
FOR DOS
TURBO C++

Index

Introduction

Turbo C++ is one of important programming language in the world , C++ language having many versions which it is apply with the programs and applications of the computer system , and the most important version between them is the version that microsoft company build the group of microsoft office by using it , this version is called (visual C++) , we here will exploring the first version of C++ that is called (Turbo C++ For Dos) this type of this language will be your way to be a programmer in the future so in this book will explain the basics of (turbo C++ for Dos) .

* first part own the agency functions this functions make turbo C++ very easier than the other programming language .
* second part own the loop repetition and the instruction of it , after that we exploring the one dimension matrixes and how we can edit it and how we can process it in turbo C++ .
* Third part own some lectures of the college for year .

here I'll leave you to discover all the subjects and I hope this book will back to you with a better primary information about this huge programmer language and make you to take a step foreword .

X-PC Seven

رموز لغة C++

<< The Character Of Turbo C++ >>

الرمز	وظيفتـــــــه
+	لعملية الجمع
-	لعملية الطرح
+ +	لعملية الزيادة بمقدار يحدده المبرمج
- -	لعملية النقصان بمقدار يحدده المبرمج
/	لعملية القسمة
%	لايجاد باقي القسمة
*	لعملية الضرب
!=	لا يساوي
==	يساوي
||	OR
&&	AND
!	NOT
[ ]	Matrix
<	اصغر
>	اكبر
<=	اصغر او يساوي
>=	اكبر او يساوي

جدول رقم (1)

الاعلان عن المتغيرات
Variables Advert

ان كل لغات البرمجة تتعامل مع ذاكرة الحاسب المؤقتة من ناحية خزن و استرجاع البيانات من الذاكرة و ان لغة البرمجة هذه تحتاج الى وسيلة تربطها بتلك الذاكرة و هذه الوسيلة تدعى بالمترجم (Compiler) و هذا المترجم يفسر الايعازات الخاصة بلغة البرمجة الى الحاسب لكي يحصل على نتائج و يعرضها على الشاشة و لكتابة اي برنامج لابد من وجود مدخلات او بيانات و هذه البيانات تدخل الى البرنامج عن طريق لوحة المفاتيح كأن تكون هذه البيانات عبارة عن معادلة رياضية تتضمن ضرب عددين مثلا و هذين العددين يتم اخالهما من خلال لوحة المفاتيح , لذلك يجب حجز مكان لكل من هذين العددين في الذاكرة و لاننسى المكان الذي سنحجزه من اجل ان توضع به نتيجة عملية الضرب , و هذا المكان الذي سوف يحجز في الذاكرة يجب وصفه برمجياً بـ (int , float , char , string) و بحسب نوع المتغير سواء كانت (اعداداً صحيحة او اعداداً حقيقية او رموز او سلاسل حرفية) .

و لو اردنا ايجاد ناتج المعادلة الرياضية السابقة و التي تتضمن ضرب عددين سنحتاج الى متغيرات يتم وصفها على انها (int) اي اعداداً صحيحة لان الاعداد التي سوف ندخلها و ناتج المعادلة الرياضية عبارة عن اعداد صحيحة و يمكن ايظاً وصف المتغير (float) بمعنى عدد حقيقي اذا كانت القيم المدخلة للبرنامج عبارة عن كسور لاحظ الجدول رقم (2)

نوع الوصف	اختصـــاصـــه
Integer int	هذا النوع من الوصف يكون خاص بالاعداد الصحيحة الموجبة و السالبة و الصفر الخالية من الفاصلة العُشرية فالـ (الصفر) عدد صحيح و (7) عدد صحيح و الـ (100) كذلك , اما الـ (2.3 و 7.2 ..... Ex) فهي اعداد غير صحيحة و انما حقيقية .
Real Float	هذا النوع من الوصف يكون خاص بالاعداد المسماة بالاعداد الحقيقية و هي التي تضم الكسور مثل الـ (0.5 و 7.2345 و 6.5678 ..... Ex)
Character Char	هذا النوع من الوصف يكون خاص بالرموز و الحروف و التي تكون موجودة على لوحة المفاتيح و تشمل الحروف الابجدية سواء كانت حروفاً كبيرة (A B C D .... Ex) او صغيرة (a b c d ..... Ex) و الرموز الاخرى مثل الـ (@ . # . ٪ . & . ؟ . ! . *) فأنها ايظاً يمكن التعبير عنها في برامجنا و ذلك بأستخدام الوصف (Char) .

جدول رقم (2)

طريقة الوصف في البرامج

ان طريقة وصف المتغيرات في لغة السي ++ سهلة جداً قياساً الى اللغات الاخرى حيث يتم الوصف بعد الـ (Begin , {) لاحظ الشكل المجاور , و لقد احتوى الشكل على العديد من الاوصاف و لكن نحن في برامجنا الاعتيادية نصف فقط ما هو مطلوب او قد يحتاج برنامجك على جميعها او بعضها فلا توجد مشكلة في ذلك .

```
#include<iostream.h>
main( )
{
int      i , j ,  a[10] ;
char     i , j ,  a[10] ;
float    i , j ,  a[10] ;
.
.
.
}
```

الـــدوال المكتبيـــة

الدالة المكتبية :- عبارة عن ايعاز يكتب في بداية كل برنامج لكي يفسر ايعازات اخرى سوف تكتب لاحقا في البرنامج و الدالة المكتبية هي احد اسرار سهولة و بساطة لغة البرمجة (السي ++) .

ان لكل لغة برمجة مترجم خـــاص بها يساعد الحاسوب على فهم الايعازات التي تقوم بكتابتها داخل البرنامج و هذا المترجـــم يدعى بـــ (Compiler) و لغة السي ++ لها Compiler خاص بها , و لقد قسم مؤسسي اللغة الـــ Compiler الى العديد من الدوال المكتبية و كل دالة تختص بتفسير ايعازاتها الخاصة , و اليك البعض من هذه الدوال :-

```
1- #include<iostream.h>
2- #include<math.h>
3- #include<conio.h>
4- #include<stdio.h>
```

#include<iostream.h>

ضمن لي .. يحوي على < جدول العناوين الرأسية للادخال والاخراج مثلا >

ان رمز المربع او (#) الذي يسبق الدالة المكتبية يخبر المترجم (compiler) ان الايعاز الذي كُتب هو عبارة عن دالة مكتبية .

include : بمعنى ضمن او احتوي على ما بين علامة الاكبر و الاصغر التي تلي كلمة include .

iostream.h : الـــ (io) بمعنى (input \ output) اي ادخال واخراج و الـــ (stream) جدول او المكان الرئيسي المتواجدة به الـــ (io) اما الـــ (h) فتعني العنوان الرأسي headers .

اما بقية الدوال المكتبية فهي نفس الدالة السابقة و لكن من المؤكد ان عملية التضمين ستختلف حسب اسم الدالة المتواجد بين (< >) و الذي سيحدد عمل الايعازات التي ستكتب لاحقا في البرنامج

#include<iostream.h>

. اولآ - دالة القراءة و الطباعة

هذه الدالة هي واحدة من اهم الدوال المكتبية , اذ لا يمكن ان يكون البرنامج لا يحوي على مدخلات او مخرجات و نتائج لذلك وجود هذه الدالة في كل البرامج يكون اكيداً , و الايعازات التي تفسرها هذه الدالة للحاسب هي :-

Cin >> هو ايعاز يكتب في البرنامج لادخال قيم معينة من خلال لوحة المفاتيح وهذه القيم تم وصف متغيراتها مسبقا .

Cout<< هو ايعاز يكتب في البرنامج لاظهار النتائج او المعلومات التي تمت معالجتها في البرنامج .

* مثال (1)

* Example (1)

```
#include<iostream.h>
main( )
{
int x , y , z ;
Cin>> x >> y ;
z = x + y ;
Cout<< z ;
}
```

```
#include<iostream.h>
main( )
{
int x , y , z ;
Cin>> x ;
Cin>> y ;
z = x + y ;
Cout<< z ;
}
```

البرنامج السابق كانت مهمته جمع عددين يتم ادخالهما من خلال لوحة المفاتيح لذلك هذا البرنامج يحتاج الى دالة تفسر ايعاز ادخال البيانات الى البرنامج و الدالة هي دالة (القراءة و الطباعة) بعد ذلك شاهدنا الايعاز (main) و هو ايعاز بداية البرنامج الرئيسي و التعبير عن بداية البرنامج بلغة السي ++ اسهل مما هو في لغة البرمجة (Turbo Bascal) و ذلك بأستخدام الرمز ({) بعد رمز البداية السابق يأتي دور الوصف للمتغيرات التي سوف نستخدمها في البرمجة .

ان القيم التي سوف ندخلها لابد ان يكون لها مكاناً محجوزاً في ذاكرة الحاسب و هذه الاماكن المحجوزة يجب ان يكون لها عنوان معين لتمكين المترجم الـــ (Compiler) من الوصول اليها بدون مشاكل و العناوين التي استخدمناها في البرنامج السابق هي (X , Y , Z) و هذه القيم تم وصفها استناداً الى انها اعداداً من النوع الصحيح (Int)

الان يأتي دور ادخال قيم عددية من لوحة المفاتيح و وضعها في الاماكن التي تم حجزها في الذاكرة و لأتمام عملية الادخال سنحتاج الى ايعاز الـــ (Cin>>) لاحظ المثال رقم (1) **(عند ادخال اكثر من قيمة للبرنامج نستخدم الرمز >> لادخال المتغير التالي و يمكننا الاستغناء عن هذه الطريقة اي يمكن ادخال متغيرين للبرنامج و ذلك بتكرار ايعاز الادخال *Cin*)** لاحظ المثال رقم (1) و كيفية اتمام عملية الادخال .

بعد وصف المتغيرات و ادخال القيم لابد من وجود ايعاز آخر مكمل للبرنامج لكي يقوم بما هو مطلوب منه (عملية الجمع) و الايعاز في هذه الحالة يقع ترتيبه على عاتق المبرمج ففي البرنامج السابق تمت البرمجة على ان المتغير (X) هو احد الاعداد المدخلة و المتغير (Y) هو العدد الثاني , اما المتغير (Z) فهو سيمثل ناتج العملية الرياضية اي ان الـــ (Z) هو القيمـــة التي ستظهر على الشاشة و الايعاز الخـــاص بعمليـــة الجمع هو (z=x+y) .

الان اصبحت لدينا قيمة في المتغير (Z) ألا و هي ناتج عملية الجمع و لأخراج هذا الناتج و عرضه على الشاشة لابد من وجود ايعاز يختص بأظهار البيانات و الايعاز هو (Cout<<) هذا الايعاز سيظهر ناتج عملية الجمع المخزونة في المتغير (Z) و يعرضه على الشاشة .

الان جاء دور نهاية البرنامج و للتعبير عن النهاية في لغة البرمجة سي++ بالرمز (}) .

* Note :- الان حاول كتابة البرنامج و قم بتنفيذه و أدخل اعداد معينة و انظر الى النتيجة

* العبارة التوضيحية

The Explaining Phrase

العبارة التوضيحية : عبارة عن جملة تظهر عند تنفيذ البرنامج تبين للمستخدم وظيفة البرنامج و طريقة عمله .
فالعبارة التوضيحية عبارة عن وظيفة ثانوية مهمة لا تؤثر على مُجريات عمل البرنامج اي لا يؤثر وجودها و عدم وجودها فهي وضِعت لتفسير شيئ مبهم للمستخدم كأن يكون ذلك الشيئ هو عمل البرنامج فمثلآ لو كان البرنامج يختص بضرب عددين فأن المبـــرمـــج سيـضع العبارة (Insert Two Number To Multiply Them) اي بمعنى ادخل عددين ليتم ضربهما , هذه العبارة بَيَنت عمل البرنامج و بشكل دقيق فأذا ترك المبرمج برنامجه هذا لفترة و عاد اليه و قام بتنفيذه سيرى على الشاشة العبارة التي قام بكتابتها مسبقآ و بالتالي ذَكرته بما قام من برمجة لاحظ المثال رقم (2) .

* مثال (2)

* Example (2)

```
#include<iostream.h>
main( )
{
Cout<<" Welcome In Turbo C++" ;
}
```

* مثال (3)

* Example (3)

```
#include<iostream.h>
main( )
{
int  i ;
Cout<<" Insert A Number To Print It " ;
Cin>> i ;
Cout<< i ;
}
```

في المثال السابق (3) كانت مهمة البرنامج ادخال رقم و طبعه على الشاشة و لقد بينت العبارة التوضيحية ذلك لاحظ مكان كتابة ايعاز العبارة قبل عملية الادخال للمتغير و ذلك لكي تظهر العبارة و من ثم عملية الادخال اي انه نفهم عمل البرنامج من خلال العبارة التوضيحية و من ثم ادخال المتغيرات .

*** Note :-** في بعض اصدارات برامج السي ++ عند تنفيذك المثال رقم (2) قد تظهر لك شاشة التنفيذ و تختفي بصورة سريعة عندئذ لن تستطع رؤية نتيجة البرنامج لذلك قم بكتابة ايعاز توقيف الشاشة (" " << Cin) بعد الوصف و الذي سيُمَكن الشاشة السوداء من البقاء و يساعدك على مشاهدة النتائج او قم بالضغط على (Ctrl F5) بعد التنفيذ .

#include<math.h>

ثانياً - الدالة الرياضية

و هي الدالة التي تفسر الايعازات الرياضية التي نقوم بكتابتها في البرنامج للحاسب لاحظ الجدول رقم (3) و الذي احتوى على البعض من الايعازات الرياضية .
و الفائدة من هذه الدالة هي لايجاد ناتج معادلة رياضية بحيث هذه المعادلة تكتب بشكل برمجي مناسب يفهمه الـ (Compiler) لايجاد المطلوب منها .

برمجيـــــــــــآ	رياضياً
S = Sin (X) ;	S = Sin X
S = Cos (X) ;	S = Cos X
S = Tan (X) ;	S = Tan X
S = Pow (X , Y) ;	$S = X^Y + X^Y + X^Y + X^Y$

جدول رقم (3)

Power Instruction

ايعاز الرفع الى اس معين

اذا كانت لديك مثل المعادلة المبينة في الجدول رقم (3) حيث نرى المتغير (X) مرفوعة قيمة معينة كأن تكون المتغير (Y) فالايعاز البرمجي لها و المبين في الجدول هو الذي سندرجه في برامجنا لايجاد قيمة الـ (S) .
لاحظ الايعاز البرمجي للـ (Power) و كيفية ترتيب الاس و الاساس للمعادلة الرياضية بالصورة البرمجية .. لاحظ المثال رقم (4) و الذي سيقوم بحل المعادلة مفترضاً قيمة الاسس (Y) تساوي (2) و لو قمت بتنفيذ البرنامج و ادخلت قيمة الـ (X) فأذا كانت تساوي (2) فأن قيمة الـ (S) ستصبح لديك تساوي (16)

* مثال (4)

* Example (4)

```
#include<iostream.h>
#include<math.h>
main( )
{
int  s , x ;
Cin>> x ;
s = pow( x , 2 ) + pow( x , 2 ) + pow( x , 2 ) + pow( x , 2 ) ;
Cout<< s ;
}
```

في هذا البرنامج استخدمنا دالة مكتبية جديدة و عمل هذه الدالة ترجمة ايعاز (pow) للحاسب ولقد قمت بوصف الـــ (X) كمتغير سيحل محل الاساس و العدد (2) هو الاس للاساس (X) و عند تنفيذ هذا البرنامج سيطلب منك ادخال قيمة الـــ (X) للمعادلة بعد ذلك يأتي دور الايعاز (pow) و الذي سيقوم بتربيع العدد الذي قمت بأدخاله بعدها يجمع ناتج التربيع الاول مع ناتج التربيع الثاني وهكذا الى ان تنتهي المعادلة و ناتج هذه العملية سيخزن في المتغير (S) و المتغير (S) ما هو الا موقع فارغ لوضع النتيجة فيه بعد ذلك يأتي دور طباعة النتيجة على الشاشة و ذلك بأستخدام الايعاز (Cout<<) الذي سيقوم بطبع الناتج المخزون في المتغير (S) على الشاشة لاحظ المثال رقم (5) و الذي ستكون فيه الاسس مختلفة .

* مثال (5)

* Example (5)

```
#include<iostream.h>
#include<math.h>
main( )
{
int s , x ;
Cin>> x ;
s = pow( x , 3 ) + pow( x , 4 ) * pow( x , 2 ) + pow( x , 6 ) ;
Cout<< s ;
}
```

الان ماذا لو يقوم المبرمج بتحديد الاس للمعادلة ! اي هل من الممكن ان نقوم بتنفيذ البرنامج و ادخال اس و اساس للمعادلة ؟؟ .
الجواب هو البرنامج التالي :-

* مثال (6)

* Example (6)

```
#include<iostream.h>
#include<math.h>
main( )
{
int s , x , i  ;
Cout<<" Insert The Asas "  ;
Cin>> x  ;
Cout<<" Insert The As "  ;
Cin>> i ;
s = pow( x , i ) + pow( x , i ) + pow( x , i ) + pow( x , i ) ;
Cout<< s ;
}
```

كالعادة وفي كل برنامج نكتب دالة القراءة و الطباعة و الدالة الجديدة هي الدالة الرياضية و التي هي احد اسس البرنامج بعدها يأتي وصف المتغيرات التي سنستخدمها في البرمجة و الجديد في هذا البرنامج هو العبارة التوضيحية و التي ستوضح للمستخدمين لبرنامجك هذا متى يدخل الاساس و الاس للمعادلة حيث ان الـــ (X) قام بدور الاساس و الـــ (i) قام بدور الاس و عند تنفيذ هذا البرنامج ستظهر على الشاشة العبارة التوضيحية (Insert The Asas) اي الـــ (X) ادخل قيمة معينة و اضغط (Enter) ستظهر العبارة (Insert The As) اي الـــ (i) بعد ادخال القيم اضغط (Enter) او (Ctrl F5) و لاحظ النتائج .

* Note :- حاول كتابة البرنامجين اعلاه و قم بأدخال قيم معينة و لاحظ النتائج .

ثالثا - دالة مسح الشاشة

#include<conio.h>

ان العمل على تنفيذ برنامج معين اكثر من مرة و ادخال القيم اليه و الحصول على نتائج كل هذا سيظهر على الشاشة السوداء (شاشة التنفيذ) و لجعل الشاشة فارغة خالية من اي مدخلات او نتائج مسبقة عند التنفيذ نكتب الدالة المكتبية (conio.h) و التي تفسر او تترجم الايعاز (Clrscr ( )) و الذي سينظف الشاشة عند كل تنفيذ لاحظ المثال رقم (7) .

* مثال (7)

* Example (7)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
Cout << '' Turbo C++ Clear Screen Instruction '' ;
}
```

في المثال السابق استخدمنا الدالة المكتبية (conio.h) و التي ترجمة الايعاز (Clrscr ( )) و البرنامج السابق كانت مهمته عرض العبارة (Turbo C++ Clear Screen Instruction) و عند تنفيذ البرنامج ستظهر العبارة على الشاشة السوداء و لو نفذنا البرنامج مرة اخرى لن نرى عبارتين على الشاشة بل عبارة واحدة لان الايعاز الذي قمنا بكتابته قام بمسح ما موجود على الشاشة اي مسح النتائج الخاصة بالتنفيذ الاول .. و هكذا يعمل الايعاز عند كل تنفيذ جديد , و فائدة هذا الايعاز هي لمنع تراكم المدخلات و النتائج عند كل تنفيذ لذلك يلجئ المبرمج الى المسح لاظهار النتائج واضحة للمستخدم .

* مثال (8)

* Example (8)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int x , y , z ;
Cin>> x >> y ;
z = x * y ;
Cout<< z ;
}
```

البرنامج رقم (8) و ظيفته ضرب عددين ولقد اضفنا الدالة الخاصة بالمسح اليه بحيث عند تنفيذ البرنامج و ادخال قيم معينة اليه بالتأكيد ستظهر النتائج الخاصة بالتنفيذ الاول للبرنامج ولو قمنا بتنفيذه مرة اخرى فسوف لن ترى ما قمت بأدخاله مسبقا اي كأن البرنامج ينفذ لأول مرة لان هذه الدالة عملت على تنظيف الشاشة السوداء من كل ما هو خاص بالتنفيذ الاول .

* Note :- حاول كتابة هذه الدالة في جميع برامجك الخاصة و ليكن القصد ترتيباً للبرنامج و حرصاً على ظهور النتائج واضحة للمستخدم عند التنفيذ .

الان سنعيد كتابة المثال رقم (4) و لكن بأستخدام جميع الدوال و الايعازات السابقة و لقد كان عمل المثال رقم (4) هو ايجاد ناتج المعادلة الرياضية ادناه . في المثال رقم (9) لاحظ ترتيب الايعازات .

$$S = X^{Y} + X^{Y} + X^{Y} + X^{Y}$$

* Example (9)

* مثال (9)

```
#include<iostream.h>
#include<math.h>
#include<coino.h>
main( )
{
Clrscr ( ) ;
int  X , S ;
Cout<<" Enter The One Value " ;
Cin>> X ;
S = Pow( X , 2 ) + Pow( X , 2 ) + Pow( X , 2 ) + Pow( X , 2 ) ;
Cout<< " S " << " = " << S  ;
}
```

* في المثال السابق لو كان العدد المُدخل للبرنامج هو (2) فستكون النتيجة كما في الشكل ادناه ...

```
S = 16
```

#include<stdio.h>

رابعآ - دالة فتح الملفات و التعديل عليها

ان الملفات المخزونة في الذاكرة الدائمية للحاسب (HDD) يمكن ادراجها في برامجنا و التي تكون ذات الامتداد (doc . txt . exe) و غيرها و هذه الدالة المكتبية لها اهمية كبيرة في برمجة البرامج التنفيذية و التي تكون اساس للبعض من البرامج التي نستخدمها و لهذه الدالة العديد من الايعازات و التي تساعد على فتح و غلق و طبع البيانات من خلال البرنامج و من هذه الايعازات :-

الايعاز	وظيفتـــــــه
fopen	تفتح ملف موجود على (HDD) .
fclose	تغلق الملف الذي قمت بفتحه مسبقا او تغلق ملف خارج البرنامج و ذلك بكتابة المسار الخاص به .
putc	تطبع لك رمزآ داخل الملف و هي مثل الـ (Char) .
getc	تدخل لك رمزآ الى داخل الملف و هي مثل الـ (Char) .
fseek	و تستخدم للبحث عن نص داخل الملف الذي قمت بفتحه داخل البرنامج .
fprintf	و تستخدم للكتابة داخل الملف .

جدول رقم (4)

* مثال (--)

* Example (--)

```
#include<iostream.h>
#include<stdio.h>
main ( )
{
Char   x [ 10 ] ;
int  i  ;
FILE * q  ;
q = fopen ( " w . dat " , " w " )  ;
for ( i = 0 ; i <=10 ; i ++ )
{
Cin >> x  ;
fprintf ( q , x ) ;
}
fclose ( q )  ;
}
```

الاستاذ : عبد الاله

... حلقات التكرار ...

... The Loop Repetition ...

1st - The Loop Repetition - For

اولا - حلقة التكرار (For)

حلقة التكرار :- توفر لغة السي ++ كسائر لغات البرمجة عدداً من اساليب التكرار و من هذه الاساليب الاسلوب المتمثل بأستعمال (For) و هي ذات قوة و مرونة اكثر من غيرها من حلقات التكرار .
و يستخدم الـــ (Loop) لتكرار عدد من البيانات لتسهيل قراءتها عند الادخال و تسهيل طباعتها بعد المعالـجة و للتكرار عدة انواع من الايعازات في لغة السي ++ و الـــ (For) هي احد هذه الايعازات و ابسطها .
السؤال الان هو : كيف يمكن كتابة ايعاز الـــ (For) ؟ و ماذا لو اردت طباعة عدد من الارقام بـدون الحـاجـة الـى ايعـــاز الـ (Cin) ؟ لاحـــظ الامثلـــة ادنـــاه .

```
for ( Var = 1st Value ; Var <= Last Value ; Increment Magnitude )  Ex \  for ( i=0 ; i <=10 ; i ++ )
```

```
for ( Var = 1st Value ; Var >= Last Value ; Increment Magnitude )  Ex \  for ( i=10 ; i >=0 ; i - - )
```

* Note :- القواعد اعلاه ثابتة يمكن التغيير فيها لكن حسب نوعية البرنامج و طريقة عمله و هذا التغيير يعتمد على اشارة الاكبر و الاصغر و بداية الحلقة و نهاية الحلقة (شرط التوقف) ... لاحظ ايعاز التكرار فأنه لايحوي على فارزة منقوطة في النهاية .

1 - Var = المتغير
2 - first value = القيمة الابتدائية - بداية التكرار
3 - Last value = القيمة النهائية - القيمة التي تتوقف عندها حلقة التكرار
4 - Increment Magnitude = مقدار الزيادة

- 1 - المتغير : عبارة عن وصف يستخدم في الحلقة لتحديد بداية الحلقة و نهايتها و خطوات سيرها لتكرار قيم محددة من قبل المبرمج .
- 2 - القيمة الابتدائية : و هي بداية التكرار , و لو اردنا طباعة ارقام من صفر الى 7 نحدد القيمة الابتدائية بالصفر .
- 3 - القيمة النهائية : و هي القيمة التي تتوقف عندها حلقة التكرار .
- 4 - مقدار الزيادة : و هو الايعاز المستخدم لتحديد خطوات سير حلقة التكرار , لاحظ المثال رقم (10)

* مثال (10)

* Example (10)

```
#include<iostream.h>
#include<conio.h>
main( )
{
int  i  ;
Clrscr ( ) ;
for ( i = 0 ; i <= 10 ; i ++ )
Cout<< "  " << i ;
}
```

البرنامج السابق و ظيفته طبع الاعداد من 0 الى 9 بدون اللجوء الى ايعاز الـــ (Cin) لادخال الاعداد , لاحظ البداية كانت من الصفر و شرط توقف حلقة التكرار هو الوصول الى الرقم العاشر و مقدار الزيادة هو واحد (i++) و ناتج هذا البرنامج هو كما في الشكل ادناه .

```
0 1 2 3 4 5 6 7 8 9
```

في المثال السابق (10) كان ترتيب ايعاز الـــ (For) للارقام تصاعدياً (لاحظ طريقة كتابة الايعاز التصاعدي في المثال رقم 10) الان و في المثال التالي سيكون البرنامج لطباعة الارقام تنازلياً

* مثال (11)

* Example (11)

```
#include<iostream.h>
#include<conio.h>
main( )
{
int  i  ;
Clrscr ( ) ;
for ( i = 10 ; i >= 0 ; i -- )
Cout<< "  " << i ;
}
```

البرنامج الســـابق و ظيفتـــه طـــبع الاعداد من 10 الى 0 لاحظ شرط توقف حلقة التكرار هو الوصول الى الرقم صفر و مقدار النقصان هو واحد (--i) و ناتج هذا البرنامج هو كما في الشكل ادناه .

```
10 9 8 7 6 5 4 3 2 1 0
```

في المثال السابق لو كان شكل ايعاز الـــ (For) كما في الشكل ادناه فأنه سيعطي نفس الناتج

```
for ( i = 10 ; i >= 0 ; i - = 1 )
```

الان ماذا لو اردنا ان يكون مقدار الزيادة لسير حلقة التكرار اكثر من واحد ؟ لاحظ الايعازات التالية لحلقة التكرار (For) :-

هذا الايعاز يطبع الاعداد الزوجية ما بين الـ 0 و الـــ 10	for (i = 0 ; i <= 10 ; i+=2)
هذا الايعاز يطبع الاعداد الفردية ما بين الـــ 1 و الـــ 10	for (i = 1 ; i <= 10 ; i+=2)
هذا الايعاز يطبع الاعداد الفردية ما بين الـــ 9 و الـــ 1 تنازلياً .	for (i = 9 ; i >= 0 ; i-=2)

* Notes :

1- امعن النظر علـــى القيمة الابتدائية للحـــلقة اذا كانت تبـــدأ بعدد زوجي و مقدار الزيادة 2 فالناتج هو الاعداد الزوجية و اذا كانت القيمة الابتدائية عدداً فردياً و مقدار الزيادة 2 فالناتج اعداداً فردية .

2- من الواضح ان القيمة الابتدائية و مقدار الزيادة هما من يحددان الاعداد التي ستطبع و القيمة النهائية ما هي الا شرط لتوقف عمل حلقة التكرار .

3- حاول كتابة الايعازات السابقة في البرنامج اعلاه و لاحظ النتائج .

4- لاحظ الايعاز الاخير (التنازلي) القيمة الابتدائية هي 9 اي انه اي حلقة تكرار تنازلي فردي تكون القيمة الابتدائية للحلقة هي عدداً فردياً , و القيمة النهائية (شرط التوقف للحلقة) هي الصفر او 1 فأنه يعطي نفس الناتج لاحظ المثال رقم (12) .

* مثال (12)

* Example (12)

```
#include<iostream.h>
#include<conio.h>
main( )
{
int  i  ;
Clrscr ( ) ;
for ( i = 17 ; i >= 0 ; i - = 2 )
Cout<< "  " << i ;
}
```

```
17 15 13 11 9 7 5 3 1
```

في المثال السابق لو كان شرط التوقف يساوي 1 فأنه سيعطي نفس الناتج لانه حلقة التكرار ستطبع الاعداد التي هي اكبر او تساوي شرط التوقف .

مثال ١ اكتب برنامج يقوم بطباعة مضاعفات العدد 7 بشكل عمودي .

* مثال (13)

* Example (13)

```
#include<iostream.h>
#include<conio.h>
main( )
{
int  i  ;
Clrscr ( ) ;
Cout << " Multiplied Of Number Seven "  ;
for ( i = 7 ; i <= 70 ; i + = 7 )
Cout<< "  " << i << endl  ;
}
```

```
7
14
21
28
35
42
49
56
63
70
```

الجديد في المثال رقم (13) هو الترتيب العمودي للنتائج و الامر بسيط جداً فهو متمثل بالايعاز (endl) و الذي سيرتب ناتج هذا البرنامج بالشكل العمودي و فائدة هذا الايعاز هو ترتيب البرنامج و اظهار نتائجه بشكل خطوات مترتبة الواحدة فوق الاخرى و ليس خطية .

* Note :- قد تكون هذه الطريقة مفيدة لصنع برنامج يعمل كجدول ضرب لأي عدد يحدده المبرمج في حلقة التكرار (Loop) .

2nd - The Loop Repetition - While

ثانيا - حلقة التكرار (While)

While : و تختلف اختلاف كبير عن حلقة التكرار (For) من حيث كتابة الايعـــاز و معـــــالجة البيانات و هي بسيطة جداً لاحظ المثال رقم (14) .

* Example (14)

* مثال (14)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i = 0  ;
While ( i <= 9 )
{
i ++ ;
Cout<< "  " << i  ;
}
}
```

البرنامج الســـــابق و ظيفتـــه طـــبع الاعداد من 1 الى 10 لاحظ شرط توقف حلقة التكرار هو الوصول الى الرقم 9 و مقدار الزيادة هو واحد (i++) و ناتج هذا البرنامج هو كما في الشكل ادناه .

```
1 2 3 4 5 6 7 8 9 10
```

The Global Base

القاعدة العامة

```
While ( Variable .. < .. > .. == .. <= .. >= .. != .. ! .. Last Value )
{
Variable ++ .. -- ;
any value that you want to repeat it ;
}
```

ان لكل حلقة تكرار بداية و الـ (While) ايضا لها بداية و لكن لم تظهر بدايتها في القاعدة العامة السابقة لها .. ؟؟
الجواب : لو نلاحظ المثال رقم (14) و الذي اختص بطباعة الارقام من 1 الى 10 نرى ان الوصف هو الذي قام بتحديد بداية عمل الـ (Loop) حيث تم وصف المتغير (i) و اعطائه بداية الحلقة (صفر) و شرط توقف الحلقة هو الـ (Last Value) و الذي سيتوقف عنده التكرار و مقدار الزيادة variable ++ , -- ...

* Note :- لاحظ المثال رقم (14) و شاهد اقواس البداية و النهاية للحلقة و هذا يعني انه اي قيمة يراد تكرارها تكتب مابين هذين القوسين و في المثال قمنا بتكرار قيمة الـ (i) عشر مرات و لو كتبت الايعاز ادناه لتكررت قيمة الـ (x) عشر مرات

```
Cout<< " X " ;
```

الـــ (Loop) التنازلي لحلقة التكرار (While) .

* مثال (15)

* Example (15)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i = 11  ;
While ( i >= 1 )
{
i -- ;
Cout<< "  " << i  ;
} ———————— نهاية الحلقة
} ———————— نهاية البرنامج
```

```
10 9 8 7 6 5 4 3 2 1 0
```

الـــ (Loop) التنازلي لحلقة التكرار (While) و لكن لاحظ كيفية كتابة ايعاز النقصان بواحد .

* مثال (16)

* Example (16)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i = 11  ;
While ( i >= 1 )
{
i - = 1 ; ———————— ايعاز النقصان بواحد
Cout<< "  " << i  ;
} ———————— نهاية الحلقة
} ———————— نهاية البرنامج
```

```
10 9 8 7 6 5 4 3 2 1 0
```

3rd - The Loop Repetition - Do_While

ثالثا - حلقة التكرار (Do_While)

Do While : و تختلف اختلاف بسيط عن حلقة التكرار (While) من حيث كتابة الايعاز و معالجة البيانات و هي بسيطة ايضاً ..
لاحظ القاعدة العامة لها :

```
Do
{
variable + = 1 ;      Or      variable - = 1 ;
any value that you want to repeat it  ;
}
while ( variable .. < .. > .. == .. <= .. >= .. != .. ! .. Last Value )  ;
}
```

* Example (17)

* مثال (17)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i = 0  ;
Do
{
i + = 1 ;
Cout<< i  << "  "  ;
}
While ( i <= 10 ) ;
}
```

البرنامج السابق و ظيفته طبع الاعداد من 1 الى 10 لاحظ شرط توقف حلقة التكرار هو الوصول الى الرقم 9 و مقدار الزيادة هو واحد (++i) او (i+=1) كما جاء في المثال و ناتج هذا البرنامج هو كما في الشكل ادناه .

```
1 2 3 4 5 6 7 8 9 10 11
```

ماذا لو اردنا ان يكون ناتج المثال رقم (17) بشكل عمودي ؟ و ما هو الايعاز الذي يرتب ناتج البرنامج بالشكل العمودي ؟؟
الجواب : لاحظ الايعاز الاتي و لو اردت استبداله بالايعاز الموجود في البرنامج لسوف ترى ترتيب الاعداد بالشكل عمودي و الـ (endl) هو المسؤول .

```
Cout<< i << " " << endl  ;
```

مثال \ اكتب برنامج لطباعة الارقام الزوجية الاصغر من عشرة بأستخدام حلقة التكرار (Do_While) و اطبع الناتج تصاعديا و بشكل عمودي .

* مثال (18)

* Example (18)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i = 0  ;
Do
{
i + = 2 ;
Cout<< i  << "  " << endl ;
}
While ( i <= 10 ) ;
}
```

... المصفوفات ذات البعد الواحد ...

... The One Dimension Matrixes ...

ان طريقة التعامل مع المتغيرات و الاعداد المدخلة في المواضيع السابقة كانت عبارة عن وصف يتكون على الاكثر من ثلاث مواقع محجوزة في الذاكرة يتم ادخال القيم اليها عن طريق لوحة المفاتيح و السؤال الان ماذا لو ازداد عدد المتغيرات ليصل الى سبع قيم او اكثر مثلا في برنامجنا ؟ في هذه الحال ستكون المصفوفات هي الحل الامثل و التي ستكون اختصار لسبع ايعازات او اكثر للـ (Cin) حسب الوصف في البرنامج .

المصفوفات : عبارة عن مواقع فارغة محددة تحجز في الذاكرة لكي يتم ملئ هذه المواقع عند تنفيذ البرنامج او يمكن اعطائها قيم عند البرمجة ليتم معالجتها لاحقا و تكتب المصفوفة برمجياً بالشكل (a [ ]) حيث نكتب بين القوسين سعة المصفوفة و (a) عبارة عن متغير يمثل اسم المصفوفة .

لاحظ البرنامج التالي و قارن بينه و بين البرنامج رقم (20) حيث البرنامج رقم (19) اختصر خمسة ايعازات للادخال و كلاهما يعطي نفس الناتج :

* مثال (19)

* Example (19)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i , a[ 5 ] ;
for ( i = 0 ; i <= 5 ; i + = 1 )
Cin >> a[ i ] ;
Cout<< a[ i ] << "  " ;
}
```

* مثال (20)

* Example (20)

```
#include<iostream.h>
#include<conio.h>
main( )
{
int  i , j , k , l , m ;
Clrscr ( ) ;
Cin>> i ;
Cin>> j ;
Cin>> k ;
Cin>> l ;
Cin>> m ;
Cout<< i << j << k << l << m   ;
}
```

لاحظ في المثال رقم (19) كيف تمت عملية قراءة المصفوفة بأستخدام حلقة التكرار (For) و استخدمنا المتغير (i) لتحديد بداية و نهاية الحلقة بالأضافة الى مقدار الزيادة (i++) و الـ (Cout) كانت للمتغير نفسه ففي هذه الحالة ستبدأ الحلقة بالعدد صفر و تنتهي بالعدد خمسة فعند مرور الحلقة على الاعداد يقوم ايعاز الـ (Cout) بطباعة العدد الذي تمر عليه الحلقة (الـ Loop في المثال السابق ترتيبه تصاعدي لاحظ مقدار الزيادة كيف تمت كتابة الايعاز الخاص به) .

التعامل مع مصفوفة محددة بقيم مبرمجة مسبقا

Deal With The Matrix That Values Given Already

ان البرمجة بأي لغة كانت عبارة عن افكار تكتب لحل المسائل المعقدة و البسيطة بأختصار و المصفوفة هي احد طرق الاختصار في البرمجة ففي المثال رقم (19) عند تنفيذه يطلب خمسة قيم لأدخالها الى البرنامج حسب سعة المصفوفة التي تم وصفها و الفائدة من جعل البرنامج يطلب قيم من المستخدم هو لجعل البرنامج اكثر مرونة في التعامل مع القيم المُدخلة لحل مسائل اكثر , اما الان فسوف نتعامل مع مصفوفة مقيدة بقيم يحددها المبرمج لتؤدي و ظيفة معينة مبرمجة مسبقاً مثال رقم (21) .

```
a [ 10 ] = { X , X , X , X , X , X , X , X , X , X }
```

مثال ١ اكتب برنامج لطباعة مصفوفة محددة بارقام معينة داخل البرنامج .

* مثال (21)

* Example (21)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i  ,  a [ 5 ] = { 20 , 30 , 10 , 40 , 50 }  ;
for ( i = 0 ; i <= 5 ; i + = 1 )
Cout<< a[ i ] << "  " ;
}
```

```
20 30 10 40 50
```

* راجع المثال رقم (26) .

... الجملة الشرطية (if) ...

... The Condition Phrase (IF) ...

الجملة الشرطية : عبارة عن ايعاز برمجي ذو طريقين يكتب و معه قيمة معينة تحدد سير البرنامج الى احد هذين الطريقين فأذا تحقق الكود الذي يأتي بعد جملة الشرط فالبرنامج يسلك الطريق الاول اما اذا لم يتحقق فأنه سيسلك الطريق الثاني للبرنامج , لاحظ طريقة كتابة الايعاز للجملة الشرطية (IF) ..

```
IF ( variable 1 < , > , == , >= , <= , % , != .... variable 2 )
1st Way ;
Else
2nd Way ;
```

القاعدة السابقة احتوت على المتغير الاول و الذي سيكون مقيد بحدوث المتغير الثاني (Variable 2) فأذا تحققت العملية الرياضية الموجودة بين المتغيرين فأنه سيسلك احد الطرق التي يحددها له المبرمج و اذا لم يتحقق فأنه سيسلك الطريق الثاني الذي حدد من قبل المبرمج ايضاً .. لاحظ المثال رقم (22) :

* مثال (22)

* Example (22)

```
#include<iostream.h>
#include<conio.h>
main ( )
{
Int   i , z  ;
Clrscr ( ) ;
Cin >> i >> z ;
if ( i > z )
Cout<< i <<" Is Large Number " ;
if ( i < z )
Cout<< i <<" Is Small Number " ;
if ( i == z )
Cout<< " There Is No Difference Between Them " << i << " = " << z ;
}
```

لو كانت مدخلات البرنامج السابق هي (3 and 7) سيكون الناتج كالاتي :

```
7  Is Large Number
```

و لو كانت مدخلات البرنامج هي (17 and 4) سيكون الناتج كالاتي :

```
4  Is Small Number
```

او كانت (17 and 17) سيكون الناتج كالاتي :

```
There Is No Difference Between Them  17 = 17
```

* Note :- في التنفيذ الاول كانت القيم هي (3 و 7) حيث الرقم 7 وضع في المتغير (i) لانه المتغير (i) سبق المتغير (z) عند الادخال في ايعاز الـــ (Cin) اي انه عند التنفيذ سندخل قيمة الـ (i) و من ثم قيمة الـــ (z) .

امثلة حول الـ (IF) و علاقتها بحلقات التكرار

Examples About The Relation Between (IF) And Loop Repetition .

مثال \ اكتب برنامج لطباعة الارقام الزوجية بين الصفر و 10 بأستخدام حلقة التكرار (For) .

* مثال (23)

* Example (23)

```
#include<iostream.h>  ---------- ( 1 )
#include<conio.h>  ---------- ( 2 )
main( )  ---------- ( 3 )
{  ---------- ( 4 )
Clrscr ( ) ;  ---------- ( 5 )
int  i  ;  ---------- ( 6 )
for ( i = 0 ; i <= 10 ; i ++ )  ---------- ( 7 )
if ( i % 2 == 0 )  ---------- ( 8 )
Cout << i ;  ---------- ( 9 )
}  ---------- ( 10 )
```

* (1) دالة مكتبية تفسر ايعازات الادخال و الاخراج للبرنامج .
* (2) دالة مكتبية تفسر ايعاز مسح الشاشة (Clrscr) و وجودها او عدم وجودها لا يؤثر على نتائج البرنامج فهي لمسح مخلفات تنفيذ سابق .
* (3) البرنامج الرئيسي .
* (4) بداية البرنامج الرئيسي .
* (5) ايعاز مسح الشاشة .
* (6) وصف المتغيرات .
* (7) حلقة التكرار و التي ستقرأ الاعداد ما بين الصفر و الـ (10) جميعها .
* (8) جملة الشرط و هي التي ستحدد الاعداد التي ستطبع على الشاشة و الشرط هو ان تكون الاعداد الموجودة ما بين الصفر و الـ (10) لاتحوي على باقي عند قسمتها على العدد (2) و الايعاز الذي اهتم بهذه العملية (باقي القسمة) هو الايعاز (%) فأذا كان باقي القسمة للمتغير عند قسمته على العدد (2) يساوي صفرا اطبع الاعداد التي لا تحوي على باقي عند قسمتها على (2) .
* (9) ايعاز اظهار النتائج على شاشة التنفيذ بعد المعالجة .
* (10) نهاية البرنامج الرئيسي .

مثال \ اكتب برنامج لطباعة الارقام الفردية بين الصفر و 10 بأستخدام حلقة التكرار (For) .

* مثال (24)

* Example (24)

```
#include<iostream.h>  ---------- ( 1 )
#include<conio.h>  ---------- ( 2 )
main( )  ---------- ( 3 )
{  ---------- ( 4 )
Clrscr ( ) ;  ---------- ( 5 )
int  i  ;  ---------- ( 6 )
for ( i = 0 ; i <= 10 ; i ++ )  ---------- ( 7 )
if ( i % 2 != 0 )  ---------- ( 8 )
Cout << i ;  ---------- ( 9 )
}  ---------- ( 10 )
```

* (1) دالة مكتبية تفسر ايعازات الادخال و الاخراج للبرنامج .
* (2) دالة مكتبية تفسر ايعاز مسح الشاشة (Clrscr) و وجودها او عدم وجودها لا يؤثر على نتائج البرنامج فهي لمسح مخلفات تنفيذ سابق .
* (3) البرنامج الرئيسي .
* (4) بداية البرنامج الرئيسي .
* (5) ايعاز مسح الشاشة .
* (6) وصف المتغيرات .
* (7) حلقة التكرار و التي ستقرأ الاعداد ما بين الصفر و الـ (10) جميعها .
* (8) جملة الشرط و هي التي ستحدد الاعداد التي ستطبع على الشاشة و الشرط هو ان تكون الاعداد الموجودة ما بين الصفر و الـ (10) تحوي على باقي عند قسمتها على العدد (2) و الايعاز الذي اهتم بهذه العملية (باقي القسمة) هو الايعاز (=! بمعنى لا يساوي) فأذا كان باقي القسمة للمتغير عند قسمته على العدد (2) يساوي واحد اطبع الاعداد التي تحوي على باقي .
* (9) ايعاز اظهار النتائج على شاشة التنفيذ بعد المعالجة .
* (10) نهاية البرنامج الرئيسي .

مثال \ بأستخدام مصفوفة ذات N من الاعداد اكتب برنامج لطباعة الارقام الزوجية بشكل عمودي بواسطة حلقة التكرار (Do-While) .

* مثال (25)

* Example (25)

```
#include<iostream.h>  ---------- ( 1 )
#include<conio.h>  ---------- ( 2 )
main( )  ---------- ( 3 )
{  ---------- ( 4 )
Clrscr ( ) ;  ---------- ( 5 )
int  i = 0 , a[ 10 ] ;  ---------- ( 6 )
Do  ---------- ( 7 )
{  ---------- ( 8 )
Cin>>a[ i ] ;  ---------- ( 9 )
IF ( a[ i ] % 2==0 )  ---------- ( 10 )
Cout<<a [ i ] << "  " << endl  ;  ---------- ( 11 )
}  ---------- ( 12 )
While ( i < 10 )  ;  ---------- ( 13 )
}  ---------- ( 14 )
```

* (1) دالة مكتبية تفسر ايعازات الادخال و الاخراج للبرنامج .
* (2) دالة مكتبية تفسر ايعاز مسح الشاشة (Clrscr) و وجودها او عدم وجودها لا يؤثر على نتائج البرنامج فهي لمسح مخلفات تنفيذ سابق .
* (3) البرنامج الرئيسي .
* (4) بداية البرنامج الرئيسي .
* (5) ايعاز مسح الشاشة .
* (6) وصف المتغيرات .
* (7) و هو الايعاز الذي سينفذ الايعازات المتواجدة ما بين الاقواس .
* (8) هذا القوس سيكون بداية الحلقة و كل ما موجود بداخله سيتكرر لاحقا عن طريق الـ (While) في الخطوة رقم (13) .
* (9) هنا سيتم ادخال البيانات الى المصفوفة و لتكن الارقام من واحد الى عشرة .
* (10) شرط الطباعة : و هو اذا كانت اعداد المصفوفة تقبل القسمة على (2) بدون باقي .... .
* (11) اطبع الارقام التي تقبل القسمة على (2) بدون باقي .
* (12) قوس نهاية الايعازات التي ستكرر .
* (13) حلقة التكرار : و التي سوف تكرر الاعداد ابتداءآ من الصفر و تنتهي بشرط التوقف .
* (14) نهاية البرنامج الرئيسي .

* أختبار :: في البرنامج السابق لو اردت طباعة الارقام الفردية فما التغيير الذي سيحدث في البرنامج ؟

Examination :: In Last Program If You Want To Print Odd Numbers What You Will Change In It ?

مثال \ البرنامج التالي احتوى على مصفوفة مكونة من عشرة مواقع و هي مقيدة بقيم محددة و سنستخدم في هذا البرنامج الايعاز المنطقي (&&) بمعنى (و) و البرنامج سيقوم بطباعة الاعداد الموجودة داخل المصفوفة بشرط ان يكون العدد اكبر من 50 و اصغر من 200 كالاتي :

* مثال (26)

* Example (26)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr ( ) ;
int  i  ,  a [ 10 ] ={ 111 , 40 , 177 , 100 , 199 , 30 , 70 , 50 , 20 , 10 } ;
for ( i = 0 ; i <= 10 ; i + = 1 )
IF ( ( a[ i ] >=50 ) && ( a[ i ] <=200 ) )
Cout<< a[ i ] << "  " ;
}
```

* Note :- لاحظ طريقة ربط الشرط الاول بالثاني بواسطة (&&) حيث وضِع كلآ منهما داخل قوسين و تم ربط القوسين بالأيعاز (&&) و كِلا الشرطين داخل اقواس آخرى و هي الاقواس الاساسية للجملة الشرطية (IF) .

اختبار :: أجرِ التعديلات الاتية على البرنامج رقم (26) بعدها نفذ و لاحظ النتائج .

* اجعل البرنامج يطبع الارقام الاكبر من 30 و الاصغر من 100 .
* ضع فارزة بين الاعداد التي ستظهر بعد التنفيذ .
* غير ايعاز الزيادة بواحد في الـ (Loop) بأيعاز آخر مشابه بالعمل .
* أضف اعداداً الى المصفوفة و غير ما يلزم تغييره . القصد الى ما بين القوسين { } .
* نفذ !! .

مثال \ اكتب برنامج يقوم بفحص عدداً ما اذا كان اولياً ام لا .

* مثال (27)

* Example (27)

```
#include<iostream.h>
#include<conio.h>
main( )
{
Clrscr( ) ;
int   i , j=0 , x ;
Cin>> x ;
For ( i = 1 ; i <= x ; i++ )
IF ( x % i == 0 )
++ j  ;
IF (j<=2)
Cout<< x << " Is Prime Number " ;
Else
Cout<< x << " Is Not Prime Number "  ;
}
```

المثال رقم (27) : كـانت وضيفتـه هـي اختبار العدد الـمُدخل للبرنامج اذا كان يقبل القسمة على نفسه و على الواحد فقط (اي عدداً اولياً) او غير ذلك و فكرة البرنامج بسيطة و تكمن في مهارة المبرمج في التحكم بعمل حلقة التكرار و الشرط لذلك احتجنا في البرنامج الى ثلاث متغيرات و هي الـ (x) و الذي يمثل القيمة التي ستعالج و الـ (i) لعمل حلقة تكرار , و متغير آخر سيعمل كعداد و سيأتي شرحه لاحقا .
طريقة عمل البرنامج : عند التنفيذ سيطلب البرنامج ادخال قيمة عددية من لوحة المفاتيح و لتكن الرقم (7) بعدها سيأتي عمل حلقة التكرار حيث سنقوم بصنع حلقة تكرار تكون بدايتها هي الرقم (1) و تنتهي بالعدد الذي نقوم بأدخاله (العدد الذي سيُفحص < x >) .. لماذا ؟؟ .
الجواب : عندما تبدأ حلقة التكرار بالعمل ستظهر لديها الاعداد مابين الواحد و (7) (1 2 3 4 5 6 7) و جملة الشرط التي ستأتي بعد حلقة التكرار ستقوم بقسمة العدد (7) على الاعداد التي ظهرت في الحلقة و كلما قبل العدد (7) القسمة على احد اعداد الحلقة بدون باقي فأن المتغير (j) سوف يزداد بمقدار واحد .. هذا هو الشرط الاول ..
اما الشرط الثاني فهو : اذا كان المتغير (j) يحوي على قيمة اصغر او تساوي (2) فأن العدد الذي قمت بأدخاله اولياً و اذا كانت بخلاف ذلك فأن العدد غير اولي بمعنى آخر اذا اصبح للمتغير (j) اكثر من قيمتين هذا يعني ان العدد قد قبل القسمة على اكثر من عددين في الحلقة لذا فالعدد غير اولي .. و هذا هو الشرط الثاني .

... الدوال ...

... Functions ...

Function : عبارة عن برنامج فرعي (كيان مستقل) استدعائه ذاتي يحوي على مجموعة من الايعازات و التي تقوم بوظيفة معينة حيث يقوم المبرمج بأستدعائها في البرنامج الرئيسي لكي تؤدي وظيفتها داخله .

ان الدوال هي جزء من البرنامج و لقد صممت لغرض الاختصار في كتابة الاكواد الخاصة باللغة و لها فوائد اخرى منها :

1 - تختصر البرنامج اي يكفي استدعائها بواسطة كتابة اسمها في البرنامج الرئيسي لكي تقوم بالعمل المطلوب منها .

2 - تجنب المبرمج التكرار في كتابة الايعازات .

3 - توفر مساحة خزنية في الذاكرة .

4 - تختصر زمن البرمجة اذا كانت لديك دوال جاهزة .

5 - البرنامج المكتوب بواسطة الدوال تكون معالجته للبيانات المدخلة اليه اسرع عند التنفيذ .

The Global Base — القاعدة العامة

```
#include<iostream.h>
int , float , char , string variables ; ———— ( 1 )
Function Name ( ) ———— ( 2 )
{ ———— ( 3 )
the instructions of function ———— ( 4 )
} ———— ( 5 )
main( ) ———— ( 6 )
{
Function Name ( ) ; ———— ( 7 )
}
```

* (1) وصف المتغيرات .
* (2) اسم الدالة و هو اي اسم يكتبه المبرمج و يحبذ ان يكون دال على عمل الدالة .
* (3) قوس بداية الدالة .
* (4) ايعازات الدالة او الكود الذي سيكتب لكي يُنَفذ عند الاستدعاء .
* (5) قوس نهاية الدالة .
* (6) البرنامج الرئيسي .
* (7) هنا عند كتابة اسم الدالة بالطريقة المذكوره اعلاه (القاعدة العامة) يكون البرنامج الرئيسي قد استدعى الدالة للتنفيذ اي استدعى الدالة للقيام بالوظيفة الخاصة بها و المُبَرمَجة على اساسها .

* Note : توجد طريقة أخرى لكتابة الدوال وهي ان نجعل وصف المتغيرات و ايعاز الادخال داخل الدالة نفسها .. كيف ؟؟ لاحظ القاعدة الثانية ادناه .

```
#include<iostream.h>
Function Name ( )
{
int , float , char  variables ;
Cin >> variable ;
the instructions of function ;
}
main ( )
{
Function Name ( ) ;
}
```

في القاعدة الثانية للدالة سيكون لديها نوع من الاستقلالية عن البرنامج الرئيسي حيث ان المبرمج سوف يقوم بالوصف و كتابة ايعاز الادخال داخل الدالة نفسها و ان وصف المتغيرات داخل الدالة يكون مقتصراً عليها اذ لا يمكن وصف متغير داخل الدالة و استخدامه في دالة اخرى .. مثال رقم (28) على القاعدة الثانية .

مثال : بأستخدام الدالة اكتب برنامج لأختبار عدد ما اذا كان فردياً او زوجياً و قم بجعل الدالة تأخذ البيانات و تعالجها و من ثم تقوم بطباعتها .

* مثال (28)

* Example (28)

```
#include<iostream.h>
#include<conio.h>
one ( ) ———————————————— ( 1 )
{ ———————————————— ( 2 )
int  x  ;
Cin >> x  ;
IF ( x % 2==0 )
Cout << " Even Number "  ;
Else
Cout << " Odd Number "  ;
} ———————————————— ( 3 )
main ( )
{
Clrscr ( ) ;
one ( ) ; ———————————————— ( 4 )
}
```

عند تنفيذ البرنامج يقوم المترجم (Compiler) بقراءة محتوى البرنامج الرئيسي (Main) و سوف يقوم بقراءة الاسم (one) داخل الـ (Main) ماذا سيفعل عندما وجد هذا الاسم الذي يأتي بعده قوسين و فارزة منقوطة ؟؟

الجواب : سيفهم من خلال الاسم و القوسين و الفارزة المنقوطة ان هنالك دالة تحمل الاسم الذي قام بقراءته و من خلال الاسم و الذي هو عبارة عن عنوان الدالة سيذهب الى العنوان (one) ليرى ما موجود بداخله لكي يقوم بتنفيذه .

* (1) هنا يتم تعيين اي اسم للدالة و حاول ان يكون الاسم يعبر عن وظيفة الدالة .
* (2) قوس بداية الدالة .
* (3) قوس نهاية الدالة .
* (4) الان و في البرنامج الرئيسي نكتب اسم الدالة منتهي بالفارزة المنقوطة لكي تُنَفذ و هذا ما يسمى بالاستدعاء .

* **Note** : لو تلاحظ الدالة في البرنامج رقم (28) نرى انها عبارة عن برنامج متكامل من حيث الوصف و الادخال و الاخراج لذلك يمكن ان تسمى الدوال بالبرامج الفرعية و لو احتوى البرنامج الرئيسي على اكثر من دالة عندئذٍ يسمى بالبرنامج المتفرع او المتعدد الوظائف .

مثال : بأستخدام الدالة اكتب برنامج لأختبار عدد ما اذا كان اولياً او لا .

* مثال (29)

* Example (29)

```
#include<iostream.h>
#include<conio.h>
two ( )
{
int   i , j=0 , x ;
Cin>> x ;
For ( i = 1 ; i <= x ; i++ )
IF ( x % i == 0 )
j ++ ;
IF (j<=2)
Cout<< x << " Is Prime Number " ;
Else
Cout<< x << " Is Not Prime Number "  ;
}
main ( )
{
Clrscr ( ) ;
two ( ) ;
}
```

في الامثلة السابقة كانت البرامج مكونة من دالة واحدة و لو اردنا ان يكون برنامجنا اكثر اقبالا من قبل المستخدمين لابد ان يكون هنالك تنويع من حيث معالجة البيانات اي يجب علينا جعل البرنامج يقوم بتوفير اكثر من نوع من المعالجة لزيادة اقبال المستخدمين عليه اذ انه سيوفر لهم جمع البرامج لانك قد قمت مثلا بأختصار برنامجين في برنامج واحد . كيف ؟؟

الدوال مَكنت المبرمجين من جعل البرامج اكثر مرونة في معالجة البيانات حيث اصبح المبرمج يصنع اكثر من دالة و كل واحدة تختص بوظيفة معينة و يقوم بدمجها و استدعائها في البرنامج الرئيسي لتأدية وظائفها .

ان الدوال المتعددة في البرنامج تحتاج الى وسيلة تربطها بالبرنامج الرئيسي و هذه الوسيلة هي المتغيرات و بما ان البرنامج الرئيسي ارتبط بالدالة بواسطة متغير اصبح لدينا ما يسمى بـ (تمرير البيانات) و القصد منه نقل محتوى المتغير الذي يربط بين الدالة و الـ (main) الى البرنامج الرئيسي نفسه .. كيف ؟؟ لاحظ القاعدة رقم (3) و المثال رقم (30) خاص بها .

```
#include<iostream.h>
int Function Name ( int  2nd Variable ) ———————— ( 1 )
{
the instructions of function  ;
Return ( 2nd Variable )  ; ———————— ( 2 )
}
main ( )
{
int   1st Variable  ; ———————— ( 3 )
Function Name ( 1st Variable )  ; ———————— ( 4 )
}
```

* (1) لاحظ الوصف قبل اسم الدالة كان صحيحاً اي ان هذه الدالة تتعامل مع الاعداد الصحيحة اما بعد اسم الدالة يأتي وصف المتغير الذي سيقوم بربط الدالة بالبرنامج الرئيسي و هذا المتغير سيمتلك ناتج العملية التي حصلت داخل الدالة لهذا السبب قمنا بأعادته الى البرنامج الرئيسي .
* (2) هذا الايعاز سيقوم بأعادة المتغير الذي يحمل نواتج الدالة الى البرنامج الرئيسي .
* (3) هنا في البرنامج الرئيسي يجب وصف متغير ليقوم بأستقبال القيمة التي اعادتها الدالة الى البرنامج الرئيسي .
* (4) أخيرا نكتب اسم الدالة و المتغير الذي وصفناه لكي تتم المعالجة .

* مثال (30)

* Example (30)

```
#include<iostream.h>
#include<conio.h>
int even ( int x )  ——— الدالة الاولى : دالة فحص الرقم اذا كان فرديا او زوجيا
{
if ( x % 2 == 0 )
Cout<< " even "  ;
Else
Cout<< " odd " ;
Return ( x )  ;
}
int Prime ( int  y )  ——— الدالة الثانية : دالة فحص الرقم اذا كان اولي او غير اولي
{
int   i , j=0 ,  x  ;
for ( i =1 ; i <= x ; i++ )
if ( x % i == 0 )
j++ ;
if ( j <= 2 )
Cout<<" = "<<" prime "  ;
Else
Cout<< " = "<< " not prime "  ;
Return ( y ) ;
}
main ( )  ——— البرنامج الرئيسي الذي سيتم استدعاء الدوال فيه
{
Clrscr ( )  ;
int c ;
Cin>> c  ;
Even ( c )  ;
Prime ( c )  ;  ——— استدعاء الدوال
}
```


GALAXY

مثال : اكتب برنامج لايجاد ناتج المعادلة :- $S = X^{10} + X^{8} + X^{6} + X^{4} + X^{2}$

* مثال (31)

* Example (31)

```
#include<iostream.h>
#include<math.h>
main ( )
{
int  S=0 , X , i  ;
Cin>> X  ;
For ( i = 10 ; i >=2 ; i-=2 )
S+= pow ( X , i )  ;
Cout<< S  ;
}
```

مثال : نفس البرنامج السابق لكن بدون استخدام الدالة الرياضية :

* مثال (32)

* Example (32)

```
#include<iostream.h>
main( )
{
int  s=0 , x , k , j , i  ;
Cin>> x  ;
For ( i=10 ; i>=2 ; i-=2 )
{
k = 1  ;
For ( j=1 ; j<= i ; j++ )
k * = x  ;
S+=k  ;
}
Cout<<S  ;
}
```

مثال : اكتب برنامج لفحص عدد معين اذا كان اولي او غير اولي .

*** مثال (33)**

*** Example (33)**

```
#include<iostream.h>
main( )
{
int   i , j=0 , x ;
Cin>> x ;
For ( i = 1 ; i <= x ; i++ )
IF ( x % i == 0 )
++ j  ;
IF (j<=2)
Cout<< x << " Prime " ;
Else
Cout<< x << " Not Prime "  ;
}
```

مثال : اكتب برنامج يقرأ (50) رقم و يستخرج الاكبر من بينهم بأستخدام الدالة :

*** مثال (34)**

*** Example (34)**

```
#include<iostream.h>
void max ( )
{
int a[50] , i=0  ;
while ( a[ i ] !=10 )
{
Cin>> a[ i ]  ;
i++;
}
for ( i=1 ; i<50 ; i++ )
if ( a[ 0 ] < a[ i ] )  a[ 0 ] = a [ i ]  ;
Cout<< a[ i ]  ;
}
main ( )
{
max ( ) ;
}
```

مثال : اكتب برنامج لقراءة عدد من الرموز في مصفوفة ذات (N) من الاعداد ثم اوجد ما يلي :
** زد كل حرف في المصفوفة بمقدار 3 (شفر الحرف) .
** احسب تردد الحروف (اي كم مرة تم ذكر الحرف في الادخال)
** قم بعكس المصفوفة .

*** Example (35)** *** مثال (35)**

```
#include<iostream.h>
Char inc ( char b[ 100 ] ) ———————— دالة زيادة الحرف بمقدار 3
{
int  m , n  ;
for ( n=0 ; n<100 ; n++ )
{
m = b[ n ] + 3  ;
if ( m < 122 ) m- = 26  ;
b[ n ] = m  ;
}
for ( n=0 ; n<100 ; n++ )
Cout << b[ n ]  ;
Return b[ n ]  ;
}
Char freq ( Char c[ 100 ] ) ———————— دالة حساب تردد الحرف
{
int  j , k , F [ 26 ] ;
for ( j=0 ; j<26 ; j++ )
F [ j ] = 0 ;
for ( j=0 ; j<100 ; j++ )
{
k = c [ j ] - 97  ;
F [ k ] + = 1  ;
}
for ( j=0 ; j< 26 ; j++ )
Cout<< Char ( j + 97 ) << '' = '' << F [ j ] << endl  ;
Return F [ j ]  ;
}
Char rev ( Char d [ 100 ] )  ; ———————— دالة عكس المصفوفة
{
int  S , t  ;
for ( s=0 ; s<50 ; s++ )
{
t = d [ s ]  ;
d [ s ] = d [ 99 - s ]  ;
d [ 99 - s ] = t  ;
}
for ( s=0 ; s< 100 ; s++ )
Cout << d [ s ] ;
Return d [ s ] ;
}
main ( ) ———————————————— البرنامج الرئيسي
{
Char a [ 100 ]  ;
int  i  ;
for ( i=0 ; i<100 ; i++ )
Cin >> a [ i ]  ;
inc ( a )  ;
freq ( a )  ; ———————————————— استدعاء الدوال
rev ( a )  ;
}
```

* برنامج لحساب تردد الحروف المُدخلة اليه بأستخدام الدالة .

* مثال (36)

* Example (36)

```
#include<iostream.h>
#include<conio.h>
Char freq ( char  c [ 100 ] )
{
int   j , k , f [ 26 ]  ;
For ( j=0 ; j<26 ; j++ )
f [ j ]=0  ;
For ( j=0 ; j<100 ; j++ )
{
k = c [ j ] - 97 ;
f [ k ]+= 1  ;
}
For ( j=0 ; j<26 ; j++ )
Cout<< Char ( j + 97 )<<" = "<< f [ j ] << endl   ;
Return f [ j ]   ;
}
main( )
{
Clrscr ( ) ;
int   i  ;
Char a[ 100 ]  ;
for( i=0 ; i<=100 ; i++ )
Cin>> a [ i ]  ;
Freq ( a )  ;
}
```

*Note :- تردد الحرف يقصد به عدد المرات التي ذُكر بها الحرف في نص مُدخل الى البرنامج .

* برنامج لحساب تردد الحروف المُدخلة اليه بدون أستخدام الدالة .

* مثال (37)

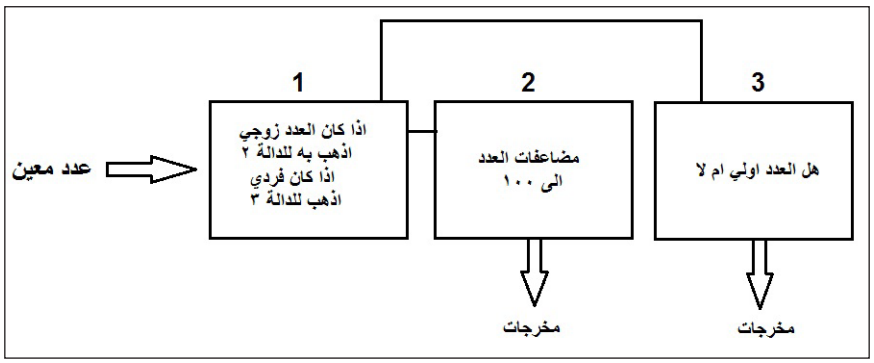
* Example (37)

```
#include<iostream.h>
#include<conio.h>
main ( )
{
Clrscr ( ) ;
Char  C [ 100 ]  ;
int    i , j , k , f [ 26 ]  ;
For ( i=0 ; i<100 ; i++ )
Cin>> C [ i ]  ;
For ( j=0 ; j<26 ; j++ )
f [ j ]=0  ;
For ( j=0 ; j<100 ; j++ )
{
k = c [ j ] - 97 ;
f [ k ]+= 1  ;
}
For ( j=0 ; j<26 ; j++ )
Cout<< Char ( j + 97 )<<" = "<< f [ j ] << endl   ;
Return f [ j ]   ;
}
```

... التمارين العامة ...

... Annum Examinations ...

** اكتب برنامج لايجاد معدل اربع درجات .
** اكتب برنامج لطباعة اكبر رقم من بين ثلاثة ارقام .
** اكتب برنامج لطباعة الاعداد الزوجية ما بين 7 و 1 تنازلياً .
** اكتب برنامج لطباعة مضاعفات العدد 5 الاصغر من 35 .
** اكتب برنامج يقوم بفحص الرقم اذا كان سالبا او موجب .
** اكتب برنامج لطباعة ارقام مصفوفة ذات (N) من الاعداد بالمقلوب .
** اكتب برنامج يقوم بالبحث عن اسم داخل مصفوفة .
** اكتب برنامج لعمل دالتين الاولى للجمع و الثانية للضرب و اجعل القراءة و الطباعة في البرنامج الرئيسي .
** اكتب برنامج لفحص اعداد مصفوفة فأذا كان العدد فردي ضعه في مصفوفة أخرى و من ثم اطبعها .
** اكتب برنامج لقراءة اعداد مصفوفة ثم رتب الناتج تصاعديا .
** اكتب برنامج لطباعة الـ (asci) لسبع حروف .
** اكتب برنامج لقراءة مجموعة من الاعداد الصحيحة ثم بين عدد الارقام التي تقبل القسمة على 7 بدون باقي .
** اكتب برنامج يقوم بقراءة عدد من الارقام الصحيحة و افحص الرقم اذا كان سالب او موجب بأستخدام الدالة .
** اكتب البرنامج الاتي :



The Author And Book Programmer
Hussam Munthir - B
Book Auditor
PROF : Yamama Abd Al-Hussain
Mohammed Ahmed
Programming Auditor
Mohammed Ahmed
PDF Creator
X-PC Seven : Hussam Munthir
PDF Created Using
Adobe InDesign CS4
Date
2012/4/30

The End

For Any Suggestion And Objection To Make
This Book Having Many Information
For It's Subject C++ Report Us At
xcxcomputer@yahoo.com
xpc.seven@yahoo.com