خطوة رقم(٢) وطلب من المستخدم إدخال وادخل (alxs) ونفذ خطوة رقم(٣) وطلب من المستخدم إدخال وادخل (hussien) وبعدها نفذ خطوة رابعة وطبع الاسم الثاني

** أما إذا ادخل المستخدم في خطوة رقم(٢) جملة مكونة من أكثر من كلمة وهي



فالذي يحدث كما قلنا يأخذ حتى الفراغ الأول ويعتبره كمدخل لدالة الإدخال الأولى

وهو (Hussien) ويترك بقية الجملة . وعندما ينفذ خطوة رقم(٣) لا يطلب من المستخدم

الإدخال إنما يعتبر ما تبقى من الجملة الأولى وهو (Ahmmed Taleb) كمدخل له ويأخذ أيضا حتى الفراغ الثاني أي

سوف يأخذ (Ahmmed) فقط وبهمل البقية وسوف يطبع في خطوة رقم(٤) محتويات (string2)

إذن الحل مع هذه المشاكل توجد دوال إدخال تأخذ الجملة المدخلة كاملة وهي:
(cin.get) تستخدم هذه الدالة في لغة (C++) لإدخال جملة كاملة وشكلها يكون

دالة الإدخال بلغة C++

```
Cin.get(string,number of input)
```

١. (string) هي المصفوفة المراد إدخالها ك (string).

٢. (number of input) هي عدد الحروف المحتملة التي سندخلها من شاشة التنفيذ ودائما اجعلها أكثر من احتياجك وان لا تتجاوز حجم المصفوفة. وأننا إذا تجاوزنا الإدخال هذا الرقم سوف سيهمل البقية .

(cin.getline) نفس طريقة الدالة (cin.get) لكنها تقرأ سطر واحد فقط

مثال: لو كان لدينا (string1) وحجمها (٢٥) ونريد إدخالها بهذه الدالة فيكون الكود هكذا

دالة الإدخال بلغة C++

```
Char string1[25]  
Cin.get(string1,25)
```

(gets) تستخدم هذه الدالة في لغة (C) لإدخال جملة كاملة فقط نكتبها ونكتب اسم (string) التي نريد أن ندخلها بها

دالة الإدخال بلغة C++

```
Char string1[25]  
gets(string1)
```

دوال الإخراج في لغة (C++) نستخدم نفس دالة الإخراج الاعتيادية نكتب اسم (string) وستطبع سلسلة كاملة

دالة الإدخال بلغة C++

```
cout<<string1;
```

دوال الإخراج في لغة (C) نستخدم نفس دالة الإخراج الاعتيادية ونستخدم في دالة الطباعة الرمز (%s) واسم (string) وستطبع سلسلة كاملة في شاشة التنفيذ

دالة الإدخال بلغة C++

```
Printf("%s",string1);
```

*لا نستخدم "%c" لان "%c" تطبع حرف واحد فقط وليس جملة

لنصح المثل السابق الذي كانت به مشكلة بدوال الإدخال باستخدام هذه الدوال الجديدة (افحص المثل وشاهد الفرق)

C++	البرمجة بلغة	C	البرمجة بلغة
<pre>#include<iostream.h> int main() {1.char string1[20],string2[20]; 2.cin.get(string1,20) ; 3 . cin.get(string2,20) ; 4.cout<<string2 ; }</pre>		<pre>#include<stdio.h> int main() {1.char string1[20],string2[20]; 2.gets(string1); 3 . gets(string2); 4.printf("%s",string2); }</pre>	

هناك عدة دوال للتعامل مع (string) تقع ضمن مكتبة <string.h> وهي:

1. strlen(). تستخدم هذه الدالة لإيجاد طول السلسلة النصية (أي عدد الأحرف في السلسلة وكذلك عدد الفراغات).

وطريقة استخدامها هي

هيكلية الدالة

```
strlen(string)
```

(string) هي المصفوفة المراد إيجاد طولها

مثال: إيجاد طول المصفوفة التالية

البرمجة

```
char string1[5]="alxs go";
int len;
len=strlen(string1); // len=7
```

يكون طول المصفوفة (7) لان حتى الفراغ أيضا يعتبر كحرف في حساب الطول السلسلة

مثال: برنامج ندخل سلسلة ويطبعها بالمقلوب .؟

تحليل: لطباعة أي مصفوفة بالمقلوب (أي آخر حرف يطبع أول حرف) يكون بوضع المؤشر على آخر حرف ثم نتنازل إلى أول حرف بالتسلسل فتتم الطباعة ونستطيع معرفة آخر حرف بواسطة دالة معرفة طول السلسلة سيعطيك طولها ونطرح من طولها واحد نجد آخر رقم بالمصفوفة لان ترقيم المصفوفة يبدأ من الصفر

C++	البرمجة بلغة	c	البرمجة بلغة
<pre>#include<iostream.h> #include<string.h> int main() {1.char string1[40]; 2.int l,len; 3 . cin.get(string1,40) ; 4.len=strlen(string1)-1; 5.for(i=len ;i>=0;i--) 6.cout<<string1[i];}</pre>		<pre>#include<stdio.h> #include<string.h> int main() {1.char string1[40]; 2.int i,len; 3 . gets(string1); 4.len=strlen(string1)-1; 5.for(i=len ;i>=0;i--) 6.printf("%c",string1[i]);}</pre>	

توضيح الخطوات:

1. خطوة رقم (1) عرفنا مصفوفة نصية،

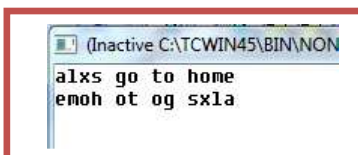
2. خطوة رقم (2) عرفنا عداد للمصفوفة ومتغير (len) لكي نخزن فيه طول السلسلة

3. خطوة رقم (3) قمنا بإدخال السلسلة بواسطة دوال الإدخال

4. خطوة رقم (4) خزن طول السلسلة بالمتغير (len) وطرنا من طول السلسلة واحد لان ترقيم المصفوفة يبدأ من

الصفر وطول المصفوفة يعطيك عدد الأحرف في المصفوفة لذلك يجب طرحه بواحد

5. عداد يبدأ بالعدد من آخر عنصر بالمصفوفة ويتناقص إلى أول عنصر ويطبع كل عنصر في خطوة (6)



```
(Inactive) C:\TCWIN45\BIN\NON
alxs go to home
emoh ot og sxla
```

* ولو تلاحظ في الطباعة بلغة (c) استخدمنا "%c" لاننا سنطبع حرف حرف وليس سلسلة

٢. `strcpy()` تستخدم هذه الدالة لنسخ جميع محتويات سلسلة إلى سلسلة أخرى وتكون طريقة النسخ انه يبدأ بإضافة عناصر السلسلة الثانية بمكان عناصر السلسلة الأولى التي لها نفس التسلسل بالموقع

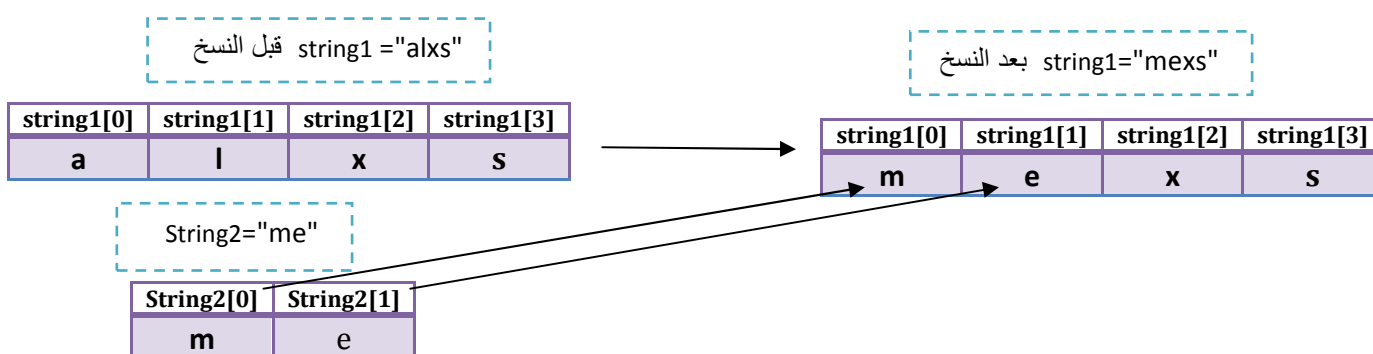
هيكلية الدالة

```
strcpy(string1, string2);
```

مثال: لنسخ جميع محتويات مصفوفة إلى مصفوفة أخرى

البرمجة

```
char string1[5]="alxs ";
char string2[5]="me ";
strcpy(string1, string2);
```



لو تلاحظ أن (String2) بقيت محافظة على محتوياتها نفسها بعد النسخ لأن النسخ يكون منها إلى (string1) وان (string1) تم أبدال محتويات كل موقع بما يكافئه في (String2) وبقيت المواقع التي لا يقابلها قيم من (String2) محتفظة بقيمتها

٣. `strncpy()` تستخدم هذه الدالة لنسخ عدد محدد من محتويات سلسلة إلى سلسلة أخرى وتكون طريقة النسخ انه يبدأ بإضافة عناصر السلسلة الثانية بمكان عناصر السلسلة الأولى التي لها نفس التسلسل بالموقع

هيكلية الدالة

```
strncpy(string1, string2, number of copy);
```

(number of copy) هو عدد الأحرف المراد نسخها من السلسلة الثانية إلى السلسلة الأولى

مثال: لنسخ ثلاث عناصر من محتويات مصفوفة إلى مصفوفة أخرى

البرمجة

```
char string1[5]="alxs ";
char string2[5]="suha muhamed ";
strncpy(string1, string2,3);
```

سوف ينسخ الحروف الثلاثة الأولى من السلسلة الثانية وهي (suh) إلى السلسلة الأولى ويكون محتويات (string1=suhs)

٤ (/). strcat تستخدم هذه الدالة للإلحاق محتويات سلسلة في نهاية سلسلة أخرى ومحافظة على محتوياتها .

هيكلية الدالة

```
strcat(string1, string2);
```

مثال: لإلحاق جميع محتويات مصفوفة إلى مصفوفة أخرى

البرمجة

```
char string1[7]="alxs ";
char string2[5]="me ";
strcat(string1, string2);
```

string1="alxs" قبل النسخ

string1[0]	string1[1]	string1[2]	string1[3]
a	l	x	s

string1 = "alxsme" بعد النسخ

string1[0]	string1[1]	string1[2]	string1[3]	string1[4]	string1[5]
a	l	x	s	m	e

String2 = "me"

String2[0]	String2[1]
m	e

لو تلاحظ أن (String2) بقيت محافظة على محتوياتها نفسها بعد الدمج لأن الإلحاق يكون منها إلى (string1) وان (string1) تم إضافة محتويات سلسلة (String2) إلى نهايتها . يجب أن يكون حجم المصفوفة المراد الإضافة إليها مساوي لعدد أحرفها وعدد الأحرف المضافة لو تلاحظ أن في المثال وضعنا حجم (string1) هو (٧) وهي مكونة من أربعة أحرف لأننا سنضيف إليها سلسلة مكونة من حرفان فيصبح طولها سبعة

٥ (/). strncpy تستخدم هذه الدالة بإلحاق عدد محدد من محتويات سلسلة في نهاية سلسلة أخرى وتكون طريقة

هيكلية الدالة

```
strncpy(string1, string2, number of copy);
```

(number of copy) هو عدد الأحرف المراد نسخها من السلسلة الثانية إلى السلسلة الأولى

مثال: لإلحاق ثلاث عناصر من محتويات مصفوفة إلى مصفوفة أخرى

البرمجة

```
char string1[5]="alxs ";
char string2[5]="suha muhamed ";
strncpy(string1, string2,3);
```

سوف ينسخ الحروف الثلاثة الأولى من السلسلة الثانية وهي (suh) إلى السلسلة الأولى ويكون محتويات (string1=alxssuh)

٦. `strcmp()` تستخدم هذه الدالة للمقارنة بين سلسلتين وتكون بالشكل التالي

هيكلية الدالة

```
strcmp(string1, string2);
```

هناك ثلاثة نتائج للمقارنة بين سلسلتين وهي.

١. فإذا كانت نتيجة المقارنة صفر فإن (String1) تساوي (String2)
٢. فإذا كانت نتيجة المقارنة اكبر من صفر فإن (String1) اكبر من (String2)
٣. فإذا كانت نتيجة المقارنة اصغر من صفر فإن (String1) اصغر من (String2)

مثال: للمقارنة بين سلسلتين

البرمجة

```
char string1[7]="aa ";
char string2[5]="ab ";
int cmper;
cmper=strcmp(string1, string2) ;//cmper<1
```

كانت نتيجة المتغير (cmper) سالبة لان (String1) اصغر من (String2)

٧. `strncmp()` تستخدم هذه الدالة للمقارنة بين عدد محدد من الأحرف من سلسلتين

هيكلية الدالة

```
strncmp (string1, string2,number of comper);
```

(number of comper) هو عدد الأحرف المراد مقارنتها من كلا السلسلتين

مثال: لمقارنة أول عنصر من محتويات مصفوفة الأولى والمصفوفة الثانية

البرمجة

```
char string1[5]="hussien ";
char string2[5]="hakmet ";
int cmper;
cmper=strncmp(string1, string2,1) ;//cmper=0
```

كانت نتيجة المتغير (cmper) صفر لان الحرف الأول في (String1) يساوي الحرف الأول في (String2)

مثال: برنامج ندخل سلسلة حرف ويحسب عدد مرات ظهور الحرف (a) في السلسلة

تحليل: بما انه قال سلسلة حروف نستخدم دوال الإدخال الخاصة ب(string) وبعدها نضع شرط بسيط للتحقق إذا كان احد الحروف هو (a) ونزيد عداد في كل مرة يجد فيها حرف (a)

البرمجة بلغة C++	البرمجة بلغة c
<pre>#include<iostream.h> #include<string.h> int main() {1.char string1[40]; 2.int i,len,number_appear=0; 3 . cin.get(string1,40) ; 4.len=strlen(string1)-1; 5.for(i=0;i<=len;i++) 6.If (string1[i]=='a') 7. number_appear=number_appear+1; 8.cout<<"number appear of (a)="<< number_appear;}</pre>	<pre>#include<stdio.h> #include<string.h> int main() {1.char string1[40]; 2.int i,len,number_appear=0; 3 . gets(string1); 4.len=strlen(string1)-1; 5.for(i=0;i<=len;i++) 6.If (string1[i]=='a') 7. number_appear=number_appear+1; 8.printf("number appear of (a)=%d", number_appear);}</pre>

توضيح الخطوات:

١. خطوة رقم (٥) كونا عداد يبدأ بأول حرف بالسلسلة وينتهي بأخر حرف لكي يتحقق من أحرف السلسلة حرف حتى ما وجد حرف (a) ستحقق الشرط في الخطوة رقم (٦) وينفذ خطوة رقم (٧) يزيد فيها قيمة العداد بواحد دلالة على انه وجد حرف جديد وكلما يجد الحرف يزيد العداد وفي النهاية تكون قيمة العداد بقدر عدد مرات ظهور الحرف (a) في السلسلة

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
hussien ahmed taleb is ago to his stage
number appear of (a)=4
```

مثال: برنامج ندخل سلسلة حرف ويحسب عدد مرات ظهور أحرف العلة وطباعتها أينما وجدت في السلسلة.؟

تحليل: فكرة هذا السؤال نفس فكرة السؤال السابق فقط الذي يغير شرط التحقق فبدلاً من أن يتحقق من حرف واحد سيتحقق من خمس حرف وأينما وجدها يطبعها

البرمجة بلغة C++	البرمجة بلغة c
<pre>#include<iostream.h> #include<string.h> int main() {1.char string1[40]; 2.int i,len,number_appear=0; 3 . cin.get(string1,40) ; 4.len=strlen(string1)-1; 5.for(i=0;i<=len;i++) 6.If ((string1[i]=='a') (string1[i]=='u') (string1[i]=='o') (string1[i]=='i') (string1[i]=='e')){ 7.cout<<string1[i]<<"\n"; 8. number_appear=number_appear+1;} 9.cout<<"number appear of vowel="<< number_appear;}</pre>	<pre>#include<stdio.h> #include<string.h> int main() {1.char string1[40]; 2.int i,len,number_appear=0; 3 . gets(string1); 4.len=strlen(string1)-1; 5.for(i=0;i<=len;i++) 6.If ((string1[i]=='a') (string1[i]=='u') (string1[i]=='o') (string1[i]=='i') (string1[i]=='e')){ 7.printf("%c",string1[i]); 8. number_appear=number_appear+1;} 9.printf("number appear of vowel =%d\n", number_appear);}</pre>

توضيح الخطوات: الشرط في خطوة رقم (٦) يتحقق متى ما جاء أي حرف من حروف العلة الخمسة يطبعه في خطوة رقم (٧) ويزيد قيمة عداد عدد مرات ظهور أحرف العلة في خطوة رقم (٨).

٢. استخدام الحروف في المصفوفات الثنائية:

لا يختلف استخدام الحروف في المصفوفات الثنائية عن استخدام الأرقام في المصفوفات الثنائية تستخدم في الإدخال حرف مكونة مصفوفة حرف وحتى في الطباعة تطبع حرف حرف
طريقة التعريف (على سبيل المثال مصفوفة حرفية ثنائية حجمها (3*3))

هيكلية الدالة

```
char first_arraychar [3][3];
```

وطريقة الإدخال أيضا ثابتة فقط العدادات تتغير حسب حجم المصفوفة وهي

البرمجة بلغة	c
c++ 1.char first_arraychar [3][3]; 2.for (i=0;i<3;i++) 3.for (j=0;j<3;j++) 4 scanf("%c",& first_array[i][j]) ;	1.char first_arraychar [3][3]; 2.for (i=0;i<3;i++) 3.for (j=0;j<3;j++) 4.cin>> first_arraychar [i][j] ;

** لاحظ خطوة رقم (٤) في الإدخال استخدمنا الرمز ("%c") في لغة (C) لأننا سندخل حرف حرف وليس سلسلة حروف

وتدخل بالشكل التالي

a	f	c
d	e	l
k	m	o

وطريقة الطباعة أيضا ثابتة فقط العدادات تتغير حسب حجم المصفوفة وهي

البرمجة بلغة	c
c++ 1.for (i=0;i<3;i++) 2.for (j=0;j<3;j++) 3.printf("%c", first_array[i][j]) ;	1.for (i=0;i<3;i++) 2.for (j=0;j<3;j++) 3.cout<< first_arraychar [i][j] ;

** أيضاً يجب رسم المصفوفة قبل البدء بالحل بأي سؤال

مثال: برنامج ندخل مصفوفة (4*4) ويحسب عدد أحرف العلة في المصفوفة.

تحليل: نفس طريقة الحل السابقة فقط نغير السلسلة إلى مصفوفة ثنائية الحجم

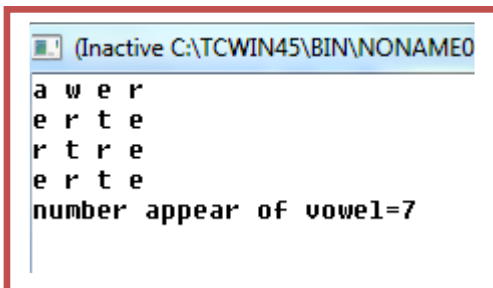
C++	البرمجة بلغة C
<pre>#include<iostream.h> main() {1.int i,j,k; 2.int number_appear ; 3. number_appear =0; 4.char vowel [7]='a','o','u','i','e','n','\0'; 5.char a[4][4]; 6.for(i=0;i<4;i++) 7.for(j=0;j<4;j++) 8.cin>>a[i][j]; 9.for(k=0;k<6;k++) 10.for(i=0;i<4;i++) 11.for(j=0;j<4;j++) 12.if(a[i][j]== vowel [k]) 13.number_appear = number_appear +1; 14.cout<<"number appear of vowel="<< number_appear;}</pre>	<pre>#include<stdio.h> main() {1.int i,j,k; 2.int number_appear ; 3. number_appear =0; 4.char vowel[7]='a','o','u','i','e','n','\0'; 5.char a[4][4]; 6.for(i=0;i<4;i++) 7.for(j=0;j<4;j++) 8.scnaf("%c",& a[i][j]) ; 9.for(k=0;k<6;k++) 10.for(i=0;i<4;i++) 11.for(j=0;j<4;j++) 12.if(a[i][j]== vowel [k]) 13.number_appear = number_appear +1; 14.printf("number appear of vowel=%d", number_appear);}</pre>

توضيح الخطوات:

١. خطوة رقم (٤) تم تعريف مصفوفة خزنا فيها جميع أحرف العلة

٢. خطوة رقم (٩) هو عداد للمصفوفة التي تحوي أحرف العلة حيث يعمل هذا العداد على مقارنة كل عنصر في مصفوفة (vowel) مع جميع العناصر المخزونة في المصفوفة المدخلة (a) فكلما يجد حرف علة يتحقق الشرط بخطوة رقم (١٢) لكي ينفذ خطوة رقم (١٣) ويزيد قيمة العداد بواحد دالا على إضافة حرف علة

٣. خطوة رقم (١٤) تتم طباعة عدد أحرف العلة كما في الشكل



```
(Inactive C:\TCWIN45\BIN\NONAMEO)
a w e r
e r t e
r t r e
e r t e
number appear of vowel=7
```

الفصل الخامس

الدوال (function)

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصول السابقة وفهماً كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على الدوال وفوائدها

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب : ثلاث ساعات

الدوال (Function):

تطرقنا في ما مضى على عدد كبير من الدوال الخاصة بلغة (c,c++) وبيننا طريقة استخدامها ومنها دالة (pow) التي تستخدم لايجاد قيمة الرقم المرفوع إلى قوى . وهذه الدوال كلها مصممة من قبل مصممي لغة (c,c++) لكي يسهل العمل على مبرمجين فبدلاً أن يكتب المبرمج عدد من الأسطر لإيجاد قيمة رقم المرفوعة إلى قوى على سبيل المثال إذا أردنا قيمة خمسة مرفوع إلى قوى ثلاثة (3^5) فبدون دالة (pow) نجد هكذا

إيجاد القوى بدون دالة (pow)

```
1.int i,pow1=1;
2.For(i=0 ;i<3 ;i++)
3.Pow1=pow1*5;
```

يحلها مباشرة باستخدام الدالة (pow)

إيجاد القوى باستخدام دالة (pow)

```
1.int Pow1=pow(5,3)
```

فوجدنا الفرق الطريقة الأولى استخدمنا ثلاث أسطر برمجية وثاني طريقة استخدمنا سطر برمجي واحد وكانت النتيجة نفسها لذلك فائدة الدوال هو تقليل الأسطر البرمجية ومنع تكرار اكواد لأكثر من مرة أي لو احتاجنا على سبيل المثال رفع رقم إلى قوى عشر مرات في برنامجنا في حال دون استخدام الدالة (pow) فسنضطر إلى تكرار الخطوات الثلاث عشر مرات كلما احتجنا إليها التي سوف تزيد من تعقيد البرنامج بينما بواسطة هذه الدالة سوف تقلل الاكواد وتمنع التكرار كلما نحتاجها نستدعيها. إذا هذا الكلام بالنسبة للدوال الجاهزة وأيضا نستطيع بناء دوال بنفسنا لكي نمنع تكرار اسطر برمجية معينة ونزيد من سهولة تتبع البرنامج كما فعل المبرمجون في بناء دوال

هيكلية دوال (function)

```
Type name(parameter1, parameter2,.....)
{
Statement;
Return(type);
}
```

• (Type) : هو نوع القيمة التي سوف ترجعها الدالة (function) بعد أن تنتهي من عملها . لان الدوال تكون على نوعين

١. احدها تعيد بعد استدعائها وتكون بشكل التالي عند استدعائها ويجب أن تحتوي على (return) لتعيد قيمة للبرنامج

كود

```
1. reslt =name(var1,var2,.....);
```

وبما أنها تعيد قيمة يجب خزن القيمة المعادة في متغير بنفس نوع المصفوفة ا ي لو كان نوع المصفوفة هو (integer) يجب أن يتم تعريف (reslt) على انه متغير من نوع (integer) لتخزن به النتيجة القيمة المرجعة

٢. الثانية لا تعيد أي قيمة للبرنامج تستدعي لتنفيذ ما في داخلها وينتهي كل شيء ولا تحتوي في داخلها على (return) لأنها لا تعيد أي قيمة وطريقة استدعائها في البرنامج فقط نكتب اسم الدالة هكذا

```
كود
name(var1,var2,.....);
```

- **(name):** هو اسم الدالة (function) ويمكن أن يكون أي اسم لكن للوضوح اجعل اسم الدالة دالا على عمل دالتك فلو بنيت دالة تجمع رقميين فسمي الدالة (addition) حتى تكون واضحة.
- **(parameter):** هي متغيرات يتم إرسال قيم إلى الدالة لنمثلها داخل الدالة لغرض تنفيذ العرض المطلوب وتقوم الدالة باستقبال المتغيرات المرسله حسب الترتيب. وكل متغير داخل الدالة يجب تعريفه مثال.

```
كود
Int disp(parameter1, parameter2,.....) { statement};
```

```
Reslt=disp(var1 , var2 ,.....)
```

كما تلاحظ في الاستدعاء كان بالتسلسل أسندت قيمة (parameter2= var2, parameter1= var1)

تسلسل تنفيذ الخطوات في البرنامج

```
# include<iostream.h> //or #include <stdio.h> for user of C language
Change_position()
{
6.
7.
8.
}
Main()
{
1.
2.
3.
4.
5. Change_position()
9.
10.
11.
}
```

عندما انتهى من تنفيذ ماموجود في الدالة واصبح يعود الى البرنامج الاصلي جاعلا خطوة التنفيذ التالية هي بعد (٨) وهي (٩) ويستمر البرنامج

من هنا يبدأ تنفيذ البرنامج خطوة بخطوة (منذ اول خطوة)

لا حظ في السطر الخامس احتاج دالة تقع خارج Main نقل تنفيذ البرنامج لها اصبح الخطوة (٦) عندها

مهم

تسلسل تنفيذ البرنامج يبدأ خطوة خطوة لكن لاحظ عن وصول إلى الخطوة رقم (٥) تم استدعاء دالة لذلك سينقل تنفيذ البرنامج لها لكي ينفذ الخطوات رقم (8-6) التي في داخلها ثم يعود إلى البرنامج حتى ينفذ خطوة رقم (٩ و ١٠ و ١١)

- **(Statement):** هي العملية المراد من الدالة تنفيذها عند استدعائها.

١. المتغيرات المعرفة داخل الدوال تنتهي حياتها بانتهاء تنفيذ آخر سطر في الدالة. أي لو كان متغير (i) في الدالة أصبحت قيمته (i=5) عند انتهاء تنفيذ الدالة فعند استدعاء الدالة مرة أخرى لا تكون قيمته خمسة لان حياته انتهت بانتهاء الاستدعاء السابق وعاد إلى قيمته الأولية قبل التغيير.....!
٢. وتكتب الدالة بعد التصريح عن المكتبات مباشرة
٣. الدالة التي تتم كتابتها تعامل داخل البرنامج حالها كحال أي دالة من دوال اللغة

مثال: دالة (function) تقوم بجمع رقمين وتعيد النتيجة إلى البرنامج..؟
 تحليل: من السؤال نفهم أن هذه الدالة تحتوي على (parameter) اثنان كل واحد خاص برقم معين وتعيد قيمة من نوع (integer) وهي النتيجة.

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> 1.int addition(int a,int b) { 2.int reslt; 3.reslt=a+b; 4.return(reslt); } 5.main() 6.{int reslt; 7.reslt= addition(3,7); 8.cout<< reslt<<"\n"; 9.reslt= addition(2,3); 10.cout<< reslt<<"\n"; }</pre>	<pre>#include<stdio.h> 1.int addition(int a,int b) { 2.int reslt; 3.reslt=a+b; 4.return(reslt); } 5.main() 6.{int reslt; 7.reslt= addition(3,7); 8.printf("%d\n", reslt); 9.reslt= addition(2,3); 10. printf("%d\n", reslt); }</pre>

توضيح الخطوات:

١. خطوة رقم (١) هي تعريف دالة لجمع متغيرين وتعيد قيمة من نوع (integer)

٢. خطوة رقم (٢) عرفنا متغير جديد وخطوة رقم (٣) جمعنا المتغيرين اللذان تم إرسالهما



٣. خطوة رقم (٤) تم إعادة هذه القيمة إلى المصفوفة

٤. خطوة رقم (٦) تم تعريف متغير باسم (reslt) لكي يجمل نتيجة الجمع وهناك متغير داخل الدالة (addition) بنفس الاسم لكن لا يؤثر عليه لأن المتغيرات داخل الدالة تكون غير معرفة للمتغيرات داخل البرنامج وكذلك المتغيرات داخل البرنامج تكون غير معرفة للمتغيرات داخل الدالة

٥. خطوة رقم (٧) تم استدعاء دالة الجمع وإعطاء قيمة (a=3,b=7) هكذا

كود

Int addition(int a,int b)

Reslt= addition (3, 7)

لكي ينفذ خطوة رقم (4-1) جامعاً الرقمين

الذان تم إرسالهما للدالة في خطوة رقم (٣)

٦. خطوة رقم (٨) تم طباعة ناتج الجمع وهو عشرة (في لغة C استخدمنا "%d" لا القيمة المعادة هي integer)

٧. خطوة رقم (٩) تم استدعاء دالة الجمع مرة أخرى وإعطاء قيمة (a=2,b=3) هكذا ويطبع ناتج الجمع في الخطوة رقم (١٠) وناتج الجمع هو خمسة

دوال (function) بدون نوع - وهي الدوال التي لا تحتوي على قيمة مرجعة إلى البرنامج. أي تنفذ ما في داخلها ولا تعيد أي قيمة إلى البرنامج قد تستقبل قيم لكنها لا تعيد أي قيمة وتعرف هكذا

كود

```
name(parameter1, parameter2,.....) { statement};
```

مثال: دالة (function) تقوم بطباعة رسالة معينة عند استدعائها..؟

تحليل: هذه الدالة تطبع رسالة أي لا ترجع أي قيمة ولا تستقبل أي قيمة

البرمجة بلغة	c	البرمجة بلغة	c++
	#include<stdio.h> 1. messageShow() { 2.printf("hi Mr.hussien"); } 3.main() { 4. messageShow() }		#include<iostream.h> 1. messageShow() { 2.cout<<"hi Mr.hussien"; } 3.main() { 4. messageShow() }

توضيح الخطوات:

١. خطوة رقم (١) هي دالة لا تستقبل أي متغير ولا تعيد أي متغير لذلك تكتب هكذا

٢. خطوة رقم (٢) هي الرسالة التي سيتم طباعتها عند استدعاء الدالة

٣. خطوة رقم (٤) هي استدعاء لدالة الطباعة لطباعة الرسالة المطلوبة.

مثال : دالة تطبع رقم معين يتم إرساله لها...

تحليل: بما أنها تطبع قيمة ترسل لها أي تستقبل قيمة واحدة ولا تعيد أي قيمة.

البرمجة بلغة	c	البرمجة بلغة	c++
	#include<stdio.h> 1. messageShow(int a) { 2.printf("the number send is=%d",a); } 3.main() { 4. messageShow(3) }		#include<iostream.h> 1. messageShow(int a) { 2.cout<<" the number send is="<<a; } 3.main() { 4. messageShow(3) }

توضيح خطوات: خطوة رقم واحد هي دالة تستقبل قيمة واحد لا تعيد أي قيمة وخطوة رقم (٢) تطبع هذه القيمة وخطوة رقم (٤) هو استدعاء لهذه الدالة من داخل البرنامج.

الإرسال بالقيمة والإرسال بالمرجع:



لإرسال متغيرات إلى دالة يجب أن ترسل بأحد الطريقتين

1. **الإرسال بالقيمة:** ترسل فقط قيمة المتغير إلى الدالة أي إذا تغير قيمة المتغير داخل الدالة لا تتغير قيمته الأصلية داخل البرنامج لأننا أرسلنا فقط قيمته إلى الدالة لمعالجتها. (أي لا تعداد أكثر من قيمة واحدة إلى البرنامج الرئيسي)
- مثال: بناء دالة تعمل نفس عمل الدالة (pow) التي تجد قيمة الرقم مرفوع إلى أس (x^n) ولنسميها (powA) معناه هذه الدالة عربية فقط للتمييز بينها وبين الأصلية والاثنتان يؤديان نفس العمل.

c++	البرمجة بلغة c	البرمجة بلغة c
<pre>#include<iostream.h> 1.int powA(int x,int n) { 2.int i, reslt =1; 3.for(i=0 ;i<n ;i++) 4. reslt = reslt *x; 5.return(reslt); } 6.main() 7.{int reslt,x,n; 8.x=3,n=4; 9.reslt= powA (x,n); 10.cout<<" powA="<<reslt<<"\n"; 11.x=5,n=2; 12.reslt= powA (x,n); 13.cout<<" powA="<<reslt<<"\n";}</pre>	<pre>#include<stdio.h> 1.int powA(int x,int n) { 2.int i, reslt =1; 3.for(i=0 ;i<n ;i++) 4. reslt = reslt *x; 5.return(reslt); } 6.main() 7.{int reslt,x,n; 8.x=3,n=4; 9.reslt= powA (x,n); 10. printf("powA=%d\n", reslt) ; 11.x=5,n=2; 12.reslt= powA (x,n); 13. printf("powA=%d\n", reslt) ;}</pre>	

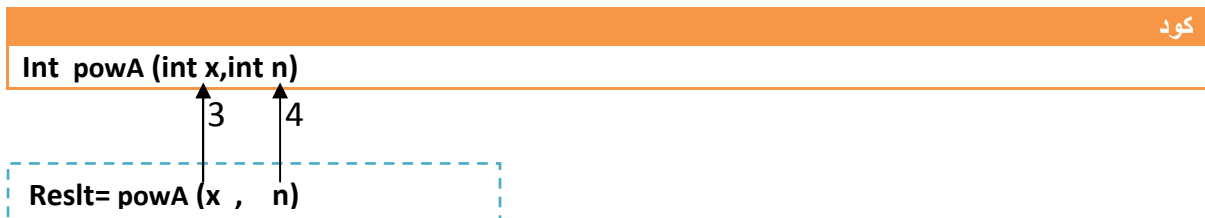
توضيح الخطوات:

1.خطو رقم(١ إلى ٥) هي دالة لإيجاد قيمة أي رقم مرفوع إلى أس

2.خطوة رقم (٨) هو إعطاء قيم للمتغير ولأس المراد إيجاده والمطلوب هنا (x^n) أي انه (3^4)



3.خطوة رقم (٩) تم إرسال قيم المتغيرين (x,n) دالة (powA) لإيجاد حل ل(3^4) هكذا



لو تلاحظ تم إرسال قيم المتغيرين وليس المتغيرين نفسيهما وهذا ما يسمى **بالإرسال بالقيمة** (أي أن المتغير (x,n) في الدالة ليس نفس المتغير (x,n) في البرنامج الرئيسي

4.خطوة رقم(١٠) تم إرسال قيم المتغيرين (x,n) دالة (powA) لإيجاد حل ل(5^2)

٢. **الإرسال بالمرجع:** ترسل موقع المتغير إلى الدالة أي أن الدالة تستقبل المتغير نفسه المرسل بنفس الاسم أو بأس ماخر. أي إذا تغير قيمة المتغير داخل الدالة تتغير قيمته الأصلية داخل البرنامج لأننا أرسلنا موقعه إلى الدالة والتغير يكون في محتوى الموقع . وشكلها هكذا

كود

```
Type name(&parameter1, &parameter2,.....) { statement};
```

نضع (&) قبل كل (parameter) نريد أن نعيد التغير في قيمته للبرنامج بمعنى أننا سنستقبل الموقع. **وفائدة الإرسال بالمرجع** هو أن الإرسال بلقيمة لا يعيد أكثر من قيمة متغير واحد إلى البرنامج بينما بالإرسال بالمرجع نستطيع إعادة أكثر من قيمة متغير إلى البرنامج

مثال: دالة نرسل لها متغيرين وتقوم بضرب كل واحد منهما بخمسة؟

تحليل: بما إننا نريد ضرب كل واحد منهما بخمسة أي أن الاثنان يتغيران ويعودان إلى الدالة الأصلية بقيم جديدة ونحن نعلم إننا نستطيع إعادة قيمة واحدة في حالة الإرسال بالقيمة لذلك سنستخدم الإرسال بالمرجع

C++	البرمجة بلغة c
<pre>#include<iostream.h> 1. mulByFive (int & x,int &n) { 2. x=x*5; 3. n=n*5; } 5.main() 6.{int x,n; 7. int y,z; 8.x=3,n=4; 9. mulByFive (x,n); 10.cout<<" x="<<x<<"\nn="<<n<<"\n"; 11.y=5,z=2; 12. mulByFive (y,z); 13. cout<<" y="<<y<<"\nz="<<z<<"\n";}</pre>	<pre>#include<stdio.h> 1. mulByFive(int & x,int &n) { 2. x=x*5; 3. n=n*5; } 4.main() 6.{int x,n; 7. int y,z; 8.x=3,n=4; 9. mulByFive (x,n); 10. printf("x=%d\nn=%d\n", x,n) ; 11.y=5,z=2; 12. mulByFive (x,n); 13. printf("y=%d\nz=%d\n", y,z) ;}</pre>

توضيح الخطوات:

١. خطوة رقم (٩) أرسلنا قيم المتغيرين (x=3,n=4) إلى الدالة وتم استقبالهما بالمرجع وطرب كل واحد منهما بخمسة لاحظ خطوة رقم (١٠) سيطبع قيمهم الجديدة مضروبة بخمسة

كود

```
Int mulByFive(int &x,int &n)
```

x n

```
Reslt= mulByFive (x , n)
```

كما نلاحظ من المخطط تم إرسال مواقع المتغيرات

٢. خطوة رقم (١٢) أرسلنا قيم المتغيرين ($y=5, z=5$) إلى الدالة وتم استقبالهما بالمرجع وطرب كل واحد منهما بخمسة لاحظ خطوة رقم (١٣) سيطلع قيمهم الجديدة مضروبة بخمسة كال احضنا لا يهتم مهما كان اسم المتغير لأنه سيستقبل موقعه بأسم آخر ويغير على القيم التي فيه

كود

```
Int mulByFive(int &x,int &n)
```

y z

```
Reslt= mulByFive (y , z)
```

كما نلاحظ من المخطط تم إرسال مواقع المتغيرات وتم التعبير عنهما بالدالة بأسماء جديدة لنفس المواقع

مهم

وباختصار إذا وضعنا (&) قبل أي (parameter). أي تغير في هذا (parameter) سوف يؤثر على قيمته في البرنامج الرئيسي. وإذا لم نضع هذه العلامة يبقى محافظا على قيمته في البرنامج الرئيسي

مهم

إسناد قيم لمتغيرات الدالة (function) : هي قيم يتم إسنادها للمتغيرات في الدالة فإذا لم نذكر هذه المتغيرات في الاستدعاء يتم الاعتماد على هذه القيم وإذا ذكرناها وأعطيناها قيمة يأخذ القيمة التي أعطيناها له.

مثال: دالة تحتوي على قيمة زائدة (دالة لضرب رقمين).؟

C++	البرمجة بلغة c	البرمجة بلغة
<pre>#include<iostream.h> 1.int mul (int a,int b=3) 2. {return(a*b);} 3.main() 4. {cout<<"mul="<< mul (3); 5. cout<<"\nmul="<< mul (3,5);}</pre>	<pre>#include<stdio.h> 1.int mul (int a,int b=3) 2. {return(a*b);} 3.main() 4. {printf("mul=%d\n", mul (3)); 5. printf("mul=%d", mul (3,5));}</pre>	

توضيح الخطوات:

١. خطوة رقم (١) هي دالة تستقبل قيمتين وتكون قيمة (b) هي قيمة زائدة مساوية ل (٣)

٢. خطوة رقم (٤) هو استدعاء للدالة وأعطينا فقط قيمة (a) ولم نذكر المتغير (b) لذلك سيعتمد المتغير (b) على القيمة الزائدة وهي (٣) ويضرب ($3*3=9$) ويطلع رقم (٩) في شاشة التنفيذ

٣. خطوة رقم (٥) هو استدعاء للدالة وأعطينا قيمة للمتغير (a,b) لذلك سيهمل القيمة الزائدة ويعتمد على قيمة ($b=5$) ويضرب ($3*5=15$) ويطلع (١٥) في شاشة التنفيذ

فائدة الدوال في تقليل عدد الاكواد البرمجية وترتيب البرنامج

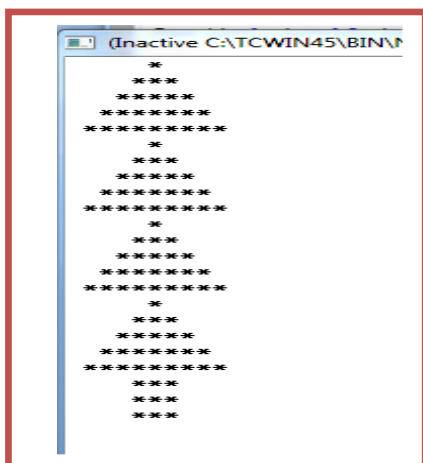
مثال :برنامج لرسم الشجرة التالية..؟

تحليل: عند تحليلنا لهذه الشجرة نرى أن الجزء المثلث يعاد أربع مرات معناه

نفس الكود وبعده كود آخر لقاعدة الشجرة. أي أننا لدينا كودان

احدهما يعاد أربع مرات وآخر مرة واحدة !..

الحل بدون دوال (function) !.....



C++	البرمجة بلغة	C	البرمجة بلغة
<pre>#include<iostream.h> main() {int i,j,k; for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) cout<<" "; for(j=i;j>0;j--) cout<<"*"; cout<<"\n"; } for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) cout<<" "; for(j=i;j>0;j--) cout<<"*"; cout<<"\n"; } for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) cout<<" "; for(j=i;j>0;j--) cout<<"*"; cout<<"\n"; } for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) cout<<" "; for(j=i;j>0;j--) cout<<"*"; cout<<"\n"; } for(i=1;i<=6;i+=2){ for(k=4;k>0;k-=1) cout<<" "; for(j=3;j>0;j--) cout<<"*"; cout<<"\n";}}</pre>		<pre>#include<stdio.h> main() {int i,j,k; for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) printf(" "); for(j=i;j>0;j--) printf("*"); printf("\n"); } for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) printf(" "); for(j=i;j>0;j--) printf("*"); printf("\n"); } for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) printf(" "); for(j=i;j>0;j--) printf("*"); printf("\n"); } for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) printf(" "); for(j=i;j>0;j--) printf("*"); printf("\n"); } for(i=1;i<=6;i+=2){ for(k=4;k>0;k-=1) printf(" "); for(j=3;j>0;j--) printf("*"); printf("\n");}}</pre>	

الحل باستخدام دوال (function) !.....

تحليل:ولو كتبنا الكود المعاد أربع مرات في دالة واستدعيناها أربع مرات وبعده نكتب كود قاعدة الشجرة لتبسط البرنامج كثيرا وأصبح واضح وسهل.

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> Draw_tree() { int i,j,k; for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) cout<<" " ; for(j=i;j>0;j--) cout<<"*"; cout<<"\n" ;}} main() { int i,j,k; Draw_tree(); Draw_tree(); Draw_tree(); Draw_tree(); for(i=1;i<=6;i+=2){ for(k=4;k>0;k-=1) cout<<" " ; for(j=3;j>0;j--) cout<<"*"; cout<<"\n" ;}}</pre>	<pre>#include<stdio.h> Draw_tree() { int i,j,k; for(i=1;i<=10;i+=2){ for(k=i;k<10;k+=2) printf(" ") ; for(j=i;j>0;j--) printf("*"); printf("\n");}} main() { int i,j,k; Draw_tree(); Draw_tree(); Draw_tree(); Draw_tree(); for(i=1;i<=6;i+=2){ for(k=4;k>0;k-=1) printf(" ") ; for(j=3;j>0;j--) printf("*"); printf("\n");}}</pre>

لو تلاحظ كم تبسط الكود وكم أصبح البرنامج واضح عندما وضعنا الجزء المكرر أربع مرات في دالة واستدعيناها أربع مرات فكون شجرة وبقية قاعدة الشجرة كتبنا كودها وحده.

مهم

المتغيرات التي تعرف تحت تعريف المكتبات تسمى **متغيرات عامة** تكون معرفة بالنسبة إلى جميع أجزاء البرنامج أي على سبيل المثال لو عرفنا متغير اسمه (item) يكون هذا المتغير معرف بالنسبة إلى جميع الدوال أو إلى البرنامج الرئيسي على خلاف **المتغيرات الخاصة** التي تعرف داخل الدوال تكون معرفة نسبة إلى الدالة

توضيح

```
#Include< >
Global variable ← متغيرات عامة معرفة إلى جميع أجزاء البرنامج

Main()
{
Local variable ← متغيرات خاصة تكون معرفة فقط داخل الدالة (main )
}
```

وهذه المتغيرات لا تنتهي حياتها إلا بانتهاء البرنامج وتبقى محفوظة على قيمها الجديدة ولا تعود لقيمتها البدائية.

نستطيع تعريف واستخدام أكثر من دالة في برنامج واحد..



الدوال الزائدة: هي مجموعة دوال لها نفس الاسم وتختلف في القيمة المعادة أو تختلف في نوع (parameter) المستقبل للدالة. فعند استدعاء احد هذه الدوال وبما أنها جميعا بنفس الاسم لذلك سوف يستدعي المترجم الدالة التي تستقبل اقرب نوع للمتغير الذي أرسلته لها أو نفس النوع.

مثال: برنامج يحوي على دوال الآتية احدها تجمع الرقمين والأخرى تقسمهم والأخرى تضربهم ولهم نفس الاسم؟

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> 1.int operation (int a, int b) 2. {return(a*b);} 3. float operation (float a, int b) 4. {return(a+b);} 5.float operation (float a, float b) 6. {return(a/b);} 7.main() 8. {int a=3,b=2; 9.float c=2.5,d=4.5; 10.cout<<"reslt="<< operation (a,b); 11. cout<<"\n reslt="<< operation (c,d); 12. cout<<"\n reslt="<< operation (c,a); }</pre>	<pre>#include<stdio.h> 1.int operation (int a, int b) 2. {return(a*b);} 3.float operation (float a, float b) 4. {return(a/b);} 5. float operation (float a, int b) 6. {return(a+b);} 7.main() 8. {int a=3,b=2; 9.float c=2.5,d=4.5; 10. printf("reslt=%d\n", operation (a,b)); 11. printf("reslt=%f\n", operation (c,d)); 12. printf("reslt=%f\n", operation (c,a)); }</pre>

توضيح الخطوات:

١. خطوة رقم (١٠) تم إرسال متغيرين (a,b) من نوع integer لذلك ستتقبله الدالة في الخطوة رقم (١) وتضرب الرقميين لان هذه الدالة تستقبل المتغيرين من نوع integer



٢. خطوة رقم (١١) تم إرسال متغيرين (c,d) من نوع float لذلك ستتقبله الدالة في الخطوة رقم (5) وتقسم الرقميين لان هذه الدالة تستقبل المتغيرين من نوع float



٣. خطوة رقم (١٢) تم إرسال متغيرين (a) من نوع integer , (c) من نوع float لذلك ستتقبله الدالة في الخطوة رقم (3) وتجمع الرقميين لان هذه الدالة تستقبل متغير من نوع integer ومتغير من نوع float



استدعاء الدالة لنفسها (Recursively):

وتسمى أيضا (Recursive Function) أي استدعاء الدالة لنفسها لمرة أو أكثر من مرة. فتكون شكلها معروف أنها من داخل دالة (function) نقوم باستدعائها لنفسها (أي يذكر اسمها في داخلها) ويكون شكلها هكذا

هيكلية دوال (Recursive function)

```
Type name(parameter1, parameter2,.....)
{
Statement;
Return( name(parameter1, parameter2,.....) );
}
```

مثال: بناء الدالة التالية (X^n) أي دالة (pow) باستخدام أسلوب (Recursive Function)

تحليل: عندما يذكر (Recursive Function) يجب أن ترك أي فكرة في ذهننا لحل السؤال بدون استدعاء الدالة لنفسها. وكما نرى أن رفع الرقم لأس معي معناه ضرب الرقم بنفسه بمقدار قيمة الأس مثال على ذلك

كود

```
4^3=4*4*4
```

أذن سنكون دالة تعيد استدعاء لنفسها بمقدار قيمة الأس المرفوع له الرقم وبكل استدعاء تضرب الرقم في نفسه

C++	البرمجة بلغة c	البرمجة بلغة
<pre>#include<iostream.h> 1.int power(int x,int n) { 2.if (n>0) 3.return(x*power(x,n-1)); 4.else 5.return 1; } 6.main() { 7.cout<<power(4,3);}</pre>	<pre>#include<stdio.h> 1.int power(int x,int n) { 2.if (n>0) 3.return(x*power(x,n-1)); 4.else 5.return 1; } 6.main() { 7.printf("%d",power(4,3));}</pre>	

توضيح الخطوات:

١. خطوة رقم (٧) أردنا إيجاد قيمة (4^3) لذلك سيستدعي الدالة في الخطوة رقم (١) جاعلا قيمة ($x=4, n=3$) ثم
ينفذ خطوة رقم (٢) لأن ($n>0$) وينفذ بعده خطوة رقم (٣) لكي يعيد الرقم مضروب باستدعاء الدالة لنفسها مرسلتها
الرقم والأس منقص منه واحد ويستمر بالاستدعاء الذاتي إلى أن تصبح قيمة الأس ($n=0$) يعيد عندها واحد ويخرج
من الاستدعاء الذاتي .

توضيح ماذا سيحصل في خطوات رقم (١ إلى ٥) في حال كان (x=4,n=3)

توضيح ماذا سيحصل في خطوات رقم (١ إلى ٥) في حال كان (x=4,n=3) (فقط الخطوات التي سوف تنفذ بكل استدعاء)

الاستدعاء الأول

1. power(x=4,n=3)

2. n=3 is large than zero

3. return(4*power(4,3-1))

الاستدعاء الثاني

1. power(x=4,n=2)

2. n=2 is large than zero

3. return(4*4*power(4,2-1))

الاستدعاء الثالث

1. power(x=4,n=1)

2. n=1 is large than zero

3. return(4*4*4*power(4,1-1))

الاستدعاء الرابع

1. power(x=4,n=0)

4. n=0 is equal to zero

5. return(4*4*4*1)

استدعت الدالة لنفسها أربع مرات ؟ والنتيجة هي (4*4*4*1=64) أذن النتيجة صحيحة..، كتبنا فقط الخطوات التي سوف تنفذ بكل استدعاء للدالة والتي لم تنفذ لم نكتبها

مثال: أيجاد مفكوك الرقم باستخدام أسلوب (Recursive Function) ..؟

تحليل: عندما يذكر (Recursive Function) يجب أن ترك أي فكرة في ذهننا لحل السؤال بدون استدعاء الدالة لنفسها. وكما نرى أن المفكوك هو ناتج من حاصل ضرب الرقم بالأرقام التي أقل منه وصولاً إلى الواحد

كود

5!=5*4*3*2*1

أذن سنكون دالة تعيد استدعاء لنفسها بمقدار قيمة الرقم مثلاً مفكوك خمسة ستستدعي الدالة نفسها خمس مرات وفي كل مرة تستدعي الدالة لنفسها نظراً من الرقم المرسل واحد ونضربه ببقية الأرقام هكذا

كود

N!=N*(n-1)!

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> 1.int fact (int x) { 2.if (x>1) 3.return(x* fact (x-1)); 4.else 5.return 1; } 6.main() { 7.cout<<fact(4);}</pre>	<pre>#include<stdio.h> 1.int fact (int x) { 2.if (x>1) 3.return(x* fact (x-1)); 4.else 5.return 1; } 6.main() { 7.printf("%d",fact(4));}</pre>

توضيح الخطوات:

- خطوة رقم (٧) استدعينا دالة المفكوك وأردنا إيجاد مفكوك الرقم أربعة فعند الدخول للدالة نرى وجود خطوة رقم (٣) هذه الخطوة تضرب الرقم بمفكوك الأرقام التي اقل منه هكذا $(n*(n-1)!)$ وتستمر بالضرب إلى أن يصل إلى الرقم صفر

توضيح ماذا سيحصل في خطوات رقم (١ إلى ٥) في حال كان $(x=4, n=3)$

توضيح ماذا سيحصل في خطوات رقم (١ إلى ٥) في حال كان $(x=4)$ (فقط الخطوات التي سوف تنفذ بكل استدعاء)
<p>الاستدعاء الأول</p> <ol style="list-style-type: none"> 1. fact(x=4) 2. x=4 is large than one 3. return(4* fact (4-1)) <p>الاستدعاء الثاني</p> <ol style="list-style-type: none"> 1. fact (x=3) 2. x=3 is large than one 3. return(4*3*power(3-1)) <p>الاستدعاء الثالث</p> <ol style="list-style-type: none"> 1. fact (x=2) 2. x=2 is large than one 3. return(4*3*2*power(2-1)) <p>الاستدعاء الرابع</p> <ol style="list-style-type: none"> 1. fact (x=1) 4. x=1 is equal to one 5. return(4*3*2*1)

استدعت الدالة لنفسها أربع مرات ؟ والنتيجة هي $(4*3*2*1=24)$ أذن النتيجة صحيحة..، كتبنا فقط الخطوات التي سوف تنفذ بكل استدعاء للدالة والتي لم تنفذ لم نكتبها



أذا كانت الدالة مكونة من سطر برمجي واحد تسمى دوال سطرية (inline function) وتكتب هكذا

هيكلية دوال (inline function)
<pre>inlineType name(parameter1, parameter2,.....) {Statement;}</pre>



المصفوفات والدوال:

يمكن إرسال مصفوفات أحادية الأبعاد إلى الدوال بذكر اسمها فقط بدون أبعاد .

مثال: لو كان لدينا مصفوفة اسمها (a) ونريد إرسالها إلى دالة اسمها (name)

إرسال مصفوفة أحادية إلى الدوال (function)

```
name( a );
```

وطريقة استقبال المصفوفة الأحادية في الدوال نضع فقط أقواس المصفوفة بدون إبعاد

إرسال مصفوفة أحادية إلى الدوال (function)

```
Type name(type a[] );
```

• (type) : هو نوع المصفوفة المرسله

يمكن إرسال مصفوفات ثنائية الأبعاد إلى الدوال فقط بذكر اسمها بدون أبعاد .

مثال: لو كان لدينا مصفوفة اسمها (a) وإبعادها (2*3) ونريد إرسالها إلى دالة اسمها (name)

إرسال مصفوفة ثنائية إلى الدوال (function)

```
name( a );
```

وطريقة استقبال المصفوفة ثنائية الأبعاد في الدوال نذكر فقط البعد الثاني هكذا

إرسال مصفوفة ثنائية إلى الدوال (function)

```
Type name(type a[][3] );
```

- المصفوفات عند إرسالها بهذه الطرق إلى الدوال ترسل بالقيمة وليس بالمرجع. إذا إي تغيير على عناصر المصفوفة في الدالة سوف لا يؤثر على القيم في البرنامج الرئيسي. أي ترسل نسخة من المصفوفة وليس المصفوفة نفسها.

مثال: خمس مصفوفات أحادية الأبعاد حجمها (7) جد جمع كل مصفوفة واكبر عدد بكل مصفوفة..؟

تحليل: كما نرى انه يريد ناتج جمع كل مصفوفة واكبر عدد فيكون البرنامج كبير جدا إذا لم نستعمل الدوال (function) لأنه لدل مصفوفة يجب كتابة كود يجد اكبر رقم ويجمع عناصر المصفوفة لكن مع الدوال (function) تكون دالة تجد مجموع عناصر المصفوفة ودالة تجد اكبر رقم ونمرر كل مصفوفة لهاتان الدالتان

البرمجة بلغة	C++	البرمجة بلغة	C
	<pre>#include<iostream.h> 1.int maxN(int array1[]) 2.{ int i, max; 3.max=array1[0]; 4.for (i=0;i<7;i++) 5.if (array1[i] > max) 6.max=array1[i]; 7.return(max);} 8.int sumN(int array1[]) 9.{ int i, sum=0; 10.for (i=0;i<7;i++) 11.sum=sum+ array1[i]; 12.return(sum);} 13.main() 14.{ int i, a[7],b[7],c[7],d[7],e[7]; 15.cout<< "enter element (1) array="; 16.for (i=0;i<7;i++) 17.cin>>a[i]; 18.cout<<"max="<<maxN(a)<<"\tsum="<<sumN(a)<<"\n"; 19.cout<< "enter element (2) array="; 20.for (i=0;i<7;i++) 21.cin>>b[i]; 22.cout<<"max="<<maxN(b)<<"\tsum="<<sumN(b)<<"\n"; 23.cout<< "enter element (3) array="; 24.for (i=0;i<7;i++) 25.cin>>c[i]; 26.cout<<"max="<<maxN(c)<<"\tsum="<<sumN(c)<<"\n"; 27.cout<< "enter element (4) array="; 28.for (i=0;i<7;i++) 29.cin>>d[i]; 30.cout<<"max="<<maxN(d)<<"\tsum="<<sumN(d)<<"\n"; 31.cout<< "enter element (5) array="; 32.for (i=0;i<7;i++) 33.cin>>e[i]; 34.cout<<"max="<<maxN(e)<<"\tsum="<<sumN(e)<<"\n";}</pre>		<pre>#include<stdio.h> 1.int maxN(int array1[]) 2.{ int i, max; 3.max=array1[0]; 4.for (i=0;i<7;i++) 5.if (array1[i] > max) 6.max=array1[i]; 7.return(max);} 8.int sumN(int array1[]) 9.{ int i, sum=0; 10.for (i=0;i<7;i++) 11.sum=sum+ array1[i]; 12.return(sum);} 13.main() 14.{ int i, a[7],b[7],c[7],d[7],e[7]; 15.printf("enter element (1) array="); 16.for (i=0;i<7;i++) 17 scanf("%d",&a[i]); 18.printf("max=%d\tsum=%d\n",maxN(a),sumN(a)); 19. printf("enter element (2) array="); 20.for (i=0;i<7;i++) 21. scanf("%d",&b[i]); 22. printf("max=%d\tsum=%d\n",maxN(b),sumN(b)); 23. printf("enter element (3) array="); 24.for (i=0;i<7;i++) 25. scanf("%d",&c[i]); 26. printf("max=%d\tsum=%d\n",maxN(c),sumN(c)); 27. printf("enter element (4) array="); 28.for (i=0;i<7;i++) 29. scanf("%d",&d[i]); 30. printf("max=%d\tsum=%d\n",maxN(d),sumN(d)); 31. printf("enter element (5) array="); 32.for (i=0;i<7;i++) 33. scanf("%d",&e[i]); 34. printf("max=%d\tsum=%d\n",maxN(e),sumN(e));}</pre>

توضيح الخطوات:

- خطوة رقم (١) هي دالة تستقبل المصفوفة وتقوم بإيجاد اكبر رقم لو تلاحظ الخطوات (٢ إلى ٧) هي خطوات إيجاد اكبر رقم نفسها التي شرحناها في المصفوفة الأحادية وكيفية إيجاد اكبر رقم فقط وضعناها في دالة
 - خطوة رقم (٨) هي دالة لجمع عناصر المصفوفة
 - خطوة رقم (١٦ و ١٧) هي إدخال المصفوفة (a) وخطوة رقم (١٨) هي استدعاء دالة اكبر رقم ودالة جمع عناصر المصفوفة وطباعة الرقم وبقية الخطوات تتكرر نفس العملية بالنسبة لبقية المصفوفات
- ** الآن أصبح لدينا دالة متى استدعيناها تجد اكبر رقم.

```

C:\Windows\BIN\WONAME1.EXE
nter element (1) array=2 3 4 3 2 5 4
ax=5 sum=20
nter element (2) array=4 6 8 7 5 4 6
ax=77 sum=149
nter element (3) array=4 5 3 8 6 5 4 3
ax=54 sum=83
nter element (4) array=34 23 56 44 34 54 34
ax=56 sum=279
nter element (5) array=65 45 34 77 45 45 34
ax=77 sum=345

```

مثال: برنامج لحساب الحرف الأكثر تكرار وعدد مرات تكراره. في مصفوفة أحادية؟

تحليل: لإيجاد الحرف الأكثر تكرار ضمن إي سلسلة يجب خزن جميع الأحرف في مصفوفة وحساب عدد مرات ظهور كل حرف في السلسلة وطباعة الحرف الذي يظهر أكثر من غيره.

البرمجة بلغة	C	البرمجة بلغة	C++
	<pre>#include<stdio.h> #include<string.h> 1.int i,m, max,fou; 2.char a[45], charSaved[255]; 3.repeat(char string[],int len) 4.{ int equavelentnumber[255]={0}; 5. for(i=0;i<255;i++) 6.charSaved[i]=char(i); 7.for(m=0;m<255;m++){ 8.for(i=0;i<len-1;i++) 9.if ((charSaved [m]== string [i]) && (string [i] !=' ')) 10.equavelentnumber [m]+= 1 ;} 11.max= equavelentnumber [0]; 12.for(i=0;i<255;i++) 13.if (equavelentnumber [i]>max){ 14.max= equavelentnumber [i]; 15.fou =i;} 16.printf("charcter more repeat=%c\n", charSaved [fou]); 17.printf("it repeat=%d\n", max); } 18.main() 19.{int count1; 20.for (count1=1; count1<6; count1++){ 21.printf("enter the Sting(%d): ", count1); 22.gets(a); 23.repeat(a,strlen(a));}}</pre>		<pre>#include<iostream.h> #include<string.h> 1.int i,m, max,fou; 2.char a[45], charSaved[255]; 3.repeat(char string[],int len) 4.{ int equavelentnumber[255]={0}; 5. for(i=0;i<255;i++) 6.charSaved[i]=char(i); 7.for(m=0;m<255;m++){ 8.for(i=0;i<len-1;i++) 9.if ((charSaved [m]== string [i]) && (string [i] !=' ')) 10.equavelentnumber [m]+= 1 ;} 11.max= equavelentnumber [0]; 12.for(i=0;i<255;i++) 13.if (equavelentnumber [i]>max){ 14.max= equavelentnumber [i]; 15.fou =i;} 16.cout<<"charcter more repeat "<< charSaved [fou]<<" \n"; 17.cout<<"it repeat="<<max <<"\n"; } 18.main() 19.{int count1; 20.for (count1=1; count1<6; count1++){ 21.cout<<"enter the Sting("<<count1<<"): "; 22.cin.getline(a); 23.repeat(a,strlen(a));}}</pre>

توضيح الخطوات:

- خطوة رقم (١) عرفنا متغيرات عامة معرفة لجميع أجزاء البرنامج
 - خطوة رقم (٢) عرفنا مصفوفة (charSaved) لنخزن فيها جميع الأحرف حتى نحسب عدد مرات ظهور كل حرف
 - خطوة رقم (٣) هي دالة تستقبل السلسلة وطولها وتطبع الحرف الأكثر تكرار وعدد مرات تكراره
 - خطوة رقم (٤) عرفنا مصفوفة (equavelentnumber) لنخزن فيها عدد مرات ظهور كل حرف
 - خطوة رقم (٥ و ٦) نخزن جميع الأحرف والرموز في مصفوفة اسمها (charSaved)
 - خطوة رقم (٧ و ٨ و ٩ و ١٠) نحسب عدد مرات ظهور كل حرف في السلسلة التي ادخلها المستخدم ونخزن في مصفوفة (equavelentnumber) وتكون هذه المصفوفة مكافئ لمواقع كل رمز أو حرف في مصفوفة (charSaved)
 - خطوة رقم (١١ إلى ١٧) نحسب الحرف الذي ظهر أكثر مرة ونطبع و نطبع عدد مرات ظهوره
 - خطوة رقم (٢٠ إلى ٢٣) هو ادخل المصفوفات وإرسالها إلى الدالة
- ** لا يجوز تعريف متغير (متغير عام) واستخدامه كعداد في البرنامج الرئيسي وهو مستخدم داخل احد الدوال كعداد أو تتغير قيمته سوف لا ينفذ البرنامج لأنه سوف يجعل عبارة التكرار في غموض

```
ive C:\TCWIN45\BIN\NONAME01.EXE
the Sting(1): hussien ahmed taleb
er more repeat=e
eat=3
the Sting(2): he go to home
er more repeat=0
eat=3
```

مثال: برنامج لترتيب ثلاث مصفوفات ثنائية الإبعاد (5*5) تصاعديا؟

تحليل: لترتيب ثلاث مصفوفات ثنائية نستخدم نفس طريقة ترتيب المصفوفات الثنائية ونضعها في دالة (function) ونستدعيها ثلاث مرات.

البرمجة بلغة	c	البرمجة بلغة
<pre> c++ #include<iostream.h> 1.int const row=5; 2.int const col=5; 4.int i,j,k,x,l ; 3.Sort2D(int array[][col]) { 4.for(k=0;k<row;k++){ 5. for(l=0;l<col;l++){ 6. for(i=0;i<row;i++){ 7. for (j=0;j<col;j++){ 8. if (array[i][j] >array[k][l]){ 9.x=array[k][l]; 10.array[k][l]=array[i][j]; 11.array[i][j]=x; 12.}} } } 13. cout<<"Here is the Array after sorted\n" ; 14. for (i=0;i<row;i++){ 15. for (j=0;j<row;j++) 16. cout<<array[i][j]<<"\t"; 17. cout<<"\n" ;}} 18. main() 19.{ int array1[row][col]; 20. int array2[row][col]; 21. int array3[row][col]; 22.cout<<"Here is the Array(1) befor sorted\n" ; 23. for (i=0;i<row;i++) 24. for (j=0;j<col;j++) 25.cin>>array1[i][j] ; 26.Sort2D(array1); 27.cout<<"Here is the Array (2) befor sorted\n" ; 28. for (i=0;i<row;i++) 29. for (j=0;j<col;j++) 30.cin>>array1[i][j] ; 31.Sort2D(array2); 32.cout<<"Here is the Array (3) befor sorted\n" ; 33. for (i=0;i<row;i++) 34. for (j=0;j<col;j++) 35.cin>>array3[i][j] ; 36.Sort2D(array3);} </pre>	<pre> c #include<stdio.h> 1.int const row=5; 2.int const col=5; 4.int i,j,k,x,l ; 3.Sort2D(int array[][col]) { 4.for(k=0;k<row;k++){ 5. for(l=0;l<col;l++){ 6. for(i=0;i<row;i++){ 7. for (j=0;j<col;j++){ 8. if (array[i][j] > array[k][l]){ 9.x=array[k][l]; 10.array[k][l]=array[i][j]; 11.array[i][j]=x; 12.}} } } 13. printf("Here is the Array after sorted\n") ; 14. for (i=0;i<row;i++){ 15. for (j=0;j<row;j++) 16. printf("%d\t",array[i][j]); 17. printf("\n");}} 18. main() 19.{ int array1[row][col]; 20. int array2[row][col]; 21. int array3[row][col]; 22. printf("Here is the Array(1) befor sorted\n") ; 23. for (i=0;i<row;i++) 24. for (j=0;j<col;j++) 25.scanf("%d",&array1[i][j]); 26.Sort2D(array1); 27. printf("Here is the Array (2) befor sorted\n") ; 28. for (i=0;i<row;i++) 29. for (j=0;j<col;j++) 30. scanf("%d",&array2[i][j]); 31.Sort2D(array2); 32. printf("Here is the Array (3) befor sorted\n") ; 33. for (i=0;i<row;i++) 34. for (j=0;j<col;j++) 35. scanf("%d",&array3[i][j]); 36.Sort2D(array3);} </pre>	

توضيح الخطوات: خطوة رقم (٣) هي دالة لترتيب عناصر مصفوفة تصاعديا ونلاحظ خطوة رقم (٢٢ إلى ٢٥) هي إدخال للمصفوفة الأولى وخطوة رقم (٢٦) هي إرسالها لترتيبها ولخطوات البقية هي إدخال بقية المصفوفات وترتيبها.

مثال: برنامج لإيجاد أحرف العلة وعددها في ثلاث مصفوفات ثنائية الإبعاد (4*4) .؟

تحليل: إيجاد أحرف العلة في مصفوفات ثنائية لأكثر من واحدة نحتاج إلى دوال (function) .

c++	البرمجة بلغة c
<pre>#include<iostream.h> 1.char vowelchar [7]={'a','o','u','i','e','n','\o'}; 2..int i,j,k; 3.int indexofprintarray=1; 4.vowel(char a[][4]) 5.{ cout<<"\nvowel Char in array("<<indexofprintarray<<")\n"; 6.indexofprintarray=indexofprintarray+1; 7.int number_appear ; 8. number_appear =0; 9.for(k=0;k<6;k++) 10.for(i=0;i<4;i++) 11.for(j=0;j<4;j++) 12.if(a[i][j]== vowelchar [k]) 13.{cout<< "\n"<<a[i][j]; 14.number_appear = number_appear +1;} 15.cout<<"\nnumber appear of vowel="<< number_appear;}} 16.main() 17.{char a[4][4],b[4][4], c[4][4]; 18.cout<<"\nenter (1) array:\n"; 19..for(i=0;i<4;i++) 20.for(j=0;j<4;j++) 21.cin>>a[i][j]; 22.cout<<"\nenter (2) array:\n"; 23.for(i=0;i<4;i++) 24.for(j=0;j<4;j++) 25.cin>>b[i][j]; 26.cout<<"\nenter (3) array:\n"; 27.for(i=0;i<4;i++) 28.for(j=0;j<4;j++) 29.cin>>c[i][j]; 30.vowel(a); 31.vowel(b); 32.vowel(c);}</pre>	<pre>#include<stdio.h> 1.char vowelchar [7]={'a','o','u','i','e','n','\o'}; 2..int i,j,k; 3.int indexofprintarray=1; 4.vowel(char a[][4]) 5.{ printf("\nvowel Char in array(%d)\n ",indexofprintarray) ; 6.indexofprintarray=indexofprintarray+1; 7.int number_appear ; 8. number_appear =0; 9.for(k=0;k<6;k++) 10.for(i=0;i<4;i++) 11.for(j=0;j<4;j++) 12.if(a[i][j]== vowelchar [k]) 13.{printf("\n%d",a[i][j]); 14.number_appear = number_appear +1;} 15.printf("\nnumber appear of vowel=%d", number_appear);}} 16.main() 17.{char a[4][4],b[4][4], c[4][4]; 18.printf("\nenter (1) array:\n"); 19..for(i=0;i<4;i++) 20.for(j=0;j<4;j++) 21 scanf("%d",&a[i][j]); 22. printf("\nenter (2) array:\n"); 23.for(i=0;i<4;i++) 24.for(j=0;j<4;j++) 25. scanf("%d",&b[i][j]); 26. printf("\nenter (3) array:\n"); 27.for(i=0;i<4;i++) 28.for(j=0;j<4;j++) 29. scanf("%d",&c[i][j]); 30.vowel(a); 31.vowel(b); 32.vowel(c);}</pre>

توضيح الخطوات: خطوة رقم (٣) عرفنا متغير (indexofprintarray) كمتغير عام ونلاحظ انه كلما نستدعي الدالة (vowel) في الخطوة رقم (٤) ستزداد قيمته بواحد في خطوة رقم (٦) لأنه كما قلنا المتغير العام لا يموت يبقى حيا حتى نهاية البرنامج ولا يرجع إلى قيمه الابتدائية (أي بعد ثلاث استدعائات لخطوة رقم (٣٠ و ٣١ و ٣٢) تصبح قيمته أربعة) ونحن احتاجيناه حتى في كل استدعاء يطبع رقم المصفوفة الجاري البحث فيها في خطوة رقم (٥)

وبقية الخطوات واضحة ومشروحة سابقا

الفصل السادس

المؤشرات (pointer)

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصول السابقة وفهماً كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على المؤشرات وطريقة التعامل مع المتغيرات عن طريق المواقع

مستوى الأداء المطلوب بعد إنهاء الفصل

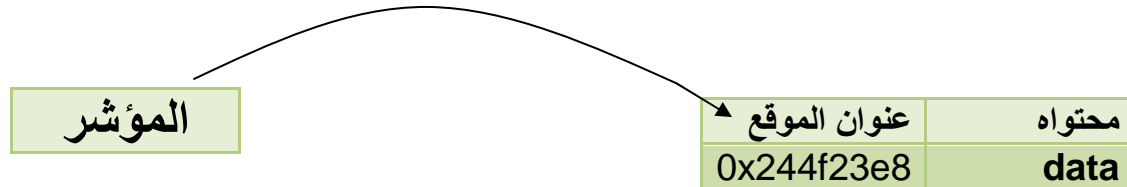
إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب : ثلاث ساعات

المؤشرات:

هو إشارة أو تأشير على موقع في الذاكرة. عرفنا سابقا إن كل متغير أو كل عنصر في المصفوفة يخزن في الذاكرة بموقع ذو عنوان معين والمؤشر سوف يؤشر على هذا العنوان ويمكننا من الوصول إلى القيمة المخزنة في داخله.



المؤشر يؤشر على عنوان المتغير في الذاكرة ويحمل قيمة هذا العنوان وهو هنا (0x244f23e8)

المؤشرات والمتغيرات:

المتغيرات تخزن في الذاكرة ويمكن أن نؤشر على قيمها بواسطة (pointer) ونتلاعب بقيمة هذا الموقع بواسطته وتكون طريقة تعريف (pointer) مع المتغيرات هكذا.

هيكلية المؤشر

Type *ptr=&var

- (ptr) : هو اسم المؤشر وقد يكون أي اسم يعرفه المستخدم ويجب وضع علامة النجمة (*) قبله
- (Type): هو نوع المؤشر ويكون نفسه نوع المتغير الذي يؤشر عليه
- (var): هو اسم المتغير الذي نريد أن يؤشر عليه المؤشر ويجب وضع علامة (&) قبله عندما نجعل المؤشر يؤشر عليه لان وضع هذه العلامة قبل أي متغير معناه المطلوب عنوان المتغير وليس قيمة المتغير نفسه والمؤشرات تؤشر على العناوين المتغيرات فلو وضعنا هذه العلامة أمام أي متغير في الطباعة سيطبع موقع المتغير وليس قيمته.

مثال: لو كان لدينا متغير (x=5) ويؤشر عليه مؤشر (ptr) بالشكل التالي فيكون المخطط هكذا

كود

```
int x=5;
int *ptr=&x;
```



- الموقع الذي يؤشر عليه (ptr) هو (0x244f23e8) ومحتواه هو (5)
- للوصول على عنوان الموقع نكتب (ptr) أو نكتب (&x) لان الاثنان أصبح لهما نفس الموقع.
- للوصول على محتويات الموقع نكتب (*ptr) أو نكتب (x) لان الاثنان أصبح لهما نفس القيمة.

مثال: تأشير على موقع متغير وطباعة قيمته.

البرمجة بلغة	c	البرمجة بلغة	C++
#include<stdio.h> main() 1.{int x=5; 2.int *ptr=&x; 3.printf("location Ptr=%d",ptr); 4.printf("\nlocation var=%d ",&x); 5.printf("\nvalue Ptr=%d ",*ptr); 6.printf("\n value var=%d",x); }	#include<iostream.h> main() 1.{int x=5; 2.int *ptr=&x; 3.cout<<"location Ptr="<<ptr; 4.cout<<"\nlocation var="<<&x; 5.cout<<"\nvalue Ptr="<<*ptr; 6.cout<<"\n value var="<<x; }		

توضيح الخطوات:

١. خطوة رقم (٢) جعلنا المؤشر (ptr) يؤشر على موقع المتغير (x)

٢. خطوة رقم (٣) طبعنا عنوان الموقع الذي يؤشر عليه المؤشر (ptr) وهو نفس عنوان موقع المتغير (x) لأنهما يؤشران على نفس الموقع. أي أننا إذا كتبنا فقط اسم المؤشر سيطبع الموقع الذي يؤشر عليه

٣. خطوة رقم (٤) طبعنا عنوان المتغير (x) لأننا إذا وضعنا علامة (&) قبل أي متغير سيطبع موقعه

٤. خطوة رقم (٥) طبعنا القيمة التي يؤشر عليها المؤشر (ptr) وهي نفس قيمة المتغير (x) لأنهما يؤشران على نفس الموقع. أي أننا إذا كتبنا نجمة قبل اسم المؤشر (ptr) سيطبع محتوى الموقع الذي يؤشر عليه

```
(Inactive C:\TCWIN45\BIN\NONAME)
location Ptr=0x24cf2420
location var=0x24cf2420
value Ptr=5
value var=5
```

٥. خطوة رقم (٦) سيطبع قيمة المتغير (x) شاهد شاشة التنفيذ

مثال : تغيير محتويات المتغير (g) بواسطة المؤشر.؟

البرمجة بلغة	c	البرمجة بلغة	C++
#include<stdio.h> main() 1.{int g=5; 2.int *ptr=&g; 3. *ptr=32; 4.printf("\nx=%d ",g);}	#include<iostream.h> main() 1.{int g=5; 2.int *ptr=&g; 3. *ptr=32; 4.cout<<"\nx="<<g;}		

توضيح الخطوات:

١. خطوة رقم (٢) جعلنا المؤشر يشير إلى موقع المتغير (g)

٢. خطوة رقم (٣) وضعنا قيمة جديدة في الموقع الذي يؤشر عليه المؤشر (*ptr) وهو موقع المتغير (g)



(NEW): هي دالة تستخدم لحجز مكان في الذاكرة لمؤشر معين لان المؤشرات بطبيعتها سوف توشر على مواقع متغيرات ولن تحجز مواقع أما مع هذا الإيعاز نستطيع حجز موقع للمؤشر وتعريفها بالشكل التالي

هيكلية المؤشر

Type *ptr=new type[size]

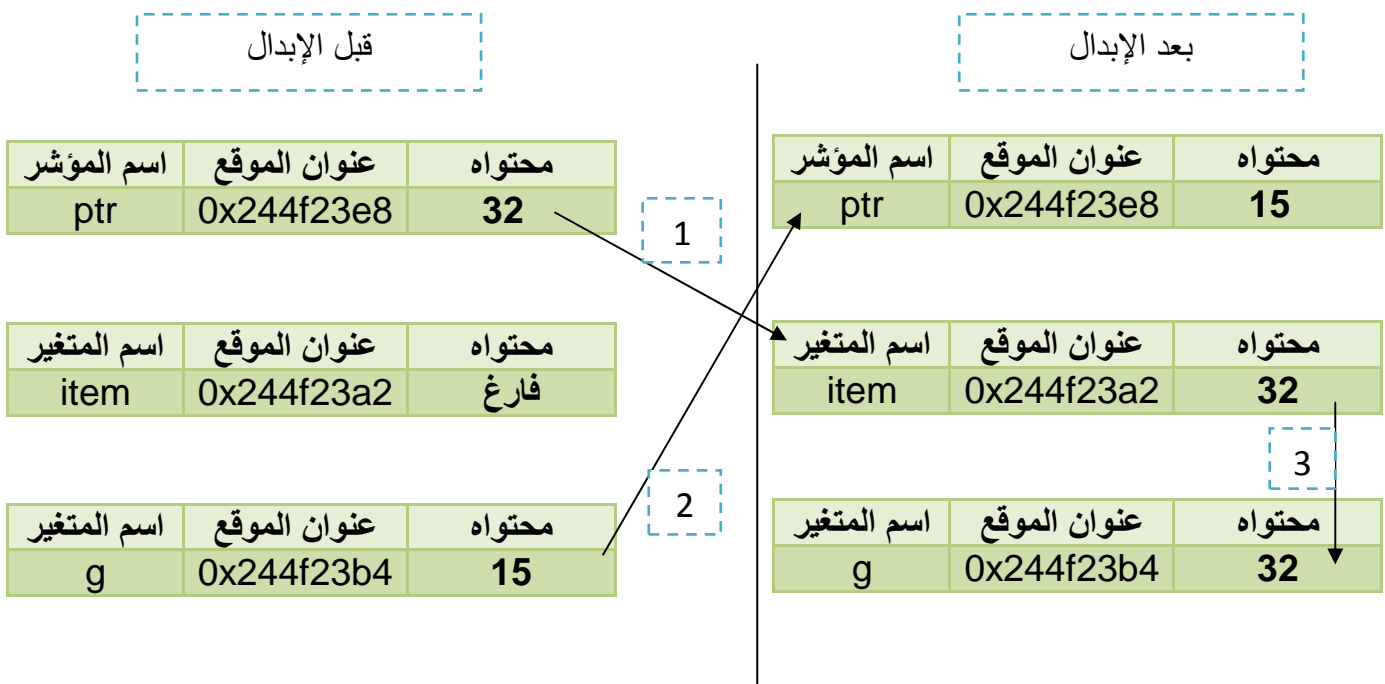
- **(ptr)** : هو اسم المؤشر وقد يكون أي اسم يعرفه المستخدم ويجب وضع علامة النجمة (*) قبله
- **(Type)**: هو نوع المؤشر ويكون نفسه نوع المتغير الذي يؤشر عليه
- **(size)**: هو الحجم أو عدد المواقع الذي سوف نحجزه للمؤشر في الذاكرة .
-

مثال : تكوين مؤشر جديد وحجز مكان جديد له وإبدال بين محتوى المؤشر ومحتويات المتغير (g).

C++	البرمجة بلغة	C	البرمجة بلغة
<pre>#include<iostream.h> main() 1.{int g=15,item; 2.int *ptr=new int [1]; 3. *ptr=32; 4.item=*ptr; 5.*ptr=g; 6.g=item; 7.cout<<"\ng="<<g<<"\n*ptr="<<*ptr;}</pre>		<pre>#include<stdio.h> main() 1.{int g=15,item; 2.int *ptr=new int [1]; 3. *ptr=32; 4.item=*ptr; 5.*ptr=g; 6.g=item; 7.printf("\ng=%d\n*ptr=%d",g,*ptr);}</pre>	

توضيح الخطوات:

1. خطوة رقم (٢) جعلنا المؤشر يشير إلى موقع جديد. وخطوة لرقم (٣) خزنا بالموقع قيمة (٣٢)
2. خطوة رقم (٤) وضعنا قيمة المؤشر (*ptr) في متغير مؤقت للإبدال
3. خطوة رقم (٥) وضعنا قيمة المتغير (g) في الموقع الذي يؤشر عليه المؤشر (*ptr)
4. خطوة رقم (٦) وضعنا قيمة المؤشر التي خزناها في متغير مؤقت إلى متغير (g)





المؤشرات والمصفوفات:

المصفوفات الأحادية والمؤشرات. عرفنا أن المصفوفة الأحادية هي مجموعة من المواقع المتتالية المحجوزة في الذاكرة ويمكن أن نستخدم المؤشر مع المصفوفة الأحادية وجعله يؤشر على احد القيم وسهولة تمريره على جميع العناصر فقط نزيد قيمة عنوان المؤشر بواحد فينتقل المؤشر ليؤشر على الموقع التالي الذي يليه.

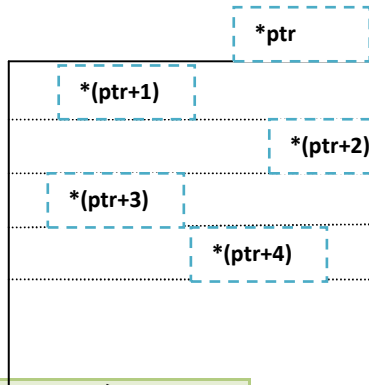
مثال توضيحي: لو عرفنا المصفوفة التالية حجمها خمسة عناصر وجعلنا المؤشر يؤشر على أول عنصر بالمصفوفة

كود

```
int first_array[5]={34,26,43,23,54};
int *ptr=& first_array[0];
```

واقترضنا أن أول عنصر في المصفوفة خزن بموقع (18126)

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0]	18126	34
first_array [1]	18127	26
first_array [2]	18128	43
first_array [3]	18129	23
first_array [4]	18130	54
	18131	data



اسم المؤشر	العنوان الذي يؤشر عليه
ptr	18126

- لاحظ أن المؤشر (ptr) يؤشر على (first_array [0]) أي على الموقع (18126)
- إذا أردنا أن يؤشر المؤشر (ptr) على الموقع الثاني بالمصفوفة نزيد قيمة الموقع الذي يؤشر عليه المؤشر (ptr) بمقدار واحد هكذا

كود

```
ptr+1 → 18126+1 → 18127
```

ليصبح المؤشر يؤشر على الموقع (18127) وهو عنوان ثاني موقع بالمصفوفة

- إذا أردنا أن يؤشر المؤشر (ptr) على الموقع الرابع بالمصفوفة نزيد قيمة الموقع الذي يؤشر عليه المؤشر (ptr) بمقدار ثلاثة هكذا . ليصبح المؤشر يؤشر على الموقع (18129) وهو عنوان رابع موقع بالمصفوفة

كود

```
ptr+1 → 18126+3 → 18129
```

مثال: مصفوفة مكونة من خمسة عناصر أصف مقدار (٤ ١) لكل عنصر باستخدام المؤشرات..؟

البرمجة بلغة	c	البرمجة بلغة	C++
#include<stdio.h> main() 1.{ int i, first_array[5]={34,26,43,23,54}; 2.int *ptr=& first_array[0]; 3. for(i=0;i<5;i++){ 4. *(ptr+i)=*(ptr+i)+14; 4.printf("\n first_array[%d]=%d",i,*(ptr+i));}}		#include<iostream.h> main() 1.{ int i, first_array[5]={34,26,43,23,54}; 2.int *ptr=& first_array[0]; 3. for(i=0;i<5;i++){ 4. *(ptr+i)=*(ptr+i)+14; 5.cout<<"\n first_array["<i<<"]="<<*(ptr+i);}}	

توضيح الخطوات:

- خطوة رقم (٢) جعلنا المؤشر (**ptr**) يؤشر على عنوان أول عنصر في المصفوفة
- خطوة رقم (٣) هو عداد يعد من (4—0) ويكرر خطوة رقم (٤ و ٥) في كل عدة
- خطوة رقم (٤) هو إضافة مقدار (٤ ١) لكل موقع من مواقع المصفوفة فمثلا عند الإضافة في الموقع الثالث تكون قيمة (i=2) فتكون خطوة رقم (٤) هكذا

كود

4. *(ptr+i)=*(ptr+i)+14; → *(ptr+2)=*(ptr+2)+14;

وبما أن المؤشر في خطوة رقم (٢) يؤشر على أول عنصر في المصفوفة فعد إضافة قيمة (٢) إلى عنوان الموقع سوف يؤشر على ثالث موقع بالمصفوفة.

مثال: مصفوفة مكونة من ثلاث عناصر أجمعها باستخدام المؤشرات..؟

البرمجة بلغة	c	البرمجة بلغة	C++
#include<stdio.h> main() 1.{ int sum=0, first_array[3]={ 43,23,54}; 2.int *ptr=& first_array[0]; 3. sum+=*ptr++; 4. sum+=*ptr++; 5. sum+=*ptr++; 6.printf("\n sum=%d ",sum);}		#include<iostream.h> main() 1.{ int sum=0, first_array[3]={ 43,23,54}; 2.int *ptr=& first_array[0]; 3. sum+=*ptr++; 4. sum+=*ptr++; 5. sum+=*ptr++; 6.cout<<"\n sum="<<sum;}	

توضيح الخطوات:

- خطوة رقم (٢) جعلنا المؤشر (**ptr**) يؤشر على أول عنصر في المصفوفة
- خطوة رقم (٣) أضفنا قيمة أول موقع إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد ليؤشر على العنصر الثاني (لو تلاحظ الزيادة بعد الجمع إي يجمع قيمة الموقع الأول ثم ينقل المؤشر للموقع الثاني)
- خطوة رقم (٤) أضفنا قيمة ثاني موقع إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد ليؤشر على العنصر الثالث
- خطوة رقم (٥) أضفنا قيمة ثالث موقع إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد



المصفوفات الثنائية والمؤشرات. عرفنا أن المصفوفة الثنائية هي مجموعة من المواقع المتتالية المحجوزة في الذاكرة ويمكن أن نستخدم المؤشر مع المصفوفة الثنائية وجعله يؤشر على احد القيم وسهولة تمريره على جميع العناصر فقط نزيد قيمة عنوان المؤشر بواحد فينتقل المؤشر ليؤشر على الموقع التالي الذي يليه.

مثال توضيحي: لو عرفنا المصفوفة التالية حجمها (2*2) وجعلنا المؤشر يؤشر على أول عنصر بالمصفوفة

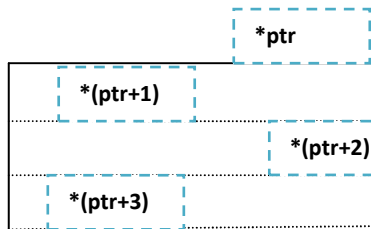
```

كود
int first_array[2][2]={{34,26},{43,23}};
int *ptr=& first_array[0][0];

```

واقترضنا أن أول عنصر في المصفوفة خزن بموقع (18126)

مواقع خلايا الذاكرة		
مواقع عناصر المصفوفة	الموقع	محتواه
	18125	data
first_array [0][0]	18126	34
first_array [0][1]	18127	26
first_array [1][0]	18128	43
first_array [1][1]	18129	23
	18131	data



اسم المؤشر	العنوان الذي يؤشر عليه
ptr	18126

- لاحظ أن المؤشر (ptr) يؤشر على (first_array [0][0]) أي على الموقع (18126)
- إذا أردنا أن يؤشر المؤشر (ptr) على الموقع الثاني بالمصفوفة (أي الصف الأول العمود الثاني) نزيد قيمة الموقع الذي يؤشر عليه المؤشر (ptr) بمقدار واحد هكذا

```

كود
ptr+1 → 18126+1 → 18127

```

ليصبح المؤشر يؤشر على الموقع (18127) وهو عنوان ثاني موقع بالمصفوفة

- إذا أردنا أن يؤشر المؤشر (ptr) على الموقع الرابع بالمصفوفة (أي الصف الثاني العمود الثاني) نزيد قيمة الموقع الذي يؤشر عليه المؤشر (ptr) بمقدار ثلاثة هكذا . ليصبح المؤشر يؤشر على الموقع (18129) وهو عنوان رابع موقع بالمصفوفة

```

كود
ptr+1 → 18126+3 → 18129

```

- تعامل المصفوفة الثنائية نفس معاملة المصفوفة الأحادية بالمؤشرات لان كلاهما عبارة عن خلايا متسلسلة محجوزة بالذاكرة



مثال: مصفوفة حجمها (2*2) أضرب كل عنصر بمقدار (٤) باستخدام المؤشرات..؟

البرمجة بلغة	c	البرمجة بلغة	C++
	<pre>#include<stdio.h> main() 1.{ int i, j,first_array[2][2]={{34,26},{43,23} }; 2.int *ptr=& first_array[0][0]; 3. for(i=0;i<2*2;i++) 4. *(ptr+i)=*(ptr+i)*4; 5.for(i=0;i<2 ;i++){ 6.for(j=0;j<2 ;j++){ 7.printf("%d\t", first_array[i][j]); 8.printf("\n");}}</pre>		<pre>#include<iostream.h> main() 1.{ int i, j,first_array[2][2]={{34,26},{43,23} }; 2.int *ptr=& first_array[0][0]; 3. for(i=0;i<2*2;i++) 4. *(ptr+i)=*(ptr+i)*4; 5.for(i=0;i<2 ;i++){ 6.for(j=0;j<2 ;j++){ 7.cout<< first_array[i][j]; 8.cout<<"\n";}}</pre>

توضيح الخطوات:

- خطوة رقم (٢) جعلنا المؤشر (ptr) يؤشر على عنوان أول عنصر في المصفوفة
- خطوة رقم (٣) هو عداد يعد من (0—4) ويكرر خطوة رقم (٤ و ٥) في كل عدة
- خطوة رقم (٤) هو ضرب مقدار (٤) لكل موقع من مواقع المصفوفة فمثلا عند الإضافة في الموقع الثالث تكون قيمة (i=2) فتكون خطوة رقم (٤) هكذا

كود

4. *(ptr+i)=*(ptr+i)*4; → *(ptr+2)=*(ptr+2)*4;

وبما أن المؤشر في خطوة رقم (٢) يؤشر على أول عنصر في المصفوفة فعد إضافة قيمة (٢) إلى عنوان الموقع سوف يؤشر على ثالث موقع بالمصفوفة. خطوة رقم (٥ و ٦ و ٧ و ٨) هو طباعة لعناصر المصفوفة بعد الضرب

مثال: مصفوفة حجمها (2*2) أجمعها باستخدام المؤشرات..؟

البرمجة بلغة	c	البرمجة بلغة	C++
	<pre>#include<stdio.h> main() 1.{ int sum=0, first_array[2][2]={{34,26},{43,23} }; 2.int *ptr=& first_array[0][0]; 3. sum+=*ptr++; 4. sum+=*ptr++; 5. sum+=*ptr++; 6. sum+=*ptr++; 7.printf("\n sum=%d ",sum);}</pre>		<pre>#include<iostream.h> main() 1.{ int sum=0, first_array[2][2]={{34,26},{43,23} }; 2.int *ptr=& first_array[0][0]; 3. sum+=*ptr++; 4. sum+=*ptr++; 5. sum+=*ptr++; 6. sum+=*ptr++; 7.cout<<"\n sum="<<sum;}</pre>

توضيح الخطوات:

- خطوة رقم (٣) أضفنا قيمة أول موقع (0,0) إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد ليؤشر على العنصر الثاني (0,1) (لو تلاحظ الزيادة بعد الجمع إي يجمع قيمة الموقع الأول ثم ينقل المؤشر للموقع الثاني)
- خطوة رقم (٤) أضفنا قيمة ثاني موقع (0,1) إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد ليؤشر على العنصر الثالث (1,0)
- خطوة رقم (٥) أضفنا قيمة ثالث (1,0) موقع إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد ليؤشر على العنصر الرابع (1,1)
- خطوة رقم (٦) أضفنا قيمة رابع موقع (1,1) إلى قيمة المتغير (sum) وزودنا قيمة الموقع بمقدار واحد .



المؤشرات والدوال (function)

١. **الدوال والمتغيرات:** عرفنا طريقة إرسال متغير بالقيمة كيف وبالمرجع كيف تكون. المؤشرات تمكنك من إرسال المتغير بالمرجع أي إرسال موقع المتغير وهذه تفيد إذا كنا نريد أن نعيد أكثر من متغير إلى البرنامج الرئيسي ونعلم إن الإرسال بالمرجع يكون أي تغيير على المتغير في الدوال يؤثر على قيمته في البرنامج الرئيسي

لو كان لدينا متغير اسمه (a) ونريد إرساله إلى دالة اسمها (name) .

إرسال مصفوفة مؤشر إلى الدوال (function)

```
name( &a );
```

- وضع علامة (&) قبله معناه أننا أرسلنا عنوان أول موقع .

وطريقة استقبال المتغير في الدوال نعرف مؤشر من نفس نوع المتغير المرسل لكي يؤشر على موقعه .

استقبال مصفوفة مؤشر إلى الدوال (function)

```
Type name(type *ptr);
```

- **(type):** هو نوع المتغير المرسل
- الآن أصبح هذا المؤشر (**ptr**) يؤشر على عنوان المتغير.

مثال: تكوين دالة تستقبل وحرف صغير وتحوله إلى حرف كبير..؟

تحليل: لتحويل حرف من صغير إلى كبير نعلم أن الفرق بين أسكي كود كل حرف صغير ونضيره الكبير هو (٣٢) لذلك لتحويل إلى حرف كبير نحول الحرف إلى أسكي كود ونطرح منه (٣٢) ونرجع نحوله إلى حرف فيتحول إلى حرف كبير

الحل باستخدام المؤشرات.....!

C++	البرمجة بلغة c	البرمجة بلغة
<pre>#include<iostream.h> 1.inline toBigLeter (char *ptr) 2.{*ptr=int(*ptr)-32;} 3.main() 4. { char inputchar; 5.cin>> inputchar; 6.toBigLeter (& inputchar); 7.cout<<"Big to it is="<< inputchar;}</pre>	<pre>#include<stdio.h> 1.inline toBigLeter (char *ptr) 2.{*ptr=int(*ptr)-32;} 3.main() 4. { char inputchar; 5 scanf("%c",&inputchar); 6.toBigLeter (& inputchar); 7.printf("Big to it is=%c", inputchar);}</pre>	

الحل بدون استخدام المؤشرات!.....!

C++	البرمجة بلغة c	البرمجة بلغة
<pre>#include<iostream.h> 1.inline toBigLeter (char charrec) 2.{ charrec =int(charrec)-32;} 3.main() 4. { char inputchar; 5.cin>> inputchar; 6.toBigLeter (inputchar); 7.cout<<"Big to it is="<< inputchar;}</pre>	<pre>#include<stdio.h> 1. inline toBigLeter (char charrec) 2.{ charrec =int(charrec)-32;} 3.main() 4. { char inputchar; 5 scanf("%c",&inputchar); 6.toBigLeter (inputchar); 7.printf("Big to it is=%c", inputchar);}</pre>	

توضيح الخطوات بالنسبة للحلين:

- خطوة رقم (١) نسبة إلى الحل باستخدام المؤشرات: إعلان عند دالة سطره تستقبل موقع الحرف وتكبر الحرف في خطوة رقم (٢)
- خطوة رقم (١) نسبة إلى الحل بدون استخدام المؤشرات: إعلان عند دالة سطره تستقبل الحرف وتكبر الحرف في خطوة رقم (٢)
- خطوة رقم (٦) نسبة إلى الحل باستخدام المؤشرات: إرسال عنوان المتغير (inputchar) إلى الدالة هكذا



- خطوة رقم (٦) نسبة إلى الحل بدون استخدام المؤشرات: إرسال قيمة المتغير (inputchar) إلى الدالة يكون هكذا (إذا قام المستخدم بإدخال الحرف a مثلا فيكون الإرسال هكذا)



- خطوة رقم (٧) نسبة إلى الحل باستخدام المؤشرات: سوف يطبع الحرف بعد التكبير لأننا في المؤشرات نتعامل مع موقع المتغير فأني تغير في الدالة على الموقع يغير في قيمة المتغير في البرنامج الرئيسي



- خطوة رقم (٦) نسبة إلى الحل بدون استخدام المؤشرات: سوف يطبع الحرف بدون أي تكبير نفس الحرف المدخل سوف يطبعه لأننا أرسلنا نسخة من المتغير إلى الدالة ولم نرسل المتغير نفسه فأني تغير على هذه النسخة لا يؤثر بقيمة المتغير في البرنامج الرئيسي

٢. **الدوال والمصفوفات الأحادية:** عرفنا سابقا طريقة التعامل مع المصفوفات في الدوال (function) و عرفنا طريقة الإرسال إلى الدالة وهي إرسال نسخة من المصفوفة وليس المصفوفة الأصلية أي كان إرسال بالقيمة. أما مع المؤشرات يكون إرسال بالمرجع أي أن أي تغيير على المصفوفة داخل أي دالة (function) سوف يؤثر على المصفوفة الأصلية في البرنامج الرئيسي التي أرسلت إلى الدالة للمعالجة لأن المؤشرات تتعامل مع مواقع الذاكرة أي مواقع المتغيرات وليس نسخة منه

ترسل المصفوفة الأحادية إلى الدالة بتحديد عنوان الموقع المرسل كأن يكون أننا نرسل عنوان أول موقع وفي الدالة عندما يعرف أول موقع يستطيع التنقل إلى باقي المواقع بزيادة قيمة المؤشر بواحد كل مرة . هكذا ترسل

مثال: لو كان لدينا مصفوفة اسمها (a) ونريد إرسالها إلى دالة اسمها (name) .

إرسال مصفوفة مؤشر إلى الدوال (function)

```
name( &a [0]);
```

- في هذه الطريقة أرسلنا عنوان أول موقع بوضع علامة (&) قبله .

و طريقة استقبال المصفوفة الأحادية في الدوال نعرف مؤشر من نفس نوع المصفوفة المرسله .

استقبال مصفوفة مؤشر إلى الدوال (function)

```
Type name(type *ptr);
```

- **(type):** هو نوع المصفوفة المرسله
- ألان أصبح هذا المؤشر (ptr) يؤشر على عنوان أول موقع بالمصفوفة

****** في إرسال المصفوفة إلى الدوال قد نرسل عنوان أول موقع أو نرسل عنوان آخر موقع أو أي موقع حسب ما نحتاجه في برنامجنا

مثال : تكوين دالة تعكس تسلسل أحرف ثلاث مصفوفات بالعكس وتطبعهم بعد العكس .؟

تحليل: بما انه يريد عكس الأحرف في المصفوفة باستخدام الدوال فيجب الإرسال بالمرجع حتى عندما يقلب أحرف السلسلة وعندما ينتهي من (function) ويعود للبرنامج الرئيسي تعكس المصفوفات أيضا في البرنامج الرئيسي .

C++	البرمجة بلغة c
#include<iostream.h> #include<string.h> 1.int i; 2.Reverse(char *string,int len) 3.{char item; 4.for(i=len; i>len/2;i--){ 5.item=* (string); 6.*(string)=*(string -i+(len-i)); 7. *(string -i+(len-i))= item; 8. string --;} } 9.main() 10. { char string1[55], string2[55], string3[55]; 11.int len; 12.cout<<"enter string (1): "; 13.cin.getline(string1,55); 14.len=strlen(string1)-1; 15.Reverse(&string1[len], len); 16.cout<< string1 ; 17.cout<<"\nenter string (2): "; 18.cin.getline(string2,55); 19.len=strlen(string2)-1; 20.Reverse(&string2[len], len); 21.cout<< string2 ; 22.cout<<"\n enter string (3): "; 23.cin.getline(string3,55); 24.len=strlen(string3)-1; 25.Reverse(&string3[len], len); 26.cout<< string3 ;}	#include<stdio.h> #include<string.h> 1.int i; 2.Reverse(char *string,int len) 3.{char item; 4.for(i=len; i>len/2;i--){ 5.item=* (string); 6.*(string)=*(string -i+(len-i)); 7. *(string -i+(len-i))= item; 8. string --;} } 9.main() 10. { char string1[55], string2[55], string3[55]; 11.int len; 12.printf("enter string (1): "); 13.gets (string1); 14.len=strlen(string1)-1; 15.Reverse(&string1[len], len); 16.printf("%s",string1); 17.printf("\n enter string (2): "); 18.gets (string2); 19.len=strlen(string2)-1; 20.Reverse(&string2[len], len); 21.printf("%s",string2); 22.printf("\n enter string (3): "); 23.gets (string3); 24.len=strlen(string3)-1; 25.Reverse(&string3[len], len); 26.printf("%s",string3);}

توضيح الخطوات:

١.خطوة رقم (٢) هي دالة تستقبل آخر موقع بالسلسلة وطول هذه السلسلة

٢.خطوة رقم (٤) هو عداد يبدأ بالعد من آخر موقع إلى المنتصف حتى يبدل المواقع الأول بالأخير والثاني بالقبل الأخير في خطوات رقم (٥ و٦ و٧ و٨) ويستمر بالإبدال

٣.خطوة رقم (١٣) هي إدخال السلسلة الأولى وخطوة (١٤) حساب طولها

٤.خطوة رقم (١٥) هي إرسال آخر موقع بالسلسلة وطول السلسلة

بقية الخطوات واضحة ومكررة.هذه صورة من شاشة التنفيذ

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter string (1): he go to home known
nwonk emoh ot og eh
enter string (2): ali is big man
nam gib si ila
enter string (3): he died fo his live
evil sih of deid eh
```

٣. **الدوال والمصفوفات الثنائية:** عرفنا سابقا طريقة التعامل مع المصفوفات في الدوال (function) وعرّفنا طريقة الإرسال إلى الدالة وهي إرسال نسخة من المصفوفة الثنائية وليس المصفوفة الأصلية أي كان إرسال بالقيمة. أما مع المؤشرات يكون إرسال بالمرجع أي أن أي تغيير على المصفوفة داخل أي دالة (function) سوف يؤثر على المصفوفة الأصلية في البرنامج الرئيسي التي أرسلت إلى الدالة للمعالجة لأن المؤشرات تتعامل مع مواقع الذاكرة أي مواقع المتغيرات وليس نسخة منه

ترسل المصفوفة الثنائية إلى الدالة بتحديد عنوان الموقع المرسل كأن يكون أننا نرسل عنوان أول موقع وفي الدالة عندما يعرف أول موقع يستطيع التنقل إلى باقي المواقع بزيادة قيمة المؤشر بواحد كل مرة . هكذا ترسل

مثال: لو كان لدينا مصفوفة اسمها (a) حجمها (4*4) ونريد إرسالها إلى دالة اسمها (name) .

إرسال مؤشر مصفوفة ثنائية إلى الدوال (function)

```
name( &a [0][0]);
```

- في هذه الطريقة أرسلنا عنوان أول موقع بوضع علامة (&) قبله .

وطريقة استقبال المصفوفة الثنائية في الدوال نعرف مؤشر من نفس نوع المصفوفة المرسله .

استقبال مؤشر مصفوفة ثنائية إلى الدوال (function)

```
Type name(type *ptr);
```

- (type): هو نوع المصفوفة المرسله
- الآن أصبح هذا المؤشر (ptr) يؤشر على عنوان أول موقع بالمصفوفة

** في إرسال المصفوفة إلى الدوال قد نرسل عنوان أول موقع أو نرسل عنوان آخر موقع أو أي موقع حسب ما نحتاجه في برنامجنا

مثال : تكوين دالة تضع واحد مكان العدد الأولي في مصفوفة (4*4) وصفر مكان العدد الغير أولي ؟.

تحليل: بما إننا نريد تحويل المصفوفة إلى أصفار و واحدات نرسلها كمؤشر إلى داله وهناك العدد الأولي نضع مكانه واحد والغير أولي نضع صفر

C++	البرمجة بلغة c
<pre>#include<iostream.h> 1.int i,j; 2.int row=4; 3.int col=4; 4.prime2d (int *string) 5.{int prime=1; 6.for(i=0; i<row*col ;i++){ 7.prime=1; 8.for(j=2;j<*string;j++) 9.if(*string % j==0) 11.prime=0; 12.*string= prime; 13. string ++;} } 14.main() 15.{ int string1[4][4]; 16. for(i=0; i<row ;i++) 17.for(j=0; j<col ;j++) 18.cin>> string1[i][j]; 19.prime2d(&string1[0] [0]); 20.for(i=0; i<row ;i++){ 21.for(j=0; j<col ;j++) 22.cout<< string1[i][j]<<"\t"; 23.cout<<"\n";}}</pre>	<pre>#include<stdio.h> 1.int i,j; 2.int row=4; 3.int col=4; 4.prime2d (int *string) 5.{int prime=1; 6.for(i=0; i<row*col ;i++){ 7.prime=1; 8.for(j=2;j<*string;j++) 9.if(*string % j==0) 11.prime=0; 12.*string= prime; 13. string ++;} } 14.main() 15.{ int string1[4][4]; 16. for(i=0; i<row ;i++) 17.for(j=0; j<col ;j++) 18.scanf("%d",& string1[i][j]); 19.prime2d(&string1[0] [0]); 20.for(i=0; i<row ;i++){ 21.for(j=0; j<col ;j++) 22.printf("%d\t", string1[i][j]); 23.printf("\n");}}</pre>

توضيح الخطوات:

١. خطوة رقم (٤) دالة تستقبل عنوان أول عنصر بالمصفوفة

٢. خطوة رقم (٥) هو متغير إذا كان الرقم الذي عليه المؤشر عدد أولي يبقى واحد وذا كان الرقم غير أولي يتحول إلى صفر لكي يخزن بدل قيمة العنصر

٣. خطوة رقم (٦) هو عدد يمر على جميع عناصر المصفوفة لكي يتحقق من العناصر عنصر عنصر في خطوات رقم (٧ و ٩ و ١٠ و ١١ و ١٢) هل العنصر عدد أولي أم لا

٤. خطوة رقم (١٣) لكي ينقل المؤشر على العنصر التالي بعد أن يتحقق من العنصر السابق

٥. خطوة رقم (١٩) هي إرسال عنوان أول عنصر بالمصفوفة بالمصفوفة إلى الدالة

شاهد شاشة التنفيذ

```
(Inactive) C:\TCWIN45\BIN\NONAME01.EXE
5 65 45 45
56 45 45 57
98 98 56 43
45 31 12 7
1 0 0 0
0 0 0 0
0 0 0 1
0 1 0 1
```



مصفوفة أحادية غير محدودة الحجم: تعلمنا سابقا أن المصفوفات حجمها ثابت ويجب تعريفه ولا يمكن كتابة مصفوفة دون تحديد حجمها أما مع المؤشرات نستطيع مع دالة (new) تكوين مصفوفة غير محدودة الحجم يحدد حجمها المستخدم وقت التنفيذ .حيث أن الحجم الذي نحجزه بقدر حجم المصفوفة المطلوب

لو أردنا أن نحجز مصفوفة حجمها (٩) عناصر من نوع integer وقت التنفيذ نحجز بشكل التالي

```
حجز مصفوفة وقت التنفيذ
int *Array=new int [9];
```

أي كأنما نقول المؤشر (*Array) يؤشر على مكان في الذاكرة حجمه تسعة

مثال : برنامج يطلب من المستخدم تحديد حجم المصفوفة وقت التنفيذ وبعدها يطلب منه أن يدخل عناصر ثم يجمع العناصر ويجد المعدل ؟

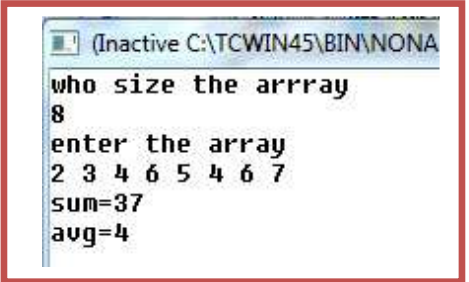
C++	البرمجة بلغة	C	البرمجة بلغة
<pre>#include <iostream.h> void main() 1.{int sizearray,j,sum,avg; 2.sum=0; 3.cout<<"who size the arrray\n" ; 4.cin>> sizearray ; 5.int *Array=new int [sizearray] ; 6.cout<<"enter the array\n" ; 7.for (j=0;j<sizearray; j++) 8. cin>> Array[j] ; 9.for (j=0;j<sizearray; j++) 10. sum=sum+Array[j]; 11. avg=sum/sizearray; 12. cout<<"sum="<<sum<<"\navg= "<<avg; }</pre>		<pre>#include <stdio.h> void main() 1.{int sizearray,j,sum,avg; 2.sum=0; 3.printf("who size the arrray\n"); 4.scanf("%d",& sizearray); 5.int *Array=new int [sizearray] ; 6.printf("enter the array\n"); 7.for (j=0;j<sizearray; j++) 8. scanf("%d",&Array[j]); 9.for (j=0;j<sizearray; j++) 10. sum=sum+Array[j]; 11. avg=sum/sizearray; 12.printf("sum=%d\navg=%d",sum,avg); }</pre>	

توضيح الخطوات :

١. خطوة رقم (٤) يطلب من المستخدم إدخال حجم المصفوفة لكي يضع الحجم الذي يدخله المستخدم في خطوة رقم (٥) إي مثلا إذا ادخل المستخدم الرقم خمسة ستكون الخطوة رقم (٤) بشكل التالي

```
حجز مصفوفة وقت التنفيذ
int *Array=new int [5];
```

إي حجز خمسة مواقع في الذاكرة



لاحظ الشكل التالي لهذا الإدخال في شاشة التنفيذ .



مصفوفة ثنائية غير محدودة الأبعاد: نستطيع مع دالة (new) تكوين مصفوفة ثنائية غير محدودة الأبعاد يحدد أبعادها المستخدم وقت التنفيذ .

لو أردنا أن نحجز مصفوفة حجمها (3*5) من نوع integer وقت التنفيذ نحجز بشكل التالي

```

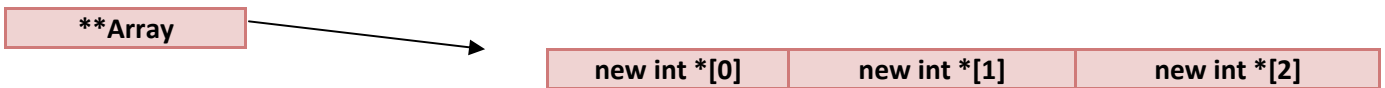
حجز مصفوفة وقت التنفيذ
1.int k;
2.int **Array=new int *[row];
3.for (k=0 ; k< row ; k++)
4.Array[k]=new int[columns];
    
```

• (row): هو عدد الصفوف

• (columns): هو عدد الأعمدة

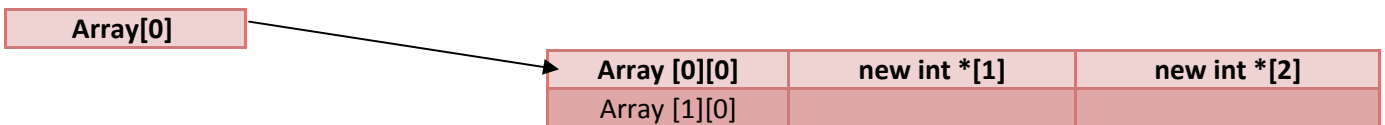
تكون طريق الحجز بشكل التالي مثلا لمصفوفة (3*2):

١. في خطوة رقم (٢) يُوَشر المؤشر (Array) على مصفوفة مؤشرات حجمها بقدر عدد الصفوف أي (٣)

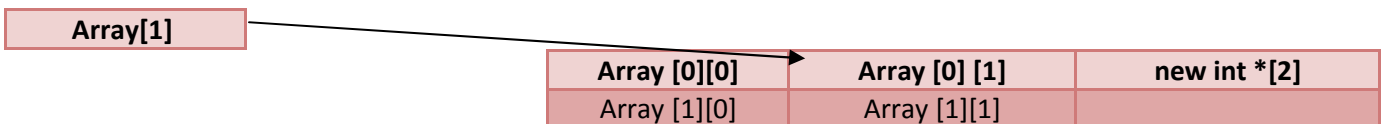


٢. خطوة رقم (٣) يبدأ بالتحرك على كل صف يُوَشر عليه المؤشر (Array) ويكون له أعمدة في خطوة رقم (٤)

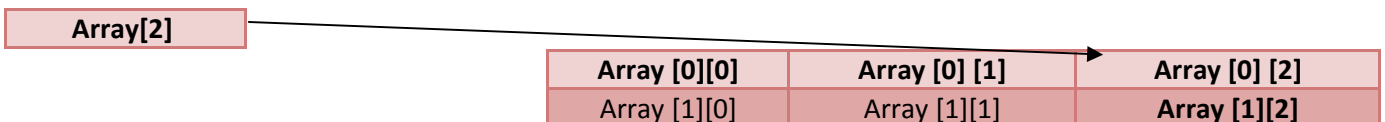
A. عندما تكون قيمة (k=0) سيُوَشر المؤشر (Array) على أول موقع بصف ويكون العمود له طوله (٢)



B. عندما تكون قيمة (k=1) سيُوَشر المؤشر (Array) على ثاني موقع بصف ويكون العمود له طوله (٢)



C. عندما تكون قيمة (k=2) سيُوَشر المؤشر (Array) على ثالث موقع بصف ويكون العمود له طوله (٢)

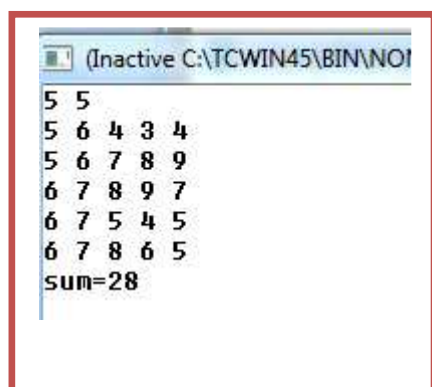


مثال : مصفوفة ثنائية غير محدودة الحجم يحدد حجمها المستخدم وقت التنفيذ ويجمع عناصر القطر الرئيسي ؟

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include <iostream.h> void main () 1.{ int i,j,k,sum,rowN,colN; 2. sum=0; 3.cin>>rowN>>colN ; 4.int **Array=new int *[rowN]; 5.for (k=0 ; k< rowN ; k++) 6.Array[k]=new int[colN]; 7.for (i=0 ; i< rowN ; i++) 8.for (j=0; j< colN ; j++) 9. cin>>Array[i][j] ; 10.for (i=0 ; i< rowN ; i++) 11. for (j=0; j< colN ; j++) 12.if (i==j) 13.sum=sum+ Array[i][j]; 14. cout<<"sum="<<sum ; }</pre>	<pre>#include <stdio.h> void main () 1.{ int i,j,k,sum,rowN,colN; 2. sum=0; 3 scanf("%d%d",&rowN,&colN); 4.int **Array=new int *[rowN]; 5.for (k=0 ; k< rowN ; k++) 6.Array[k]=new int[colN]; 7.for (i=0 ; i< rowN ; i++) 8.for (j=0; j< colN ; j++) 9. scanf("%d",&Array[i][j]); 10.for (i=0 ; i< rowN ; i++) 11. for (j=0; j< colN ; j++) 12.if (i==j) 13.sum=sum+ Array[i][j]; 14. printf("sum=%d",sum); }</pre>

توضيح الخطوات:

١. خطوة رقم (٣) هي إدخال عدد صفوف المصفوفة وعدد أعمدتها
٢. خطوة رقم (٤ و٥ و٦) تكوين المصفوفة
٣. خطوة رقم (٧ و٨ و٩) هي إدخال المصفوفة بالإبعاد التي حددناها
٤. خطوة رقم (١٠ و١١ و١٢ و١٣) هي إيجاد عناصر القطر الرئيسي وجمعها
٥. خطوة رقم (١٤) طباعة ناتج الجمع



```
(Inactive C:\TCWIN45\BIN\NOI
5 5
5 6 4 3 4
5 6 7 8 9
6 7 8 9 7
6 7 5 4 5
6 7 8 6 5
sum=28
```

لو أدخلنا مصفوفة حجمها (5*5) من شاشة التنفيذ بشكل التالي

الفصل السابع

التراكيب (Structures)

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصول السابقة وفهما كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على السجلات وطرق استخدامها

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب : ثلاث ساعات

١. التراكيب (Structures)

Structure أو **Structs** هي مجموعة بيانات (متغيرات) بأنواع مختلفة تحت اسم واحد. تستخدم في حال لدينا عدة مكونات أو أشخاص يشتركون في معلومات معينة متشابهة فتستخدم لجمع تعاريف لعدة أشخاص في سجل واحد يشتركون جميعاً بنفس المعلومات. حيث نشترك هذه المعلومات لأي شخص نريده. وتكون بشكل التالي

هيكلية تراكيب (Structs)

```
struct Structures_Name
{
Type var1;
Type var2;
.
.
.
}ObjectName1,Object_name2;
```

- **(Structures_Name)** هو اسم السجل ويمكن أن يكون أي اسم
- **(Type)** : هو نوع المتغير داخل السجل ويمكن وضع أنواع مختلفة من المتغيرات داخل سجل واحد. ويكون عدد تعريف المتغيرات غير محدد
- **(ObjectName)** : هو اسم الكائن المشتق من السجل ويمكن أن يكون أي اسم. ويمكن اشتقاق عدد غير محدد من الكائنات من سجل واحد فقط نضع فارزة بين كل كائن وآخر. ونبدأ بتعريف هذه الكائنات بعد إغلاق قوس السجل
- ويوضع السجل بعد تعريف المكتبات مباشرة.

السجلات تدخل ضمن مواضيع البرمجة كائنية التوجه

مثال: ايسط مثال على سجل هي سيارة (car) لها رقم لوحة و موديل و اسم الشركة المصنعة؟
تحليل: لتكوين سجل لهذه المعلومات الثلاثة تكتب ؟

تراكيب (Structs) لمكونات سيارة

```
struct car
{
int CarNumber;
int Model;
char factoryDesgin[20];
}HussienCar,WaeelCar;
```

نلاحظ أن رقم السيارة عرف كرقم لأنه رقم السيارة عبارة عن مجموعة أرقام و موديلها عرف رقم لان الموديل عبارة عن تاريخ وشركة المصنعة عرفت كسلسلة لان الشركة اسم المصنعة مكونة من أحرف أو رموز ونلاحظ أننا اشتقينا من السجل سيارة حسين (HussienCar) وسيارة وائل (WaeelCar) ونريد أن ندخل معلومات عن سيارة كل شخص ضمن البرنامج الرئيسي فيكون الكود بشكل التالي

تراكيب (Structs) لمكونات سيارة إدخال المعلومات

```
main()
{
HussienCar. CarNumber =18475;
HussienCar. Model=2011 ;
Strcpy(HussienCar. factoryDesgin,"BMW");
WaeelCar. CarNumber =75645;
WaeelCar. Model=2005 ;
Strcpy(WaeelCar. factoryDesgin,"KIA");
}
```

لوصول لمعلومات أي كائن نكتب اسم الكائن ثم نقطة (.) ثم معلومة التي نريد إدخالها أو طباعتها أو التعديل عليها أو معالجتها



- اسم الشركة المصنعة يكون عبارة عن مصفوفة أحرف فلا يمكن إسناد سلسلة أحرف مباشرة في السجل هكذا

إسناد قيم بطريقة خاطئة

```
HussienCar. factoryDesgin="BMW" ;
```

هذا التعبير خاطئ لذلك يجب نسخ الاسم باستخدام السلاسل كما في التالي.

نسخ سلسلة إلى سلسلة ضمن سجل

```
Strcpy(HussienCar. factoryDesgin,"BMW");
```

يمكن إدخال السلسلة بطريقة مباشرة من شاشة التنفيذ هكذا.....!

البرمجة بلغة	c	البرمجة بلغة	C++
	gets(HussienCar. factoryDesgin);		cin.get(HussienCar. factoryDesgin,20);

مثال: مجموعة من أربعة مستخدمين لكل مستخدم اسم وكلمة مرور ..؟

```

تراكيب ( Structs ) لسجل مستخدمين أربعة كل واحد له اسم وكلمة مرور
struct password_User
{
char userName[20];
int password;
}Hussien ,Waeel,Modar,Rafeed;

```

وكود البرنامج لمستخدم واحد مع الإدخال والطباعة لأسمه وكلمة مروره يكون

البرمجة بلغة	c	البرمجة بلغة	C++
	#include<stdio.h> #include<string.h> 1.struct password_User 2.{char userName[20]; 3.int password; }Hussien ,Waeel,Modar,Rafeed; 4.main() 5.{printf("enter hussien user name: "); 6. gets(Hussien. userName); 7. printf("\nenter hussien password: "); 8 scanf("%d",& Hussien. password); 9. printf("user name:%s ", Hussien. userName); 10 .printf(" \npassword: %d", Hussien. password);}		#include<iostream.h> #include<string.h> 1.struct password_User 2.{char userName[20]; 3.int password; }Hussien ,Waeel,Modar,Rafeed; 4.main() 5.{cout<<"enter hussien user name: "; 6.cin.get(Hussien. userName,25); 7.cout<<"\nenter hussien password: "; 8.cin>> Hussien. password; 9.cout<<"user name: "<< Hussien. userName ; 10.cout<<" \npassword: "<< Hussien. password ;}

توضيح الخطوات:

- خطوة رقم (١) هو سجل للمستخدمين يحوي اسم المستخدم في خطوة رقم (٢) وكلمة مروره في خطوة رقم (٣) ونلاحظ في خطوة رقم (٣) بعد أن أغلقنا السجل اشتقينا أسماء أربعة مستخدمين
- خطوة رقم (٦) هي إدخال اسم المستخدم وهو عبارة عن سلسلة
- خطوة رقم (٨) هي إدخال كلمة المرور ونلاحظ في كود لغة (c) استخدمنا ("%d") لان كلمة المرور عرفناها في خطوة رقم (٣) على أنها متغير (integer)
- خطوة رقم (٩) هي طباعة اسم المستخدم في شاشة التنفيذ ونلاحظ في كود لغة (c) استخدمنا الرمز (%s) لأننا أدخلنا اسم المستخدم بشكل سلسلة.

٢. التراكيب المتداخلة (Structure in Structure)

هي طريقة وضع سجل (Structure) داخل سجل آخر. الفائدة منها هي مثلا لو كان لدينا سجل يحوي رقم سيارة ونوعها واسم الشركة والمصنعة وكل سيارة لها ثلاث محركات بأسماء معينة فلو جعلنا أسماء المحركات داخل سجل وهو داخل سجل معلومات السيارة لكان كل وصول إلى سجل من سجلات السيارة تستطيع وصول إلى ثلاث أنواع المحركات فيكون البرنامج مرن وواضح . فتكون الهيكلية كالتالي

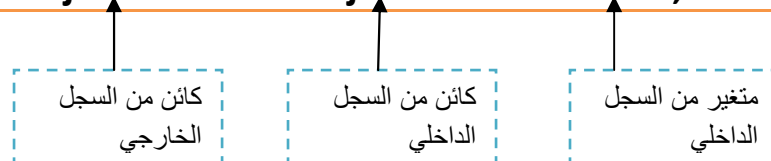
هيكلية تراكيب متداخلة

```
struct Structures_Name1
{
struct Structures_Name2
{
Type var21;
Type var22;
} ObjectName21, Object_name22;
Type var11;
Type var12;
.
.
}ObjectName11, Object_name12;
```

هنا (Structures_Name2) واقع هو ومحتوياته داخل (Structures_Name1) فإذا أردنا أن نصل إلى مكونات السجل الداخلي وهو (Structures_Name2) فنحتاج إلى كتابة كائن من السجل الخارجي ثم كائن من سجل الداخلي ثم مكونات السجل الداخلي . على سبيل المثال لو أرنا الوصول إلى (var21) نكتب

إسناد قيم بطريقة خاطئة

```
ObjectName11. ObjectName21.var21;
```



مثال: سجل لسيارة لها رقم وموديل واسم الشركة وداخلة سجل لثلاث محركات.؟

تراكيب (Structs) متداخلة لمكونات سيارة

```
struct car
{
  Struct machine
  {int Type1;
  int Type2;
  int Type3;
  }MachineCar;
  int CarNumber;
  int Model;
  char factoryDesgin[20];
  }HussienCar,WaeelCar;
```

- الآن لو أردنا الوصول إلى نوع (Type1) من المحركات في سيارة حسين (HussienCar) يكون الكود

كود

```
HussienCar . MachineCar.Type1;
```

لو تلاحظ كتبنا أولاً اسم الكائن في السجل الخارجي (HussienCar) ثم اسم الكائن في السجل الداخلي (MachineCar) ثم اسم المحرك من النوع الأول (Type1) وهذه هي طريقة الوصول الصحيحة

- الآن لو أردنا الوصول إلى نوع (Type1) من المحركات في سيارة وائل (WaeelCar) وإعطاء رقم (554) لهذه المحرك يكون الكود

كود

```
WaeelCar . MachineCar.Type1=554;
```

لو تلاحظ كتبنا أولاً اسم الكائن في السجل الخارجي (WaeelCar) ثم اسم الكائن في السجل الداخلي (MachineCar) ثم اسم المحرك من النوع الأول (Type1) وأسندنا له قيمة

- الآن لو أردنا الوصول إلى نوع (Type2) من المحركات في سيارة وائل (WaeelCar) وإعطاء رقم (5544) لهذه المحرك ولوصول لرقم سيارته وإعطائها رقم (4753) . يكون الكود

كود

```
WaeelCar . MachineCar.Type2=5544;
WaeelCar .CarNumber=4753
```

هذه كود لإدخال وطباعة معلومات سيارة واحدة فقط

C++	C
<pre> #include<iostream.h> #include<string.h> struct car {struct machine {int Type1; int Type2; int Type3; }MachineCar; int CarNumber; int Model; char factoryDesgin[20]; }HussienCar ; main() {cout<<"enter factory design: "; cin.get(HussienCar . factoryDesgin ,25); cout<<"\nenter car number: "; cin>> HussienCar. CarNumber ; cout<<"\nenter car model: "; cin>> HussienCar. Model ; cout<<"\nenter car machine Type1: "; cin>> HussienCar. MachineCar. Type1 ; cout<<"\nenter car machine Type2: "; cin>> HussienCar. MachineCar. Type2 ; cout<<"\nenter car machine Type3: "; cin>> HussienCar. MachineCar. Type3 ; cout<<" \nthe information for the car is \n"; cout<<" factory design : "<< HussienCar . factoryDesgin ; cout<<" \n car number : "<< HussienCar. CarNumber ; cout<<"\n car model: "<< HussienCar. Model; cout<<" \n machine Type1: "<< HussienCar. MachineCar. Type1 ; cout<<"\n machine Type2: "<< HussienCar. MachineCar. Type2 ; cout<<" \n machine Type3: "<< HussienCar. MachineCar. Type3 ; } </pre>	<pre> #include<stdio.h> #include<string.h> struct car {struct machine {int Type1; int Type2; int Type3; }MachineCar; int CarNumber; int Model; char factoryDesgin[20]; }HussienCar ; main() {printf("enter factory design: "); gets(HussienCar . factoryDesgin) ; printf("\nenter car number: "); scanf("%d",& HussienCar. CarNumber) ; printf("\nenter car model: "); scanf("%d",& HussienCar. Model) ; printf("\nenter car machine Type1: "); scanf("%d",& HussienCar. MachineCar. Type1) ; printf("\nenter car machine Type2: "); scanf("%d",& HussienCar. MachineCar. Type2) ; printf("\nenter car machine Type3: "); scanf("%d",& HussienCar. MachineCar. Type3) ; printf(" \nthe information for the car is \n"); printf(" factory design :%s ", HussienCar . factoryDesgin); printf(" \n car number :%d ", HussienCar. CarNumber) ; printf("\n car model: %d ", HussienCar. Model); printf(" \n machine Type1: %d ", HussienCar. MachineCar. Type1) ; printf("\n machine Type2: %d ", HussienCar. MachineCar. Type2) ; printf(" \n machine Type3:%d ",HussienCar. MachineCar. Type3); } </pre>

الكود واضح وليس بحاجة إلى أي شرح أو توضيح

٣. مصفوفة تراكيب (Structures)

عرفنا المصفوفات سابقا وعرفنا من اهم فوائدها هي الخزن المؤقت وتخزين عدد من القيم أو الحروف بشكل مصفوفة. إذن مصفوفة سجلات هي بدلا من أن نعرف عدد من الكائنات من كل سجل نعرف كائن واحد من نوع مصفوفة بديلا عن كل هذه الكائنات. فعلى سبيل المثال لو عدنا إلى مثال السيارة وأردنا أربعين مستخدم هل نعرف أربعين كائن نعرف كائن واحد من نوع مصفوفة وحجمه أربعين. فتكون الهيكلية هكذا

هيكلية مصفوفة تراكيب (Structs)

```
struct Structures_Name
{
Type var1;
Type var2;
.
.
.
}ObjectName1[size];
```

• (ObjectName1[size]): هي كائن وواحد من نوع مصفوفة له حجم معين يحدده المستخدم.

فلنعود إلى مثال معلومات سيارة وهي رقم ونوع واسم الشركة لكن نريدها لأربعين سيارة فيكون السجل بشكل التالي

تراكيب (Structs) لمكونات سيارة

```
struct car
{
int CarNumber;
int Model;
char factoryDesgin[20];
}userCar[40];
```

لو أردنا الوصول إلى معلومات صاحب أول سجل (الوصول إلى اسم الشركة المصنعة)

كود

```
userCar[0]. factoryDesgin ;
```

- لو أردنا الوصول إلى معلومات صاحب ثاني سجل (الوصول إلى اسم الشركة المصنعة)

كود

```
userCar[1]. factoryDesgin ;
```

وكذلك البقية نصل إليهم بنفس الطريقة.

- لو أردنا تعبئة معلومات صاحب أول سيارة يكون الكود

كود

```
strcpy(userCar[0]. factoryDesgin,"BMW") ;
```

```
userCar[0]. Model=2011;
```

```
userCar[0]. CarNumber=45356;
```

**الكود كامل لإدخال معلومات أربعين سيارة وطباعتها

البرمجة بلغة	c	البرمجة بلغة
C++		
#include<iostream.h> #include<stdio.h> struct car {int CarNumber; int Model; char factoryDesgin[20]; }userCar[40]; main() {int i; for (i=0;i<40;i++) {cout<<"\ninformation for("<<(i+1)<<") car : "; cout<<"enter Car factory: "; gets(userCar[i]. factoryDesgin); cout<<"enter Car number: "; cin>> userCar[i]. CarNumber ; cout<<"enter Car Model: "; cin>> userCar[i]. Model;} cout<<"\n-----the information saved...\n"; for (i=0;i<40;i++) {cout<<"\ninformation save e ("<<(i+1)<<") car : "; cout<<" Car factory: "; cout<< userCar[i]. factoryDesgin ; cout<<" Car number: "; cout<< userCar[i]. CarNumber ; cout<<" Car Model: "; cout<< userCar[i]. Model;}}	#include<stdio.h> struct car {int CarNumber; int Model; char factoryDesgin[20]; }userCar[40]; main() {int i; for (i=0;i<40;i++) {printf("\ninformation for(%d) car : ",(i+1)); printf("enter Car factory: "); gets(userCar[i]. factoryDesgin); printf("enter Car number: "); scanf("%d",& userCar[i]. CarNumber) ; printf("enter Car Model: "); scanf("%d",& userCar[i]. Model);} printf("\n-----the information saved...\n"); for (i=0;i<40;i++) { printf("\ninformation save(%d) car : ",(i+1)); printf(" Car factory: "); printf("%s",userCar[i]. factoryDesgin) ; printf(" Car number: "); printf("%d",userCar[i]. CarNumber) ; printf(" Car Model: "); printf("%d", userCar[i]. Model);}}	

توضيح: لو تلاحظ استخدمنا عداد يعد حتى (٤٠) حتى ندخل جميع معلومات السيارات بالنسبة إلى

كل (userCar[i]) عدة للعداد نقوم بإدخال سجل جديد..؟

مصفوفة تراكيب (Structures) متداخلة

لا يختلف شيئا عن تركيب داخل تركيب نفس الأسلوب لكن هنا نستخدم المصفوفات فتكون مصفوفة تراكيب داخل مصفوفة تراكيب و الهيكلية تكون بشكل التالي

هيكلية تراكيب متداخلة

```
struct Structures_Name1
{
struct Structures_Name2
{
Type var21;
Type var22;
} ObjectName21[size2];
Type var11;
Type var12;
.
.
}ObjectName11[size1];
```

فكل كائن من كائنات مصفوفة السجلات (ObjectName11[size1]) يحوي على سجل (ObjectName21[size2]) بحجم (size2) على سبيل المثال لو كان لدينا المثال التالي

مثال: سجل لعشر سيارات لها رقم وموديل واسم الشركة وداخلة سجل لخمس مكائن لكل ماكنة ثلاث محركات؟

تراكيب (Structs) متداخلة لمكونات سيارة

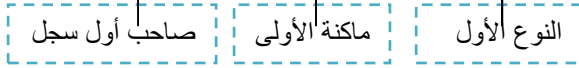
```
struct car
{
Struct machine
{int Type1;
int Type2;
int Type3;
}MachineCar[5];
int CarNumber;
int Model;
char factoryDesgin[20];
} userCar[10];
```


- لو أردنا الوصول إلى معلومات صاحب أول سجل ماكينة لأولى النوع الأول

كود

```
userCar[0]. MachineCar[0]. Type1 ;
```

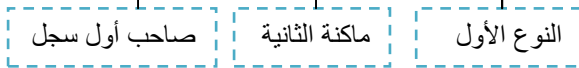
وكذلك البقية نصل إليهم بنفس الطريقة.



- لو أردنا الوصول إلى معلومات صاحب أول سجل ماكينة ثانية النوع الأول

كود

```
userCar[0]. MachineCar[1]. Type1 ;
```



- لو أردنا الوصول إلى معلومات صاحب ثاني سجل ماكينة أولى النوع الثاني

كود

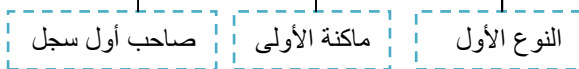
```
userCar[1]. MachineCar[0]. Type2 ;
```



- لو أردنا الوصول إلى معلومات صاحب أول سجل ماكينة لأولى النوع الأول ونعطيه قيمة (20)

كود

```
userCar[0]. MachineCar[0]. Type1=20;
```



- لو أردنا الوصول إلى معلومات صاحب أول سجل اسم الشركة المصنعة ونعطيه (BMW)

كود

```
Strcpy(userCar[0]. factoryDesgin,"BMW") ;
```

وكذلك البقية نصل إليهم بنفس الطريقة.

وهذا كود المثال كامل

البرمجة بلغة C++	البرمجة بلغة C
<pre> #include<iostream.h> #include<stdio.h> 1.struct car 2.{struct machine 3.{int Type1; 4.int Type2; 5.int Type3; 6.}MachineCar[5]; 7.int CarNumber; 8.int Model; 9.char factoryDesgin[20]; 10.} userCar[10]; 11.main() 12.{int i, cont ; 13.for (i=0;i<10;i++) 14.{ 15.cout<<"\ninformation for("<<(i+1)<<") car : "; 16.cout<<"enter Car factory: "; 17. gets(userCar[i]. factoryDesgin); 18.cout<<"enter Car number: "; 19.cin>> userCar[i]. CarNumber ; 20.cout<<"enter Car Model: "; 21. cin>> userCar[i]. Model; 22.for(cont =0; cont <5; cont ++) 23.{ 24.cout<<"\nmachine name("<<(cont +1)<<") : "; 25.cout<<"\nenter car machine Type1: "; 26.cin>> userCar[i]. MachineCar[cont]. Type1 ; 27.cout<<"\nenter car machine Type2: "; 28.cin>> userCar[i]. MachineCar[cont]. Type2 ; 29.cout<<"\nenter car machine Type3: "; 30.cin>> userCar[i]. MachineCar[cont]. Type3 ; 31.} 32.} 33.} </pre>	<pre> #include<stdio.h> 1.struct car 2.{struct machine 3.{int Type1; 4.int Type2; 5.int Type3; 6.}MachineCar[5]; 7.int CarNumber; 8.int Model; 9.char factoryDesgin[20]; 10.} userCar[10]; 11.main() 12.{int i, cont ; 13.for (i=0;i<10;i++) 14.{ 15.printf("\ninformation for(%d) car : ",(i+1)); 16.printf("enter Car factory: "); 17. gets(userCar[i]. factoryDesgin); 18.printf("enter Car number: "); 19 scanf("%d",& userCar[i]. CarNumber) ; 20.printf("enter Car Model: "); 21 scanf("%d",& userCar[i]. Model); 22.for(cont =0; cont <5; cont ++) 23.{ 24. printf("\n nmachine name (%d): ",(cont +1)); 25. printf("\nenter car machine Type1: "); 26. printf("%d",userCar[i]. MachineCar[cont]. Type1) ; 27. printf("\nenter car machine Type2: "); 28. printf("%d", userCar[i]. MachineCar[cont]. Type2) ; 29. printf("\nenter car machine Type3: "); 30. printf("%d",userCar[i]. MachineCar[cont]. Type3) ; 31.} 32.} 33.} </pre>

توضيح البرنامج :

نلاحظ في كل عدة للعداد في خطوة رقم (١٣) تتكرر الخطوات من (١٤ الى ٣٢) ففي الخطوات من (١٦ إلى ٢١) يدخل معلومات السجل الخارجي وخطوة رقم (٢٢) هو عدد بعدد سجل المكنائن الداخلي المكون من خمس مكنائن للأنواع الثلاث ويدخل كل هذه المكنائن للأنواع الثلاث في خطوة (٢٥ إلى ٣٠) لكل سجل من سجلات السجل الخارجي

الفصل الثامن

الملفات (File)

المستوى المطلوب

أن يكون القارئ ملماً بما هو في الفصول السابقة وفهما كل شيء

الأهداف:

عندما يكتمل الفصل تكون بإذن الله قد أتممت التعرف على الملفات وطرق استخدامها

مستوى الأداء المطلوب بعد إنهاء الفصل

إتقان هذه الفصل 100%

الأدوات المطلوبة: حاسوب شخصي لتجربة البرامج وقلم ودفتر لتسجيل الملاحظات

الوقت المطلوب: ثلاث ساعات

الملفات (File):

من احد سمات اللغة (C, C++) توفر إمكانية تخزين مكونات أو نتائج أو متطلبات البرنامج على القرص الصلب (hard Disk) بشكل دائم أو مؤقت حسب الحاجة وتخزن الملفات بالقرص الصلب عن طريق لغة (C, C++) بامتدادات مختلفة منها (.txt , .bin) . من الأمثلة على استخدام الملفات لو كان لدينا برنامج لا يستطيع المستخدم الدخول له إلا بكلمة مرور فتخزن كلمة المرور في القرص الصلب بداخل ملف ومتى ما فتحنا البرنامج يطلب من المستخدم إدخال كلمة المرور وإذا كانت الكلمة المدخلة مشابهة للكلمة المخزنة يفتح له البرنامج إي خزننا كلمة المرور بشكل دائم داخل الحاسوب .

✓ الدوال التي تستخدم مع الملفات في لغة (C++) تقع ضمن مكتبة <iostream.h> و <fstream.h>

✓ الدوال التي تستخدم مع الملفات في لغة (C) تقع ضمن مكتبة <stdio.h>

الكتابة في الملفات النصية:

للكتابة الى داخل إي ملف توجد ثلاث خطوات وهي:

- خطوة الأولى نعرف كائن من نوع ملف
- خطوة ثانية نفتح الملف للكتابة
- الخطوة الثالثة نكتب الى داخل الملف

١. في لغة (C++) نستخدم الدالة التالية للكتابة إلى داخل الملفات

كتابة الى داخل ملف في لغة C++

```
1. ofstream fout ;  
2. fout.open("file path", iostream family);  
3. fout << "data";
```

- الخطوة الأولى عرفنا (fout) من نوع ملف كتابة هو اسم للملف الذي نريد الكتابة فيه. للتعامل معه داخل البرنامج ويمكن أن يكون أي اسم
- (file path) : هو مسار الملف المراد خزنة فيه داخل الجهاز يوضع بين علامتي تنصيص
- (iostream family) : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معينة

جدول برموز iostream family

الرمز	وظيفته
ios::app	يلحق الإدخال الجديد بنهاية الملف
ios::ate	يقوم بالقراءة أو الكتابة من نهاية الملف
ios::trunc	في حال وجود الملف فسيقوم ببيئها أي حذف محتوياتها
ios::in	فتح الملف للقراءة وهي حالة افتراضية لكائنات ifstream
ios::out	فتح الملف للكتابة وهي حالة افتراضية لكائنات ofstream
ios::binary	فتح الملفات على هيئة ثنائية وليس نصية

- لاستخدام أكثر من رمز في الحل نضع بينهم (|) مثلاً إذا اردنا نكتب ونلحق كتابتنا بمحتويات الملف السابق (ios::app | ios::out)
- وفي خطوة رقم (3) قمنا بإدخال البيانات إلى داخل الملف

٢. في لغة (c) نستخدم الدالة التالية للكتابة إلى داخل الملفات

كتابة الى داخل ملف في لغة C

```
1.FILE *fout;
2.fout=fopen("file path","symbol");
3.fputs("data", fout,);
```

- (fout) : هو اسم للملف الذي نريد الكتابة فيه. عرفناه في الخطوة رقم (١) انه ملف
- (file path) : هو مسار الملف المراد خزنة فيه داخل الجهاز يوضع بين علامتي تنصيص
- (symbol) : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معنية توضع بين علامتي تنصيص وهنا نستخدم الرمز (w) لأننا نريد الكتابة بداخل الملف. وهذا جدول بهذه الرموز

الرمز	وصيفته	جدول برموز symbol
a	يلحق الإدخال الجديد بنهاية الملف	
r	فتح الملف للقراءة وإذا كان الملف غير صالح تعيد قيمة صفر	
w	استحداث الملف للكتابة وإذا كان الملف موجود في القرص الصلب سيمسح محتوياته	
rb,wb,ab	كتابة وقراءة وإلحاق في الملفات الثنائية	
r+	فتح الملف للكتابة أو القراءة لكن في الإضافة يكتب البيانات الجديدة فوق البيانات السابقة	
w+	استحداث الملف للكتابة أو القراءة لكن في الإضافة يكتب البيانات الجديدة فوق البيانات السابقة	

لاستخدام أكثر من رمز في الحل نضع بينهم جمع مثلا إذا أردنا نكتب ونلحق كتابتنا بمحتويات الملف السابق (w+a)

- وفي خطوة رقم (٣) قمنا بإدخال البيانات إلى داخل الملف باستخدام الدالة (fput) التي تأخذ النص المراد كتابته بداخل الملف بشكل سلسلة و اسم الملف
- ✓ يمكن كتابة حرف واحد فقط الى داخل الملف باستخدام الدالة (putc) التي تأخذ اسم الملف والحرف المراد إدخاله

كتابة الى داخل ملف في لغة C

```
putc("data", fout,);
```

☒ بعد أكمل عملنا مع إي ملف يجب إغلاقه ولا يجوز تركه مفتوح ويغلق بشكل التالي

إغلاق ملف في لغة C++

```
fout.close;
```

نكتب اسم املف المستخدم داخل البرنامج ثم نغلقه

إغلاق ملف في لغة C

```
fclose(fout);
```

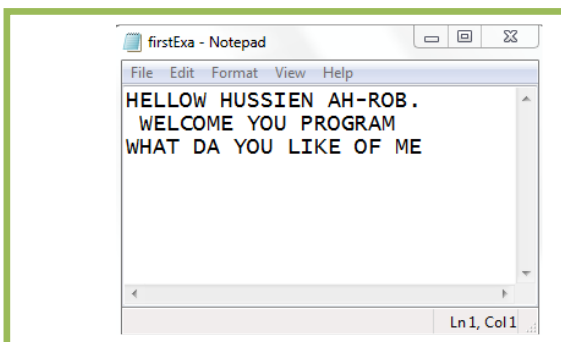
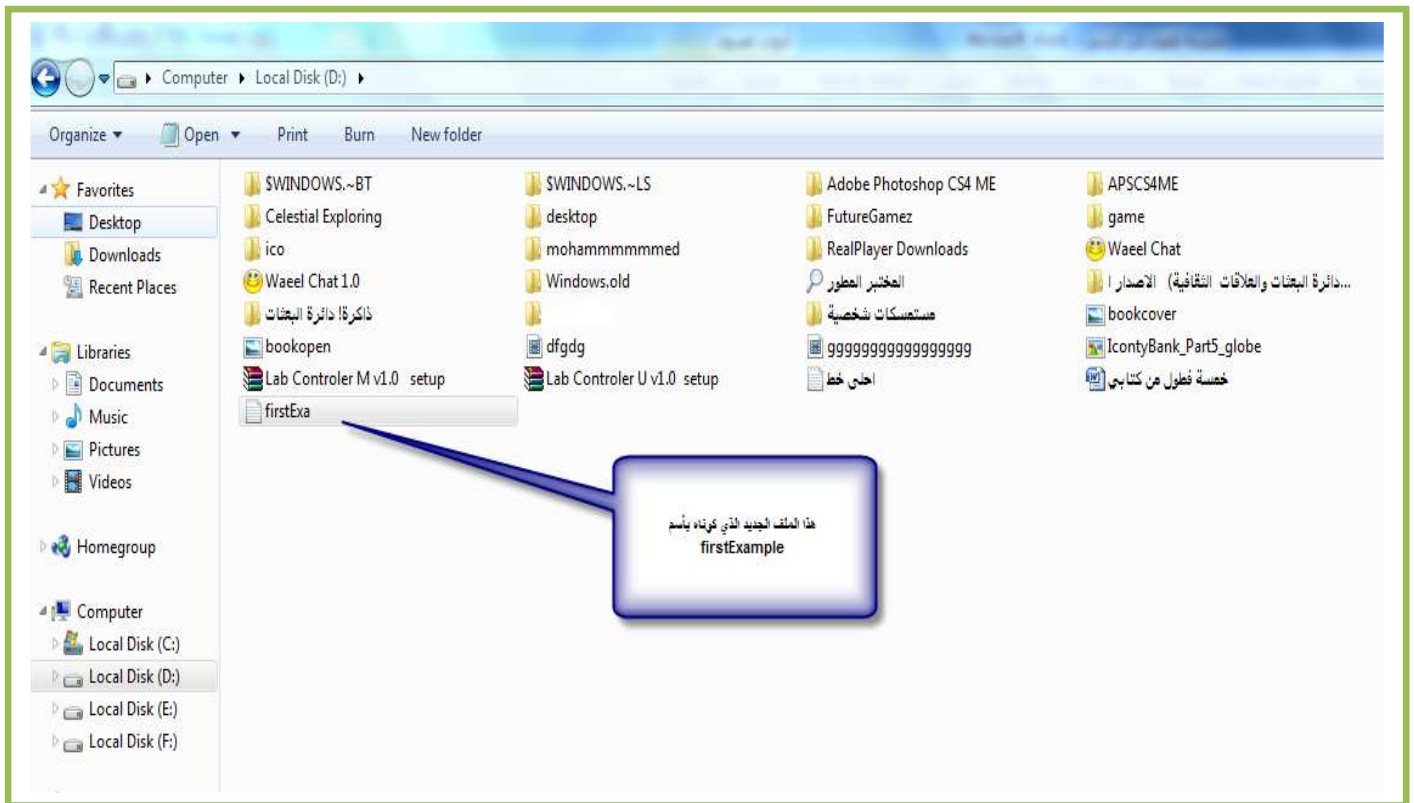
نستخدم دالة تأخذ اسم الملف داخل البرنامج لتغلقه

مثال : برنامج لكتابة جمل على أكثر من سطر داخل ملف .؟

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include <fstream.h> int main() 1. { ofstream fout; 2. fout.open("D:\\firstExa.txt"); 3. fout << "HELLOW HUSSIEN AH-ROB.\n" << "WELCOME YOU PROGRAM\n" << "WHAT DA YOU LIKE OF ME\n"; 4. fout.close(); }</pre>	<pre>#include <stdio.h> int main() 1.{ FILE * fout; 2. fout=fopen("D:\\firstExa.txt","w"); 3. fputs("HELLOW HUSSIEN AH-ROB.\n WELCOME YOU PROGRAM\n WHAT DA YOU LIKE OF ME\n ", fout); 4.fclose(fout); }</pre>

توضيح الخطوات

- خطوة رقم واحد عرفنا (fout) من نوع ملف، خطوة رقم (٢) كونا ملف بالمسار (d:\) للكتابة فيه
- خطوة رقم (٣) كتابنا في الملف لاحظ كيف نكتب بيانات على أكثر من سطر، خطوة رقم (٤) أغلقنا الملف لاحظ كيفية تكون الملف داخل جهاز الكمبيوتر



وإذا فتحنا الملف (firstExa) سنجد مكتوب فيه الأسطر التالية

خزنت فيه نفس الأسطر الذي كتبناها في

خطوة رقم (في المثال السابق إذا كان الملف **firstExa**) أصلاً مخزن فيه معلومات معينة فالذي سوف يحدث أنه سيحذف المحتويات السابقة ويضيف المحتويات الجديدة في خطوة رقم (3) بدلاً من المحتويات السابقة. وإذا أردنا أن لا يحذف المحتويات السابقة إنما يضيف المحتويات الجديدة خلفها في الملف فقط نغير الخطوة رقم (2) إلى

البرمجة بلغة	c	البرمجة بلغة	C++
	2. fout=fopen("D:\\firstExa.txt", "a");		2. fout.open("D:\\firstExa.txt", ios::app);

مثال: برنامج مذكرة نكتب في شاشة التنفيذ وهو يخزن ما نكتبه ويستمر البرنامج بالطلب من المستخدم بالكتابة إلى أن يدخل المستخدم النقطة (.).

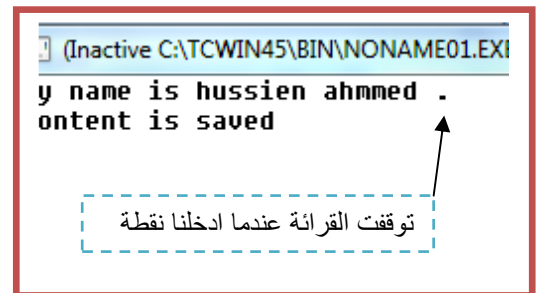
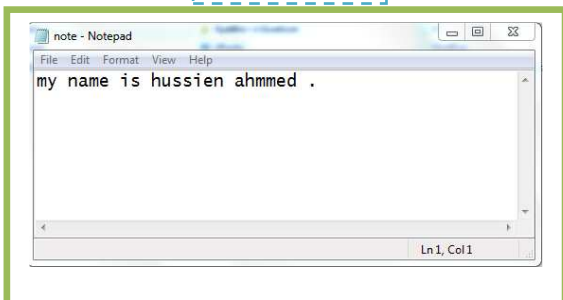
تحليل: من السؤال أنه يتوقف إذا أدخل المستخدم رمز النقطة (.). أي أن البرنامج يقرأ حرف من شاشة التنفيذ ويقارن الحرف المدخل بشرط التوقف في دارة (loop) مستمرة لا تتوقف إلا إذا أدخل المستخدم رمز النقطة وبما أننا ندخل حرف حرف ولا نريد أن يحس المستخدم أنه يدخل أحرف نستخدم الدالة (getche) في الإدخال لكي نقرأ ما يدخله المستخدم مباشرة. بما أنه يريد أن نخزن ما نكتبه باستمرار أي أنه في كل إدخال بعد إغلاق وفتح البرنامج يضيف الحرف الجديد المدخل إلى الملف ولا يمسح محتوياته إنما يلحق الكتابة الجديدة بنهاية الكتابة السابقة

البرمجة بلغة	c	البرمجة بلغة	C++
	<pre>#include <stdio.h> #include <conio.h> int main() 1. {char symbol ; 2. FILE * fout; 3. fout=fopen("D:\\note .txt", "a"); 4. do{ 5.symbol=getche(); 6. putc(symbol, fout); 7.} while(symbol != '.'); 8.printf("\ncontent is saved"); 9 fclose(fout); }</pre>		<pre>#include <fstream.h> #include <conio.h> int main() 2. {char symbol ; 3. ofstream fout; 3. fout.open("D:\\note.txt", ios::app); 4. do{ 5.symbol=getche(); 6.fout << symbol; 7.} while(symbol != '.'); 8.cout<<"\ncontent is saved"; 9. fout.close(); }</pre>

توضيح الخطوات

1. الخطوة رقم (3) فتحنا ملف للكتابة مع ميزة إلحاق الكتابة الجديدة بالكتابة السابقة
2. خطوة رقم (5) هو قراءة من شاشة التنفيذ
3. خطوة رقم (6) طباعة الحرف الذي تمت قراءته في الملف
4. خطوة رقم (4 إلى 7) تتكرر باستمرار مادام المستخدم لم يدخل رمز النقطة لأن شرط التوقف في الخطوة رقم (7) أن يكون الرمز المدخل هو نقطة

محتويات الملف



مثال: برنامج لخرن أسماء (٦) مستخدمين وكلمات مرورهم هي ملف ؟.
تحليل: نكون سجل حجمه ستة يحوي اسم مستخدم وكلمة مرور وندخله في ملف

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> #include<stdio.h> #include <fstream.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{int i; 6.for(i=0;i<6;i++){ 7.cout<<"\nenter user name: "; 8. gets(userProtection[i]. username); 9.cout<<"enter password: "; 10. gets(userProtection[i]. password);} 11.ofstream fout; 12. fout.open("D:\\up.txt"); 13.for(i=0;i<6;i++){ 14.fout << "User ID :"; fout <<<< userProtection [i]. username; fout <<<< "\tuser Password: "; fout <<<< userProtection [i]. password; fout <<<<"\n";} 15. fout.close(); }</pre>	<pre>#include<stdio.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{int i; 6.for(i=0;i<6;i++){ 7. printf("\nenter hussien user name: "); 8. gets(userProtection[i]. username); 9. printf("enter hussien password: "); 10.gets(userProtection[i]. password);} 11.FILE * fout; 12. fout=fopen("D:\\up.txt","w"); 13.for(i=0;i<6;i++){ 14.fprintf(fout,"User ID :"); fprintf(fout , userProtection [i]. username); fprintf(fout , "\tuser Password: "); fprintf(fout , userProtection [i]. password); fprintf(fout , "\n");} 15. fclose(fout); }</pre>

توضيح الخطوات:

١. خطوة رقم (١ و٢ و٤) كونا سجل حجمه ست مستخدمين فيه اسم وكلمة مرور
٢. خطوة رقم (٦ الى ١٠) هي إدخال أسماء المستخدمين وكلمات مرورهم
٣. خطوة رقم (١١) عرفنا ملف جديد وخطوة رقم (١٢) فتحنا الملف للكتابة فيه
٤. خطوة رقم (١٣) هو عداد لكي يدخل أسماء وكلمات مرور كل المستخدمين واحد يتلو الآخر في خطوة رقم (١٤)

سيخزنون في الملف في الشكل التالي

لو أدخلنا أسماء المستخدمين كما في شاشة التنفيذ

```
up - Notepad
File Edit Format View Help
User ID :hussien user Password: a35e
User ID :waeel user Password: 234e
User ID :modar user Password: 5443
User ID :Raffed user Password: f5433
User ID :aeed user Password: 2345
User ID :alxsmoanar user Password: 4533
Ln1, Col1
```

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter user name: hussien
enter password: a35e
enter user name: waeel
enter password: 234e
enter user name: modar
enter password: 5443
enter user name: Raffed
enter password: f5433
enter user name: aeed
enter password: 2345
enter user name: alxsmoanar
enter password: 4533
```


القراءة من الملفات النصية:

للقراءة من داخل إي ملف توجد ثلاث خطوات وهي.....!

- خطوة الأولى نعرف كائن من نوع ملف
 - خطوة ثانية نفتح الملف للقراءة
 - الخطوة الثالثة نقرأ من داخل الملف
١. في لغة (C++) نستخدم الدالة التالية للقراءة من داخل الملفات

قراءة من داخل ملف في لغة C++

```
1. ifstream fin ;  
2. fin.open("file path",iosstream family);  
3.fin.getline(array,80);
```

- الخطوة الأولى عرفنا (**fin**) من نوع ملف للقراءة. هو اسم للملف الذي نريد القراءة منه للتعامل معه داخل البرنامج ويمكن أن يكون أي اسم
- (**file path**) : هو مسار الملف المراد فتحه من داخل الجهاز يوضع بين علامتي تنصيص
- (**iosstream family**) : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معينة
- وفي خطوة رقم (3) قمنا بخزن سطر واحد من البيانات من الملف الى داخل مصفوفة

٢. في لغة (C) نستخدم الدالة التالية للقراءة من داخل الملفات

قراءة من داخل ملف في لغة C

```
1.FILE *f fin ;  
2. fin =fopen("file path","symbol");  
3.fgets( fin ,80, array );
```

- (**fout**) : هو اسم للملف الذي نريد القراءة منه. عرفناه في الخطوة رقم (١) انه ملف
- (**file path**) : هو مسار الملف المراد فتح الملف من داخل الجهاز يوضع بين علامتي تنصيص
- (**symbol**) : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معينة توضع بين علامتي تنصيص
- وهنا نستخدم الرمز (**r**) لأننا نريد القراءة من داخل الملف. وهذا جدول بهذه الرموز
- وفي خطوة رقم (٣) قمنا بقراءة البيانات من داخل الملف باستخدام الدالة (**fgets**) التي تأخذ اسم الملف والنص المراد القراءة منه وتخزن الناتج في مصفوفة و (80) هو عدد الأحرف المراد قراءتها من الملف
- ✓ يمكن قراءة حرف واحد فقط من داخل الملف باستخدام الدالة (**getc**) التي تأخذ اسم الملف وتعطي والحرف المراد قرائته

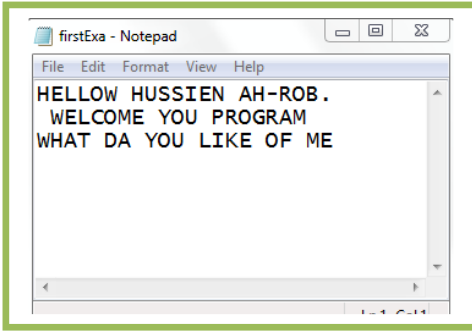
قراءة من داخل ملف في لغة C

```
var= getc(fin );
```

- ✓ (**fin**) : هو اسم الملف المراد القراءة منه
- ✓ (**var**) هو اسم المتغير المراد خزن القيمة المقروءة في داخله

☒ القراءة من الملف تبدأ بقراءة البايت الأول ثم الثاني وبالتالي

مثال : قم بقراءة جميع محتويات الملف الذي يقع في المسار التالي (D:\\firstExa.txt) ويحتوي الملف على البيانات التالية؟



```

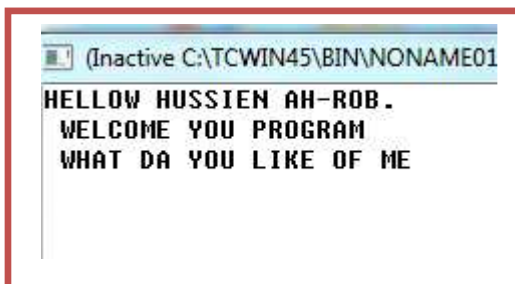
firstExa - Notepad
File Edit Format View Help
HELLOW HUSSIEN AH-ROB.
WELCOME YOU PROGRAM
WHAT DA YOU LIKE OF ME
    
```

تحليل: لقراءة جميع محتويات ملف نستخدم دوال التكرار في القراءة حيث كل ما يقرأ بايت ينتقل المؤشر إلى البايت الثاني ليقرئه في المرة القادمة لكن لا بد من وجود دالة توقف القراءة عند الوصول إلى نهاية الملف

البرمجة بلغة C++	البرمجة بلغة C
<pre> #include <fstream.h> #include <iostream.h> main() 1. { char array [80]; 2. ifstream fin; 3. fin.open("D:\\firstExa.txt"); 4.while(!fin.eof()) 5.{fin.getline(array,80); 6.cout<<array<<endl;} 7. fin.close(); } </pre>	<pre> #include <stdio.h> #include <stdlib.h> main() 8.{ char reading; 9. FILE * fin; 10.fin=fopen("D:\\firstExa.txt","r"); 11.while((reading=getc(fin)) !=EOF) 12.{ 13.printf("%c", reading);} 14 fclose(fin);} </pre>

توضيح الخطوات :

١. خطوة رقم (١) عرفنا مصفوفة لخرن الذي نقرئه فيه عند كل قراءة
 ٢. خطوة رقم (٨) عرفنا متغير حرفي لخرن الأحرف التي نقرئها حرف حرف
 ٣. خطوة رقم (٩ و ٢) عرفنا كائن من نوع ملف ،خطوة رقم (٣) فتحنا الملف للقراءة
 ٤. خطوة رقم (٣ و ١٠) فتحنا الملف للقراءة
 ٥. خطوة رقم (١١ و ٤) هي دوار لا تتوقف إلا أن ينتهي من قراءة آخر سطر في الملف (eof) هي مختصر لكلمة (end of file) هو مؤشر على نهاية الملف أي هنا استخدمناه كشرط توقف إذا قرء آخر سطر يتوقف حتى لا يستمر بالقراءة إلى ألما لانهاية
 ٦. خطوة رقم (٥) نقرأ سطر سطر من الملف تقع الدالة ضمن مكتبة
 ٧. خطوة رقم (٦) هي طباعة سطر سطر وكلمة (endl) لإنزال المؤشر إلى السطر التالي
 ٨. خطوة رقم (١٣) هي طباعة حرف حرف
- ⊠ لاحظ لأسم تبين لك كل دالة وتقع تحت إي مكتبة



```

(Inactive C:\TCWIN45\BIN\NONAME01
HELLOW HUSSIEN AH-ROB.
WELCOME YOU PROGRAM
WHAT DA YOU LIKE OF ME
    
```

الكتابة في الملفات الثنائية:

الملفات الثنائية تستخدم لتخزين المصفوفات والمتغيرات والتراكيب. وللكتابة إلى داخل أي ملف توجد ثلاث خطوات

- خطوة الأولى نعرف كائن من نوع ملف
- خطوة ثانية نفتح الملف للكتابة
- الخطوة الثالثة نكتب إلى داخل الملف

١. في لغة (C++) نستخدم الدالة التالية للكتابة إلى داخل الملفات

كتابة الى داخل ملف في لغة C++

```
1. ofstream fout ;
2. fout.open("file path",iosstream family|ios::binary);
3. fout.write((char*)& data ,sizeof(data)) ;
```

- (iosstream family) : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معنية ونستخدم هنا الرمز (ios::binary) بمعنى أن الملف المدخل هو ثنائي
- وفي خطوة رقم (3) قمنا بإدخال البيانات إلى داخل الملف باستخدام الدالة (write) التي تأخذ (data) قد يكون متغير أو مصفوفة أو سجل وحجمها

٢. في لغة (c) نستخدم الدالة التالية للكتابة إلى داخل الملفات

كتابة الى داخل ملف في لغة c

```
1.FILE *fout;
2.fout=fopen("file path","symbol");
3.fwrite(& data ,sizeof(data),n, fout);
```

- (symbol) : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معية توضع بين علامتي تنصيص وهنا نستخدم الرمز (wb) لأننا نريد الكتابة بداخل الملف ثنائي.
- وفي خطوة رقم (٣) قمنا بإدخال البيانات إلى داخل الملف باستخدام الدالة (fwrite) التي تأخذ النص المراد كتابته (data) بداخل الملف وحجمه وعدد المواقع أو سجلات (n) و اسم الملف (fout)

مثال : تخزين مصفوفة أحادية حجمها (١٠) داخل ملف ؟.

C++	البرمجة بلغة	c	البرمجة بلغة
<pre>#include<iostream.h> #include <fstream.h> main() 1.{ int Array[80],i; 2.for(i=0;i<10;i++) 3.cin>> Array[i]; 4.ofstream fout; 5. fout.open("D:\\ar.bin",ios::binary); 6. fout .write((char *) & Array , sizeof(Array)); 7.fout.close(); }</pre>		<pre>#include<stdio.h> #include <fstream.h> main() 1.{ int Array[80],i; 2.for(i=0;i<10;i++) 3.scanf("%d",&Array[i]); 4.FILE * fout; 5. fout=fopen("D:\\ar.bin","wb"); 6.fwrite(& Array ,sizeof(Array),10, fout); 7. fclose(fout); }</pre>	

توضيح الخطوات :

- خطوة رقم (٥) فتحنا ملف ثنائي لكي نكتب في داخله
- خطوة رقم (٦) أدخلنا عناصر المصفوفة في الملف وحددنا حجم البيانات المدخلة هي عشرة



وتخزن بشكل التالي

✓ الملف لا يمكن فتحه بمتصفحات الملفات الاعتيادية لأن امتداده (*bin)

مثال: برنامج لخرن أسماء (٦) مستخدمين وكلمات مرورهم هي ملف ؟.

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> #include<stdio.h> #include <fstream.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{int i; 6.for(i=0;i<6;i++){ 7.cout<<"\nenter user name: "; 8. gets(userProtection[i]. username); 9.cout<<"enter password: "; 10. gets(userProtection[i]. password); 11.ofstream fout; 12. fout.open("D:\\up.bin",ios::binary); 13.for(i=0;i<6;i++) 14. fout.write((char*)& userProtection[i] ,sizeof(userProtection[i])); 15. fout.close();}</pre>	<pre>#include<stdio.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{int i; 6.for(i=0;i<6;i++){ 7. printf("\nenter user name: "); 8. gets(userProtection[i]. username); 9. printf("enter password: "); 10.gets(userProtection[i]. password);} 11.FILE * fout; 12. fout=fopen("D:\\up.bin","wb"); 13.for(i=0;i<6;i++) 14..fwrite(& userProtection[i],sizeof(userProtection[i]),6, fout); 15. fclose(fout); }</pre>

توضيح الخطوات:

```
enter user name: hussien
enter password: 1234
enter user name: alxs1aa
enter password: 342a
enter user name: waeel
enter password: 23454
enter user name: modor
enter password: 12345
enter user name: salam
enter password: 2345
enter user name: aeed
enter password: 12345
```

١. خطوة رقم (١٢) فتحنا ملف ثنائي للكتابة فيه

٢. خطوة رقم (٦ إلى ١٠) هي إدخال بيانات

٣. خطوة رقم (١٤) قمنا بإدخال كل سجل كامل

مرة واحدة إلى داخل الملف

☒ قارن هذا الحل بالحل السابق لنفس هذا المثال في حالة كتابة الملفات النصية

القراءة من الملفات الثنائية:

للقراءة من داخل إي ملف توجد ثلاث خطوات وهي.....!

- خطوة الأولى نعرف كائن من نوع ملف
- خطوة ثانية نفتح الملف للقراءة
- الخطوة الثالثة نقرأ من داخل الملف

١. في لغة (C++) نستخدم الدالة التالية للقراءة من داخل الملفات

قراءة من داخل ملف في لغة C++

1. `ifstream fin ;`
2. `fin.open("file path",iosstream family);`
3. `fin.read((char*)& array,sizeof(data));`

- **(iosstream family)** : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معنية ونستخدم هنا الرمز **(ios::binary)** بمعنى أن الملف المدخل هو ثنائي
- وفي خطوة رقم (3) قمنا بقراءة البيانات من الملف إلى داخل مصفوفة **(data)** وهذه ممكن أن تكون مصفوفة أو سجل أو متغير وان **sizeof(array)** معناه حجم البيانات التي سنقرأها من الملف

٣. في لغة (C) نستخدم الدالة التالية للقراءة من داخل الملفات

قراءة من داخل ملف في لغة C

1. `FILE *f fin ;`
2. `fin =fopen("file path","symbol");`
3. `fread(& data ,sizeof(data),n,fin)`

- **(symbol)** : هي رموز المستخدمة مثل تستخدم للدلالة على عمليات معية توضع بين علامتي تنصيص وهنا نستخدم الرمز **(br)** لأننا نريد القراءة من داخل الملف
- وفي خطوة رقم (٣) قمنا بقراءة البيانات من داخل الملف باستخدام الدالة **(fread)** إلى **(data)** ممكن أن تكون المصفوفة أو السجل أو المتغير وحجم الذي نريده (أن نقرأه أو عدد المواقع و اسم الملف **(fout)**

مثال : قراءة محتويات مصفوفة أحادية حجمها (١٠) من داخل ملف في المسار التالي ("D:\\ar.bin") ؟.

C++	البرمجة بلغة C
<pre>#include<iostream.h> #include <fstream.h> main() 1.{ int Array[10],i; 2.ifstream fout; 3. fout.open("D:\\ar.bin",ios::binary); 4. fout .read((char *) & Array , sizeof(Array)); 5.for(i=0;i<10;i++) 6.cout<< Array[i]<<"\t"; 7.fout.close(); }</pre>	<pre>#include<stdio.h> #include <fstream.h> main() 1.{ int Array[10],i; 2.FILE * fout; 3. fout=fopen("D:\\ar.bin","rb"); 4.fread(& Array ,sizeof(Array) ,1, fout); 5.for(i=0;i<10;i++) 6.printf("%d\t", Array[i]); 7. fclose(fout); }</pre>

توضيح الخطوات :

٣. خطوة رقم (3) فتحنا ملف ثنائي لكي نقرأ من داخله
٤. خطوة رقم (4) قمنا بقراءة عناصر بحجم المصفوفة (Array) من الملف ويخزنها في المصفوفة
٥. خطوة رقم (٥و٦) وطبعنا محتويات المصفوفة بشكل التالي



هذه العناصر سبق وان خزناها في الملف في مثال سابق

مثال: برنامج يطبع أسماء المستخدمين وكلمات مرورهم التي خزنت في ملف كثنائي .؟

البرمجة بلغة	c	البرمجة بلغة	c++
	<pre>#include<stdio.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{int i; 6.FILE * fout; 7. fout=fopen("D:\\up.bin","rb"); 8. fread(& userProtection ,sizeof(userProtection) ,1, fout); 9. fclose(fout); 10.for(i=0;i<6;i++){ 11. printf("\n User ID :\t"); 12. printf("%s", userProtection [i].username); 13. printf("user Password: "); 14. printf("%s", userProtection [i]. password);} }</pre>		<pre>#include<iostream.h> #include<stdio.h> #include <fstream.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{int i; 6 ifstream fout; 7. fout.open("D:\\up.bin",ios::binary); 8. fout .read((char *) & userProtection , sizeof(userProtection)); 9. fout.close(); 10.for(i=0;i<6;i++){ 11.cout << "\nUser ID :\t"; 12.cout << userProtection [i]. username; 13.cout << "\tuser Password: "; 14.cout << userProtection [i]. password;}}</pre>

توضيح الخطوات:

١. خطوة رقم (٧) قرئنا من داخل الملف بيانات بحجم السجل وهو (٦) وكل واحد في السجل له اسم مستخدم وكلمة مرور سيقوم المترجم تلقائيا بإسناد كل اسم مستخدم وكلمة مروره مخزن داخل الملف الى موقع من مواقع السجل

٢. خطوة رقم (١٠ إلى ١٤) قمنا بطباعة محتويات السجل التي ملئناها من الملف

وتكون شكل شاشة التنفيذ هكذا حسب ما هو مخزن في الملف

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
User ID :      hussien user Password: 1234
User ID :      alxs1aa user Password: 342a
User ID :      waeel  user Password: 23454
User ID :      modor  user Password: 12345
User ID :      salam  user Password: 2345
User ID :      aeed   user Password: 12345
```

✓ لو أردنا أن يطبع محتويات سجل لمستخدمين اثنين فقط نغير خطوة رقم (٨) إلى

البرمجة بلغة	c	البرمجة بلغة
C++	8. fread(& userProtection ,sizeof(userProtection[0]) ,2, fout);	8. fout .read((char *) & userProtection , 2*sizeof(userProtection[0]));

الذي فعلناه هو أخذنا حجم سجل واحد وهو السجل الأول (**userProtection[0]**) وضربناه في اثنان لكي نحصل على حجم سجلين من داخل الملف فقط لذلك سيطبع محتويات سجلين فقط كما في شاشة التنفيذ

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
User ID :hussien          user Password: 1234
User ID :alxs1aa         user Password: 342a
User ID :                user Password:
User ID :                user Password:
User ID :                user Password:
User ID :                user Password:
```

السجلات البقية تبقى فارغة

ونستطيع تحديد عدد الملفات من الرقم المضروب بالحجم فلو أردنا ثلاث ملفات نضرب في ثلاثة لكن هذه الطريقة تطبع بعدد السجلات المطلوب لكن من الأول ثم الثاني وبالتسلسل ماذا لو أردنا طباعة فقط السجل الثالث أو فقط السجل الرابع هذا ما تقدمه دالة (**seek**) التي تمكنك من تحريك المؤشر داخل الملف لأن المؤشر بطبيعته يبدأ من أول موقع لذلك سيطبع السجلات بالتسلسل إما هذه الدالة تمكنك من تحريك مكان المؤشر وتكون بالشكل التالي

١. في لغة (C++) نستخدم الدالة التالية لتحريك المؤشر داخل الملف

تحريك داخل ملف في لغة C++

```
fin.seekg(offset, iostream family );
```

(fin) : هو اسم الملف داخل البرنامج

(offset) : مكان بدء القراءة إي من إي بايت تبدأ القراءة أو الكتابة

(iostream family) : إذا لم نضع إي رمز معناه سيبدأ (offset) من بداية الملف وإذا وضعنا (ios::end)

معناه سيبدأ (offset) من نهاية الملف

جعل المؤشر يبدأ من نهاية الملف بأزاحة مقدارها ٥٠٠

```
fin.seekg(-50,ios::end);
```

٢. في لغة (c) نستخدم الدالة التالية لتحريك المؤشر داخل الملف

كتابة الى داخل ملف في لغة ++c

```
fseek(fin,offset,whence);
```

(fin) : هو اسم الملف داخل البرنامج

(offset): مكان بدء القراءة إي من إي بايت تبدأ القراءة أو الكتابة

(whence) : إذا وضعنا (0)معناه سيبدأ (offset) من بداية الملف وإذا وضعنا (١) معناه سيبدأ (offset) من الموقع الحالي وإذا وضعنا (2) معناه سيبدأ (offset) من نهاية الملف

جعل المؤشر يبدأ من نهاية الملف بأزاحة مقدارها ٥٠ -

```
int fseek(fin,-50,2);
```

✓ في مثال السجلات لو أردنا طباعة السجل الثالث

تحليل: نضرب حجم واحد من السجلات في اثنان ونجعلها قيمة (offset) حتى يؤشر في نهاية السجل الثاني وعندما يبدأ القراءة بيدوها من السجل الثالث. ونجعل التحرك من البداية وفي دالة القراءة نجعله يقرأ سجل واحد وهو الثالث نغير الخطوة رقم (٨) في المثال إلى خطوتين بالشكل التالي لكي يطبع سجل الثالث فقط

C++	البرمجة بلغة	c	البرمجة بلغة
<pre>fout.seekg(2*sizeof(userProtection[0])); fout.read((char *)&userProtection, sizeof(userProtection[0]));</pre>		<pre>fseek(fout,2*sizeof(userProtection[0]), 0); fread(&userProtection,sizeof(userProtection[0]),1, fout);</pre>	
	نجعله يقرأ سجل		ونجعل التحرك من البداية
		نضرب حجم واحد من السجلات في اثنان ونجعلها قيمة (offset) حتى يؤشر في نهاية الملف الثاني	

✓ في مثال السجلات لو أردنا طباعة السجل الرابع

تحليل: نضرب حجم واحد من السجلات في ثلاثة ونجعلها قيمة (offset) حتى يؤشر في نهاية السجل الثالث وعندما يبدأ القراءة بيدوها من السجل الرابع ونجعل التحرك من البداية وفي دالة القراءة نجعله يقرأ سجل واحد نغير الخطوة رقم (٨) في المثال إلى خطوتين بالشكل التالي لكي يطبع سجل الثالث فقط

C++	البرمجة بلغة	c	البرمجة بلغة
<pre>fout.seekg(3*sizeof(userProtection[0])); fout.read((char *)&userProtection, sizeof(userProtection[0]));</pre>		<pre>fseek(fout,3*sizeof(userProtection[0]), 0); fread(&userProtection,sizeof(userProtection[0]),1, fout);</pre>	

وهذا كود البرنامج كامل لطباعة محتويات سجل الرابع فقط

البرمجة بلغة C++	البرمجة بلغة C
<pre>#include<iostream.h> #include<stdio.h> #include <fstream.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{ifstream fout; 6. fout.open("D:\\up.bin",ios::binary); 7. fout.seekg(3*sizeof(userProtection[0])); 8.fout.read((char *) & userProtection , sizeof(userProtection[0])); 9. fout.close(); 10.cout<< "User ID :"; 11.cout << userProtection [0]. username; 12.cout<< "\\tuser Password: "; 13.cout << userProtection [0]. password;}</pre>	<pre>#include<stdio.h> 1.struct password_User 2.{char username[20]; 3. char password[20]; }userProtection[6]; 4.main() 5.{FILE * fout; 6. fout=fopen("D:\\up.bin","rb"); 7.fseek(fout,3*sizeof(userProtection[0]), 0); 8.fread(& userProtection ,sizeof(userProtection[0]) ,1, fout); 9. fclose(fout); 10. printf("\n User ID :\\t"); 11. printf("%s", userProtection [0].username); 12. printf("user Password: "); 13. printf("%s", userProtection [0]. password); }</pre>

بما انه سجل واحد فليس بحاجة إلى عداد يعد للسنة فقط نطبع الموقع الأول الذي فيه المستخدم هكذا

✓ في مثال السجلات لو أردنا طباعة السجل الرابع والخامس والسادس تحليل: نضرب حجم واحد من السجلات في ثلاثة ونجعلها قيمة (offset) حتى يؤشر في نهاية السجل الثالث ونجعل التحرك من البداية وفي دالة القراءة نجعله يقرأ ثلاث سجلات لكي يقرأ السجل الرابع والخامس والسادس نغير الخطوة رقم (٨) في المثال إلى خطوتين بالشكل التالي لكي يطبع سجل الثالث فقط

البرمجة بلغة C++	البرمجة بلغة C
<pre>fout.seekg(3*sizeof(userProtection[0])); fout.read((char *) & userProtection, 3*sizeof(userProtection[0]));</pre>	<pre>fseek(fout,3*sizeof(userProtection[0]), 0); fread(& userProtection ,sizeof(userProtection[0]) ,3, fout);</pre>

ونغير شرط توقف العداد في خطوة رقم (١٠) إلى اصفر من ثلاثة لأننا سنطبع ثلاث سجلات فقط

النهائية