بسم الله الرحمن الرحيم

الحمد لله وكفى والصلاة على نبيينا محمد عليه أفضل الصلاة و أتم التسليم.ثم أما بعد.
هذا شرح مبسط على الدوال راعية فيه السهولة في الأمثلة و محاولة الشرح الجيد ولكن هذا جهد المقل فما كان من صواب فمن الله وما كان من خطأ فمن نفسي والشيطان و أرجوا تعديل أي خطأ هنا وأعلامنا قبل التعديل و بعده.
أرجوا أن يكون هذا العمل خالصا لوجهه الله سبحانه وتعالى.

******* الآن نذهب إلى الدرس ********.

* الدوال (Function):

* تعريفها:

هي برنامج فرعي يقوم بمعملية أو عمليات على مدخلات ما وقد يعيد قيمة وقد لا يعيد قيمة.

* تتكون الدالة من ثلاثة أشياء رئيسية هي:

1 - الإعلان عن الدالة (رأس الدالة).
2 - استدعاء الدالة.
3 – جسم الدالة (تعريف الدالة).
ويمكن الاستغناء عن الإعلان عن الدالة ويكفي تعريف الدالة فيصبح حقيقة الإعلان عن الدالة وتعريفها ولكن من الأفضل في حالات كثيرة الإعلان عن الدالة في الأعلى ومن ثم تعريفها في الأسفل وذلك حتى لا تقع في أخطاء استدعاء دالة معينة قبل تعريفها..

* - الإعلان عن الدالة (رأس الدالة).
وهو أن تعلن عن نوع القيمة المعادة من الدالة وإذا لم يكن هنالك قيمة تعديدها الدالة فيكون نوعها دائما من النوع (void). وكذلك تعلن عن اسم الدالة وتعلن عن الوسائط أو (المتغيرات المرسلة للدالة) وتسمى (parameter) وهي تأخذ الشكل العام التالي:

```
Output_Type    Function_Name(parameter1, parameter2,…);
```

حيث:

Output_Type " هي نوع القيمة المعادة من الدالة أي هل تعيد الدالة قيمة صحيحة int أم تعيد قيمة حقيقة float وغيرها من الأنواع وإذا لم يكن هنالك قيمة تعديدها الدالة فيكون نوعها دائما من النوع (void) ".

Function_Name " وهو يعني اسم الدالة وهو أي اسم صحيح تنطبق عليه قواعد التسمية بلغة c /c++ ومن الأفضل دائما اختيار أسماء تدل على معنى ما تقوم به الدالة فمثلا من أجل دالة تقوم بعملية جمع نسميها مثلا (SUM) أو (Add) أو أي اسم يخطر على بالك فقط لابد أن يحقق الشروط ".

(parameter1, parameter2,…); " بعد اسم الدالة تفتح قوساً ")" ثم تضع الوسائط المرسلة للدالة هنا يفصل بين كل متغير والأخر فاصلة ", " وكل متغير يحتوي على نوع (int,char,…) و اسم لهذا المتغير ".

ملاحظة:
يتم الإعلان عن الدالة خارج دالة ال(main) بشكل عام. و لابد أن ينتهي الإعلان بفاصلة منقوطة.

* مثال على الإعلان عن دالة تقوم بإرسال وسيطين (عددين) وتعيد مجموعهما.

```
int   Add( int  x, int  y );
```

هنا تم الإعلان عن دالة اسمها Add وتعيد قيمة صحيحة int وتأخذ متغيرين صحيحين وسائط مدخلة للدالة من النوع الصحيح وينتهي الإعلان عن الدالة بفاصلة منقوطة.
يتم الإعلان عن الدالة ليخبر المترجم للغة c++ بنوع القيمة المعادة واسم الدالة وأنواع القيم المرسلة للدالة وكم عددها وهكذا ولكن لا يتم حجز مكان في الذاكرة حتى الآن .
يتم حجز مكان في الذاكرة إذا وصل المترجم إلى جسم الدالة وهو المكان الذي تؤدي الدالة فيه عملها.

* مثال على استدعاء الدالة:
تستطيع استدعاء الدالة في أي مكان في البرنامج سواء في الدالة الرئيسية main أم استدعاء دالة من دالة أخرى أم دالة تستدعي نفسها وكل ذلك سنعرفه بالتفصيل لاحقا – إن شاء الله - .
وبشكل عام يتم استدعاء الدالة بكتابة الشكل العام التالي:

```
Function_Name (parameter1, parameter2,…);
```

أي تكتب اسم الدالة والقيم المرسلة للدالة بدون ذكر أنواعها فقط.وإذا كانت الدالة لا تأخذ أي قيم أي تم الإعلان عنها أنها لا تأخذ أي قيم مرسلو ولا أي وسائط فتكتفي في عملية الاستدعاء بكتابة اسم الدالة وقوسين فارغين وفاصلة منقوطة.
مثلا : استدعاء دالة Add يتم كالتالي:

```
int n =  Add(10,2);
```

وهنا كأننا نقول " أذهب إلى جسم الدالة Add وخذ معك القيمتين الصحيحتين 10,2 وقم بعملية جمعهما وأعد الناتج وضعه في المتغير n ".

ملاحظة:
يمكن استدعاء الدالة وطباعة الناتج مباشرة إذا كنا لا نريد تخزن الناتج لاستخدامه في أغراض أخرى كالتالي:

```
Cout  <<  Add(10,2) << "\n";
```

* مثال على تعريف الدالة :
الدالة السابقة Add تعلمنا كيف نعلن عنها وكيف نقوم باستدعائها والآن نعرف كيف نعرفها.
تعريف الدالة أو جسم الدالة هو كتابة مجموعة الخطوات البرمجية التي توصل إلى الحل وتعيد لنا القيمة إلى مكان استدعاء الدالة.
مثلا جسم الدالة add قد يكون بالشكل التالي:

```
int  Add( int  x, int  y )
{
      int  z;
      z  =  x +y;
      return  z;
}
```

جسم الدالة Add هنا مشابه للإعلان عن الدالة ولكن الاختلاف هنا أنه لا يتم وضع فاصلة منقوطة بعد التعريف أي بعد الأقواس ( ) وإنما يتم وضع أقواس من النوع { } ليدل على بداية ونهاية الدالة جسم الدالة .
جسم الدالة هنا يحتوي على تعليماتها التي سوف تنفذ وهي كالتالي .
عرفنا متغير من النوع الصحيح واسمه z لنستخدمه كمخزن نخزن فيه ناتج عملية جمع العددين الممررين للدالة و هما x,y ثم نعود بقيمة عملية الجمع المخزنة بالمتغير z عن طريق التعليمة return .

* عمل التعليمة return :

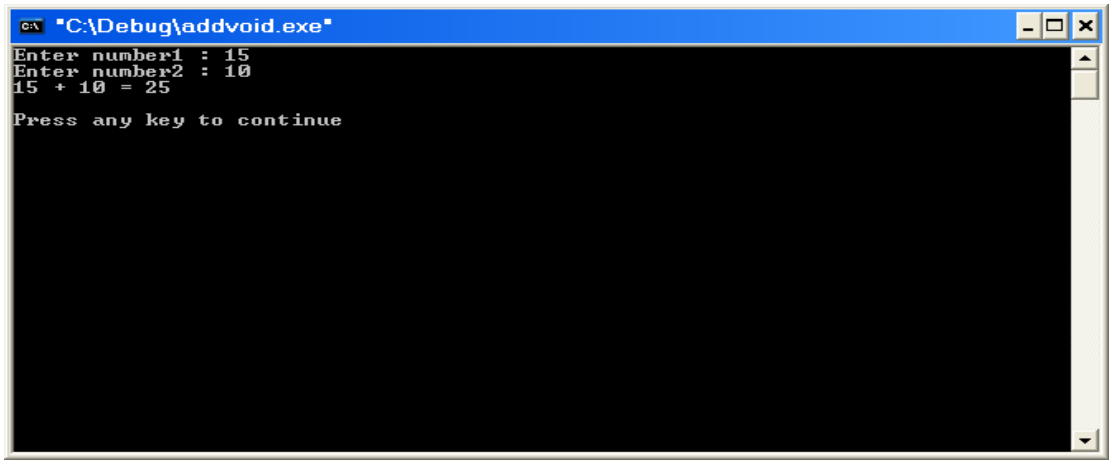
هذه التعليمة تستخدم إذا كنا نعيد قيمة من الدالة فلابد من كتابة الكلمة return ثم كتابة القيمة التي نعيدها من الدالة مثل المثال السابق return z; فهنا كأننا نقول " أعد القيمة الصحيحة z إلى المكان الذي تم استدعاء الدالة Add فيه ".
أما إذا كانت الدالة لا تعديد قيمة أي الدالة من النوع void فلا نحتاج أبدا لكتابة التعليمة return أبدا وليس لها أي وجود لأننا لا نعيد قيمة من الدالة.

أمثلة على الدوال:

1 – أكتب دالة تطلب من المستخدم إدخال عددين ثم طباعة مجموعهما؟
لحل هذه المسألة البسيطة لابد من تحليلها . هذه دالة تطالب المستخدم بإدخال عددين أي لا يمرر لها أي متغير أي أنها لا ترسل لها مدخلات وفي هذه الحالة تكون القيم (الوسائط الممررة للدالة من النوع الخالي void ويمكن أن تكتب void أو تضع أقواس فارغة من النوع ( )).
وهذه الدالة مهمتها جمع العددين اللذين يدخلهما المستخدم إذا الحل هو:

```
#include <iostream.h>


//declaration Function Add

void  Add();

int main()
{
        //Call Function Add

        Add();

        return 0;
}

//definition Function Add

void  Add()
{
```

```
        float x,y;

        cout << "Enter number1 : ";
        cin  >> x;
        cout << "Enter number2 : ";
        cin  >> y;

        cout << x << " + " << y << " = " << x+y << "\n\n";
}
```



مع بساطة هذه الدالة ألا أننا سنتكلم عنها بالتفصيل حتى إذا فهم المثال البسيط فكتابة المثال المعقد تفهم بسهولة.

شرح البرنامج:

في البداية تم الإعلان عن الدالة Add أنها من النوع void أي لا تعيد قيمة وأنها لا تأخذ أي وسائط وانتهاء الإعلان عنها بفاصلة منقوطة.

وفي الدالة الرئيسة main( ) تم استدعاء الدالة Add بكتابة اسمها فقط والأقواس الفارغة لأنه لا يمرر لها أي متغيرات.

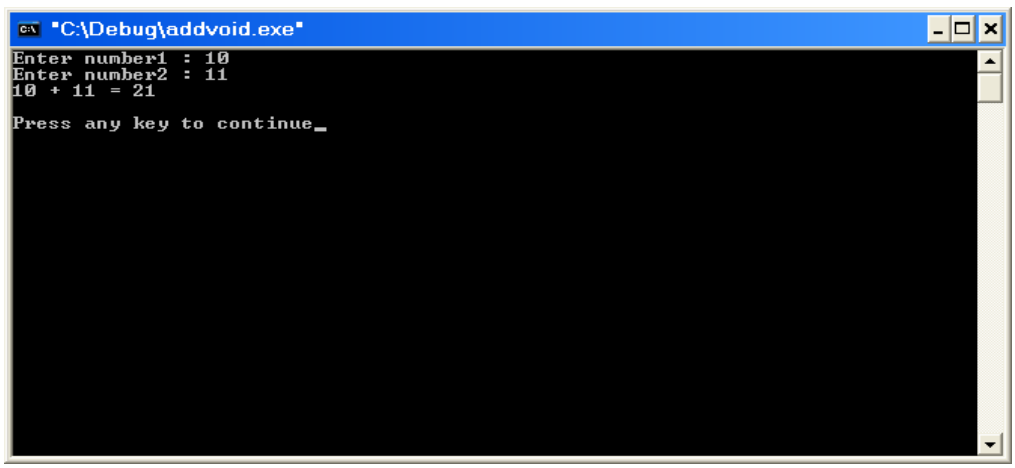
وفي جسم الدالة Add تم تعريف متغيرين من النوع الحقيقي float وهما x,y وتم عمل إدخال لهاما بجملتي cout و cin كما هو واضح وبعد ذلك تم طباعة قيمة الجمع وذلك بعبارة cout الأخيرة مع بعض التنسيق في المخرجات.

2 – نفس المثال السابق ولكن هنا الدالة Add تعيد قيمة حقيقة هي ناتج عملية الجمع:

الحل هو:

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

```
//Function Add

float  Add( float x, float y);

int main()
{
        float a,b;

        cout << "Enter number1 : ";
        cin  >> a;
        cout << "Enter number2 : ";
        cin  >> b;

        cout << a << " + " << b << " = " << Add(a,b) << "\n\n";

        return 0;
}


float  Add( float x, float y)
{
        return x+y;
}
```

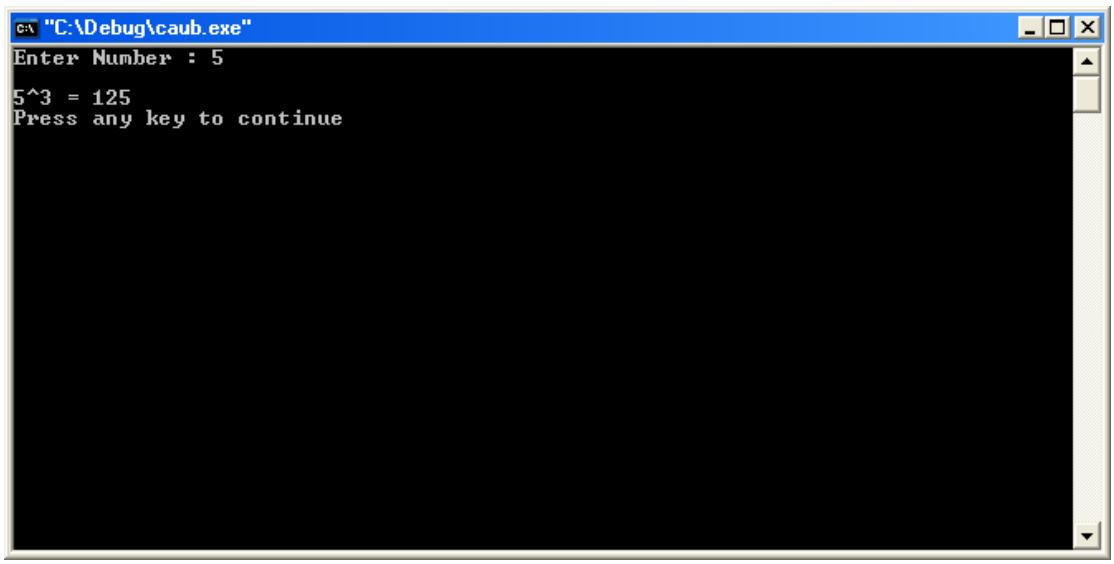


شرح البرنامج :
في هذا البرنامج تم الإعلان عن دالة اسمها Add تعيد قيمة من النوع الحقيقي float وتأخذ قيمتين تمرران إلى الدالة وهم من النوع الحقيقي float x,y وينتهي الإعلان كما قلنا من قبل بفاصلة منقوطة.
وفي جسم الدالة هنالك تعليمة واحدة وهي return x+y; والتي تعيد قيمة المجموع للعددين إلى المكان الذي تم استدعاء الدالة منه .
في الدالة main عرفنا متغيرين حقيقيين float a,b وطلب من المستخدم كما هو موضح في الحل إدخال العددين ثم طباعة المجموع وذلك باستدعاء دالة Add مع تمرير القيمتين التين نريد جمعهما كما هو موضح Add(a,b) .وتعاد القيمة للمجموع هنا وتتم طباعتها.

3 – أكتب دالة تعيد مكعب عدد :
لحل هذا المثال نعرف أن مكعب أي عدد هو ضرب العدد في نفسه ثلاث مرات أي أنه إذا كان x أي عدد فأن x^3 هي x*x*x ولذلك سنكتب الدالة التالية:

```
#include <iostream.h>


// دالة مكعب عدد

int Caub( int x );

int main()
{
	int a;

	cout << "Enter Number : ";
	cin  >> a;

	cout << "\n" <<  a << "^3 = " << Caub(a) << "\n";

	return 0;

}

int Caub( int x )
```

```
{
        return x*x*x;
}
```



شرح البرنامج:
في هذا البرنامج تم الإعلان عن الدالة caub(int x) والتي تأخذ متغير صحيح واحد فقط وتعيد قيمة صحيحة هي مكعب العدد المرسل لها.
ثم في الدالة main() عرفنا متغير صحيح هو a والذي أدخلنا قيمته و تم استدعاء الدالة caub(a) لطباعة القيمة المعادة من الدالة وهي مكعب العدد.
في جسم الدالة caub وضعنا الجملة return x*x*x والتي نعني بها أعد قيمة العدد بعد ضربها في نفسها ثلاث مرات وهي ناتج مكعب العدد ويمكن كتابة هذه الجملة على سطرين ولكن من الأفضل كتابتها على سطر واحد للاختصار في أسطر البرنامج وتكتب على سطرين هكذا

```
Int s;
s = x*x*x;
return s;
```

فنعرف متغير ونسند له قيمة مكعب عدد ثم نعيد هذا المتغير والذي يحتوي على قيمة الكعب للعدد.

4 – أكتب دالة تطبع أي رسالة نصية نرسلها لها علما أن الرسالة لا تتعدى أحرفها 80 حرفا:
من الواضح جدا أن هذه الدالة لا تعيد أي قيمة فلذلك ستكون حتما خرجها من النوع void وستأخذ متغير واحد وهو سلسلة من الحروف النصية أي من النوع char ولذلك سيكون البرنامج كالتالي:

```
#include <iostream.h>

// دالة طباعة سلسلة نصية

void PrintMasig( char string[80] );

int main()
{
	char str1[80];
	char str2[80];

	cout << "Enter string1 : ";
	cin.getline(str1,79,'\n');
	cout << "Enter string2 : ";
	cin.getline(str2,79,'\n');

	PrintMasig( str1 );
	PrintMasig( str2 );



	return 0;

}

void PrintMasig( char string[80] )
{
	cout << "\n" << string << "\n";
}
```

```
"C:\Debug\caub.exe"
Enter string1 : www.arabteam2000.com is good
Enter string2 : exit end program

www.arabteam2000.com is good

exit end program
Press any key to continue_
```

شرح البرنامج :
في هذا البرنامج تم الإعلان عن الدالة PrintMasig(char string[80) والتي من النوع void وتأخذ متغير واحد هو سلسلة من الحروف النصية ثم تقوم بطباعتها كما هو موضع في جسم الدالة أما في الدالة main() فقد تم تعريف سلسلتين نصيتين هما str1 & str2 وتم إدخالهما من قبل المستخدم وباستخدام الدالة cin.getline() وهي دالة جاهزة من دوال اللغة تأخذ ثلاث وسائط على الترتيب هي السلسلة المراد قراءتها ثم عدد الأحرف التي تريد قراءتها ثم الحرف الذي تنتهي عنده القراءة.ثم استدعاء الدالة لطباعة السلسة str1 واستدعاء الدالة لطباعة السلسلة str2 .

وهنا نوضح أنه يمكن استدعاء الدالة أكثر من مره وتقوم في كل مره بعمل ما أي مثلا في الدالة السابقة يمكن أن نضع تكرار for والذي يكرر عدد الطلاب في فصل ما ونطبع أسمائهم باستدعاء الدالة السابقة عدد من المرات مساوي إلى عدد الطلبة.

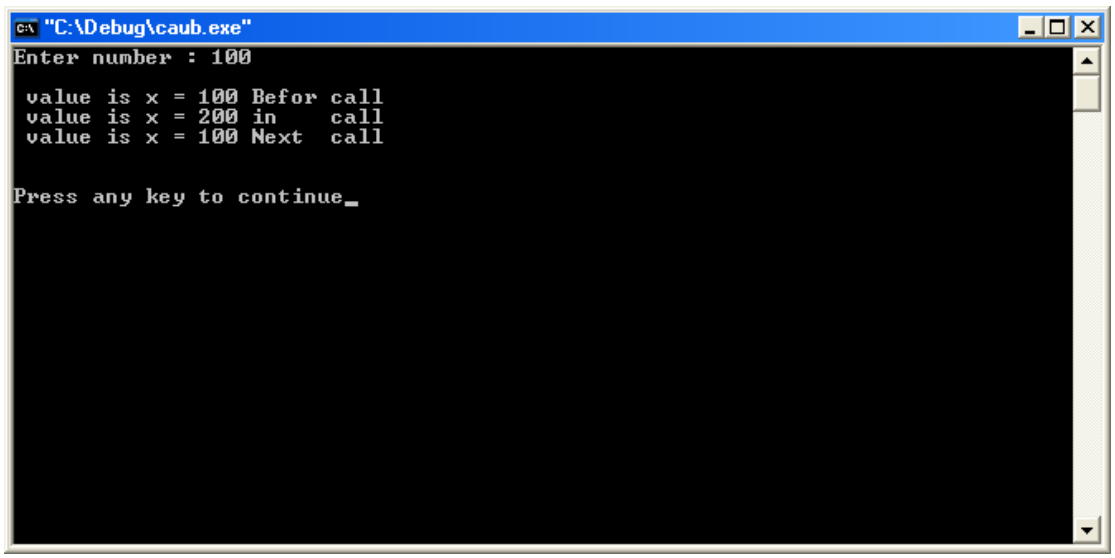
.....

سنتكلم الآن عن موضوع أنواع تمرير المتغيرات إلى الدوال:
هنالك نوعين رئيسيين هما :
1 – التمرير بالقيمة (bass value).
2 – التمرير بالمؤشرات (bass pointer).

وسأضرب لك مثال على التمرير بالقيمة لنعرف خصائصه وعيوبه:

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

```
//دالة التمرير بالقيمة
void PrintNumber( int x );

int main()
{
	int x;

	cout << "Enter number : ";
	cin  >> x;


	cout << "\n value is x = " << x << " Before call";
	PrintNumber( x );
	cout << "\n value is x = " << x << " Next
call\n\n\n";





	return 0;

}

void PrintNumber( int x )
{
	x = x * 2;
   cout << "\n value is x = " << x << " in  call";
}
```



شرح البرنامج :

- لابد أن تركز معي هنا وتحاول أن تفهم جيدا الموضوع –

هنا تم الإعلان عن الدالة PrintNumber(int x) والتي تأخذ متغير وحيد يمر لها بالقيمة وفي جسم هذه الدالة فقط نعيد القيمة بعد مضعفتها أي بعد ضربها ب2 . واستدعاء هذه الدالة يتم بالقيمة (bass value). حيث لاحظ معي الناتج في التنفيذ:

طلب من المستخدم إدخال رقم فأدخل المستخدم العدد 100 ثم تم طباعة العدد قبل استدعاءه ويظهر الناتج العدد 100 .

ثم تم استدعاء الدالة وإرسال المتغير x بالقيمة أي (أننا نرسل قيمة x حقيقة وهي العدد100) وفي جسم الدالة تم ضرب القيمة 100 في العدد 2 فأصبح الناتج 200 وتم إعادته إلى المكان الذي استدعي منه وطباعة القيمة 200 كما هو واضح في ناتج البرنامج.

وبعد استدعاء البرنامج تم طباعة قيمة x وظهر الناتج هو100 ولم يظهر 200 !!لماذا؟!.

الجواب هو أنه عند الاستدعاء بالقيمة فأن مايحدث هو عمل نسخة أو عدة نسخ من المتغير x ثم أرسالها إلى الدالة عند الاستدعاء فتتغير قيمتها في جسم الدالة وعند العودة من استدعاء الدالة تعاد قيمة النسخة بعد ضربها ب2 وتبقى قيمة x بعد الاستدعاء كما هي لم تتغير.

أي باختصار أن التمرير بالقيمة يعني عمل نسخة أو عدة نسخ ثم العمل في جسم الدالة يتم على النسخة وليس على الأصل وبذلك تعاد النسخة بعد إجراء العمليات عليها وتبقى القيمة الأصل دون تغيير.

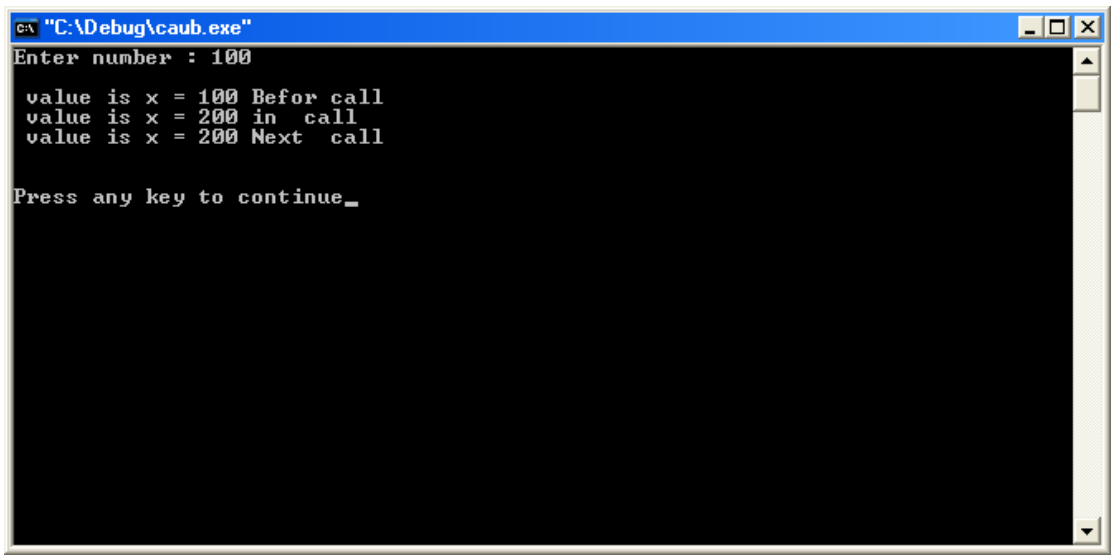
ولذلك إذا أردنا عمل تغيير في القيمة بعد استدعاء الدالة لابد أن نستخدم التمرير بالمؤشرات حيث سيتم تمرير القيمة بالعنوان أي سنرسل للدالة عند استدعائها عنوان المتغير ثم نجري العمليات على العنوان والذي سنضع القيمة المعادة في هذا العنوان وبالتالي مايحدث هو أن قيمة المتغير ستحتفظ بأخر قيمة له وليس عمل نسخة منه.

أي أن التمرير بالعنوان هو إرسال عنوان المتغير لجسم الدالة ثم القيام بالعمليات عليها ووضع القيمة للمتغير في العنوان ولذلك سيحافظ على القيمة الجديدة التي حصل عليها بعد استدعاء الدالة. ولايتم هنا عمل نسخ من المتغير كما هو موجود في التمرير بالقيمة والذي يأخذ حيز كبير في الذاكرة أما التمرير بالعنوان فلا يحتاج إلى حيز كبير في الذاكرة.

*مثال على التمرير بالمؤشرات أو التمرير بالعنوان (bass Address):
نفس المثال السابق ولاحظ التغييرات:

```
#include <iostream.h>

// التمرير بالعنوان

void PrintNumber( int * px );

int main()
{
   int x;

   cout << "Enter number : ";
   cin  >> x;


   cout << "\n value is x = " << x << " Before call";
    PrintNumber( &x );
   cout << "\n value is x = " << x << " Next
call\n\n\n";
```

```
    return 0;

}

void PrintNumber( int * px )
{
    *px = *px * 2;
   cout << "\n value is x = " << *px << " in  call";
}
```



شرح البرنامج:

في هذا البرنامج تم الإعلان عن الدالة PrintNumber(int * px) بأنها دالة من النوع void وأنها تأخذ متغير عبارة عن مؤشر إلى عدد صحيح .

أي أن التمرير للدالة سيتم بالعنوان كما سترى بعد قليل.

في الدالة main() تم تعريف المتغير الصحيح x والطلب من المستخدم إدخال قيمة فأدخل المستخدم القيمة 100 ثم بعد ذلك تم طباعة قيمة x وهي 100 قبل استدعاء الدالة.

ثم تم بعدها استدعاء الدالة PrintNumber(&x) وتم هنا التمرير بالعنوان حيث استخدمنا المعامل العنونة & وبعده اسم المتغير الذي نريد تمريره للدالة وهو x . " حيث نقول مرر عنوان المتغير x لجسم الدالة PrintNumber() ".

وهنا سينتقل تنفيذ البرنامج إلى جسم الدالة .

وفي جسم هذه الدالة تتم مضاعفة القيمة بعد ضربها ب2 ونطبع القيمة وذلك باستخدام المعامل * كما يلي:

*px = *px * 2;

ومعنى (px*) أي القيمة الموجودة بالعنوان الذي يشير إليه المؤشر px.
أي إذا كان عنوان x مثلا 0x123 والذي يحتوي على القيمة 100 فأن المؤشر px سيشير على عنوان المتغير x ولذلك قلنا px* أي القيمة التي يشير إليها المؤشر اضربها ب2 وضع الناتج في العنوان الذي يشير إليه المؤشر أي أن القيمة 200 سنضعها في العنوان 0x123 .ولذلك تصبح قيمة x = 200 .
بعد خروجنا من الدالة ثم نعود إلى السطر التالي في دالة ()main فنطبع قيمة x فنجدها 200 لأننا عندما خرجنا من جسم الدالة تغيرت قيمة x من 100 إلى 200 .

ولذلك فالتمرير بالمؤشرات أفضل لأنه يتم عمل نسخة واحدة فقط من المتغير وليس نسخ عديدة ويتم التمرير بالعنوان والذي يعطي مجال أكبر للاستفادة من العنونة .كذلك التمرير بالمؤشرات يسمح لنا بالعودة بأكثر من قيمة من الدالة كما سترى في باقي الدرس.

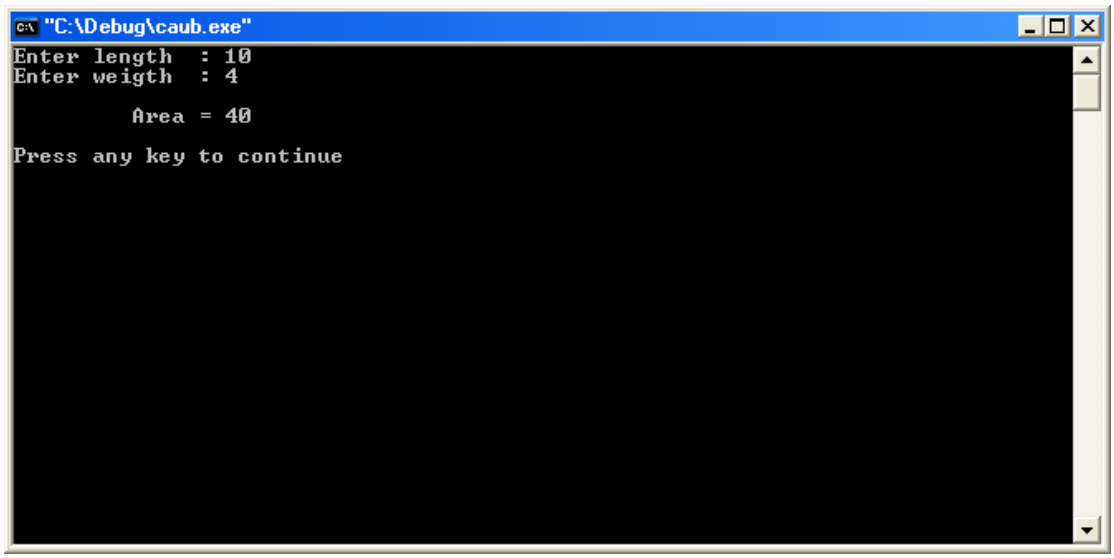
ملاحظة هنا يتم شرح الدوال ولن يتم شرح المؤشرات بل سيتم تناولها على أنك عزيز القارئ تعرف الإعلان عن المؤشرات وكيفية استخدامها في البرامج.

- مثال على التمرير بالعنوان :

* دالة حساب مساحة مستطيل والمساحة هي الطول * العرض .
البرنامج:

```
#include <iostream.h>


//compute Rectangle Area

int RastingleArea( int * length , int * weight );



int main()
{
     int   leng,wegt;
```

```
    cout << "Enter  length  : ";
    cin  >> leng;
    cout << "Enter weight  : ";
    cin  >> wegt;

    cout << "\n\t Area = " << RastingleArea(
&leng,&wegt )  << "\n\n";

    return  0;
}

int RastingleArea( int * length , int * weight )
{
    return ( *length *  *weight );
}
```



شرح البرنامج:

في البداية تم الإعلان عن الدالة RastingleArea() والتي تأخذ متغيرين هما مؤشرين إلى عددين صحيحين هما length & weight وتعود بقيمة مساحة المستطيل وهي ضرب الطول في العرض.

في الدالة main() تم الإعلان عن قيمتين هم leng,wegt صحيحتين وتم إدخال قيمة الطول والعرض كما هو واضح ثم تم استدعاء الدالة RastingleArea() وتمرير القيم بالعنوان حيث يتم ذكر معامل العنونة & ثم بعده اسم المتغير الذي تم تمريره بالعنوان هكذا &leng,&wegt ثم في جسم

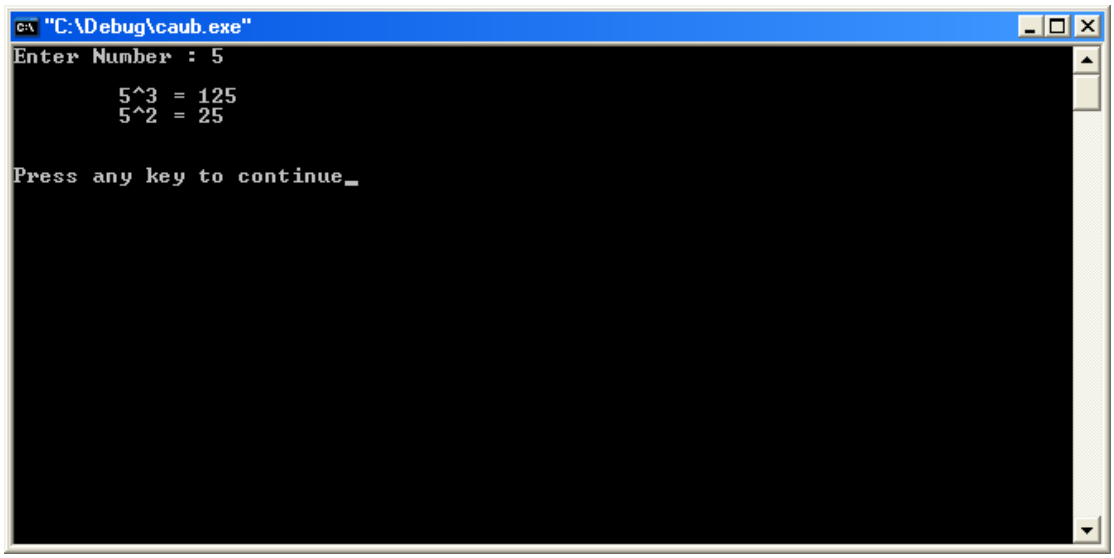
الدالة تم ضرب القيمتين باستخدام المعامل * هكذا:weight* * length*
وكأننا نقول خذ القيمة الموجودة بالعنوان length وضربها بالقيمة الموجودة بالعنوان weight ثم عد بحاصل العملية لمكان استدعاء الدالة وأطبع القيمة الناتجة.

*إعادة أكثر من قيمة من الدالة باستخدام المؤشرات والتمرير بالعنوان:
- الدالة تعيد قيمة واحدة فقط في كل مرة تستدعى ولكن يمكن تجاوز هذا القيد باستخدام المؤشرات وسنشرح الطريقة باستعمال المثال التالي وشرحه تفصيليا .

*مثال ليكن عندنا دالة ونريد أن تعود بمربع ومكعب رقم منها؟
الحل:

```
#include <iostream.h>


int DoubleNumber( int n, int * pn );

int main()
{
   int N,NSquer;


   cout << "Enter Number : ";
   cin  >> N;
   cout << "\n\t" << N << "^3 = " <<
DoubleNumber( N,&NSquer );
   cout << "\n\t" << N << "^2 = " <<  NSquer;
   cout << "\n\n\n";


   return 0;
}


int DoubleNumber( int n, int * pn )
{
   *pn = n * n;
```

```
    return n * n * n;
}
```



شرح البرنامج:

في البداية تم الإعلان عن الدالة DoubleNumber(int n, int * pn) والتي تأخذ قيمتين أحدهما عدد صحيح والأخرى مؤشر لعدد صحيح وتعود بقيمة صحيحة.

ثم في الدالة main() تم الإعلان عن متغيرين صحيحين هما N و NSquer وتم إدخال قيمة N ثم تم تمرير القيمتين إلى الدالة حيث تم تمرير N بالقيمة وتم تمرير NSquer بالعنوان (لكي نعيد أكثر من قيمة من الدالة لابد أن نمرر بالعنوان لكي نحتفظ بالقيمة في جسم الدالة في هذا العنوان ثم نطبعها في main).

وفي جسم الدالة تم إسناد مربع العدد للعنوان الذي يشير إليه المؤشر pn وهو نفسه عنوان المتغير NSquer أي ضع قيمة مربع العدد في عنوان المتغير NSquer . وتعود الدالة بمكعب العدد صراحة بتعليمة return n*n*n; و أيضا تعود بمربع العدد الموجود في عنوان المتغير Nsquer لأننا استخدمنا المؤشرات في الجملة *pn = n*n; " نقول ضع قيمة مربع العدد في العنوان الذي يشير إليه المؤشر pn وهو عنوان المتغير Nsquer " .

وهكذا تم طباعة السطر التالي:

```
cout << "\n\t" << N << "^2 = " << NSquer;
```

فتم طباعة العدد N وكذلك مربعه الموجود في المتغير Nsquer والذي تم تمريره للدالة بالعنوان فعنوانه احتفظ بالقيمة بعد العودة من الدالة.

- ولذلك من المهم معرفة إذا أردنا إعادة أكثر من قيمة من الدالة أن نستخدم المؤشرات والعنونة .والمثال المبسط السابق يشرح الطريقة ويمكنك تطبيق ذلك في أي دالة أخرى.

* Function overloading :
هي عبارة عن مصطلح يقصد به (التحميل الزائد للدوال) وأفضل شخصيا أن أطلق عليه (التحميل الزائد لأسماء الدوال) Function name overloading

*تعريفه:
هو وضع أكثر من دالة (إجراء) لها نفس الاسم مع اختلاف عدد الوسائط (parameter) المرسلة للدالة أو اختلاف ترتيبها أو أنواعها أو نوع خرجها.

- يمكنك وضع الكثير من الدوال التي تتميز فيما يلي:
- لها نفس الاسم.
- لها عدد مختلف من المتغيرات المرسلة للدالة.
- لها أنواع مختلفة من المتغيرات المرسلة للدالة.
- في جسم الدالة تقوم بعمل مختلف.

* إذا التحميل الزائد لأسماء الدوال هو كتابة أكثر من دالة تشترك بنفس الاسم ولكن لابد أن تختلف في عدد المتغيرات المرسلة للدالة أو ترتيبها أو نوع القيمة التي تعود بها الدالة.

*لماذا نستخدم التحميل الزائد لأسماء الدوال؟!
لأنه في الكثير من الأحيان عندنا عدد من الدوال تقوم بنفس العمل ولكن على متغيرات صحيحة مرة ومرة على متغيرات حقيقة وتعود بقيم مختلفة فمن السهل على المبرمج كتابة الدوال مشتركة بنفس الاسم مع اختلاف الوسائط والقيم التي تمرر للدالة.ولن تكون مهتم بطريقة الاستدعاء فستكون من مهمة المترجم (COMPILER) والذي سيحدد الدالة التي تستدعى بناء على الوسائط الممررة للدالة وأنواعها وترتيبها.

أي مثلا ليكن عندنا دالة مرة تجمع عددين صحيحين ومرة تجمع عددين حقيقيين فمن الأفضل تسميتها باسم واحد هو( )sum وليس باسمين ()sum1 و ()sum2 .

**** ولا يتضح المقال إلا بالمثال و إليك مثالين يوضحان المقصود:

* برنامج يقوم بمضاعفة القيمة المرسلة للدالة :

قد تحتاج إلى مضاعفة القيمة الممررة للدالة مرة تكون القيمة الممررة من النوع الصحيح (int) ومرة من النوع الحقيقي (float) ومرة من النوع الحقيقي المضاعف (double).

- فبدلا من أن تشغل نفسك بوضع ثلاث دوال لكل دالة اسم معين فلماذا لاتستفيد من هذه الخاصية (Function overloading) وتسمي الدوال الثلاث بنفس الاسم مع اختلاف القيم الممررة للدالة واليك البرنامج ليوضح هذا الكلام .

```
#include <iostream.h>


//الإعلان عن الدوال

int    Double( int x );
float  Double( float x );
double Double( double x );


int main()
{
    int IntNumber;
    float FloatNumber;
    double DoubleNumber;

    cout << "Enter Int Number : ";
    cin  >> IntNumber;
    cout << "Enter Float Number : ";
    cin  >> FloatNumber;
    cout << "Enter Double Number : ";
    cin  >> DoubleNumber;

  cout << "\n number = " << IntNumber << "\tDoubleInt = "
<< Double(IntNumber);
  cout << "\n number = " << FloatNumber << "\tDoubleInt =
" << Double(FloatNumber);
```

```
  cout << "\n number = " << DoubleNumber << "\tDoubleInt
= " << Double(DoubleNumber);

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

  cin.get();


    return 0;

}



//تعريف الدوال
int    Double( int x )
{
    return  x * 2;
}

float  Double( float x )
{
    return  x * 2;
}
double Double( double x )
{
    return  x * 2;
}
```

```
"C:\Debug\caub.exe"
Enter Int Number : 10
Enter Float Number : 3.5
Enter Double Number : 3.333

 number = 10     DoubleInt = 20
 number = 3.5    DoubleInt = 7
 number = 3.333  DoubleInt = 6.666


Press any key to continue
```

شرح البرنامج:

في البداية تم تعريف ثلاث دوال باسم واحد هو double ترجع قيم على التوالي هي int و float و double. وتأخذ كل دالة وسيط واجد في الدالة الأولى تأخذ وسيط من النوع الصحيح وفي الثانية تأخذ عدد من النوع الحقيقي وفي الثالثة تأخذ عدد حقيقي مضاعف وكل ماتقوم به الدوال السابقة هو مضاعفة القيمة الممررة للدالة وذلك بضربها في العدد 2 .

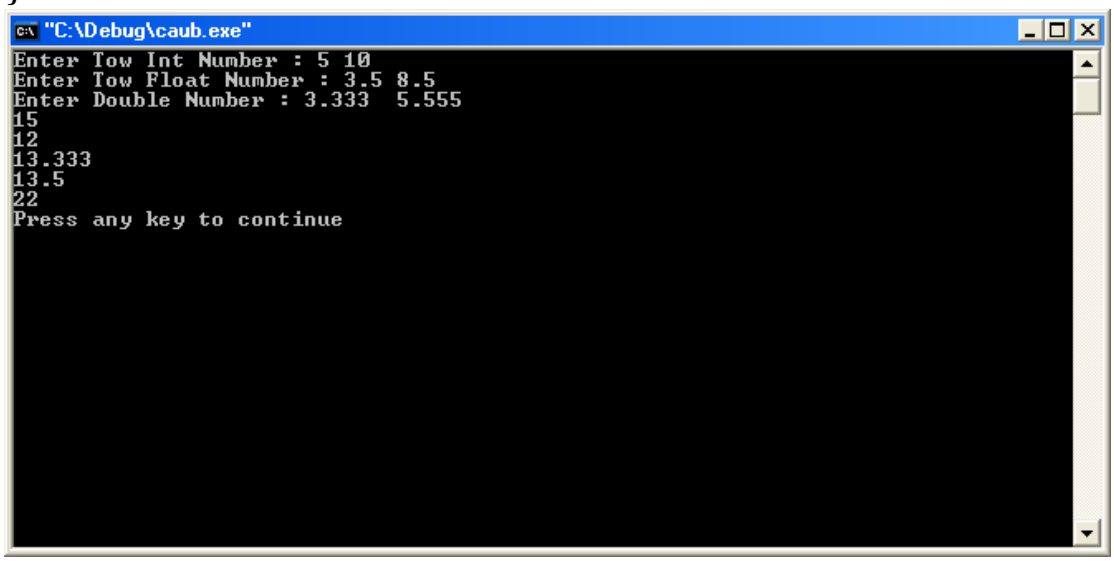
ولاحظ معي هذه الملاحظة المهمة والتي هي من فوائد التحميل الزائد لأسماء الدوال ألا وهي أنك استدعيت كافة الدوال بنفس الاسم ولم تشغل نفسك بماهو نوع القيمة التي تعود بها الدالة وكيف ستتم طريقة الاستدعاء. (هذا من عمل المترجم compiler) حيث سيقوم بالنظر في الدالة ومقارنة القيمة الممررة للدالة مع الموجود في رأس الدالة (الإعلان عن الدالة) وإذا اتفقتا في نفس النوع ينظر في القيم الممررة للدالة وأنواعها وترتيبها وإذا اتفقتا يجري العملية ويعود بالقيمة.

* مثال أخر سأقوم بشرحه بعد كتابته:

```
#include <iostream.h>


int       Add( int x, int y );
float     Add( float x, float y );
double    Add( double x, int y );
float     Add( int x, float y );
double    Add( float x,int y,float z);


int main()
```

```
{
        int IntX,IntY;
        float FloatX,FloatY;
        double DoubleX;

        cout << "Enter Tow Int Number : ";
        cin  >> IntX >> IntY;
        cout << "Enter Tow Float Number : ";
        cin  >> FloatX >> FloatY;
        cout << "Enter Double Number : ";
        cin  >> DoubleX;

        cout << Add( IntX, IntY ) << endl;
    cout << Add( FloatX, FloatY ) << endl;
    cout << Add( DoubleX, IntY ) << endl;
    cout << Add( IntX, FloatY ) << endl;
    cout << Add( FloatX, IntY , FloatY) << endl;

        return 0;


}


int    Add( int x, int y )
{
        return x + y;
}

float  Add( float x, float y )
{
        return x + y;
}

double Add( double x, int y )
{
        return x + y;
```

```
}

float  Add( int x, float y )
{
        return x + y;
}

double Add( float x,int y,float z)
{
        return x + y + z;
}
```



شرح البرنامج:

هذا البرنامج يقوم بالإعلان عن خمس دوال لها نفس الاسم تقوم بعملية الجمع للوسائط الممررة للدالة أي كان نوعها وهذه الدوال على التوالي معرفة كالأتي:

1 - تعود بقيمة صحيحة وتجمع عددين صحيحين.

2- تعود بقيمة (float) وتقوم بجمع عددين من النوع (float).

3 – تعود بقيمة من النوع (double) وتجمع عددين من النوع int & double.

4 – تعود بقيمة من النوع (float) وتجمع عددين من النوع int & float .

5 – تعود بقيمة من النوع (double) وتجمع ثلاث أعداد من النوع float & int & float.

وبعد الإعلان تم الاستدعاء بنفس الطريقة سترسل المتغيرات وتعود الدوال بقيمة بناء على القيم المرسلة ترتيبها وعددها وأنواعها. وطباعة الناتج كما هو واضح.

ولذلك استفدنا هنا كثيرا فبدل من أن نسمي دالة لجمع عددين ونسمي دالة لجمع ثلاث أعداد نسمي دالة واحدة بنفس الاسم مع اختلاف ترتيب وأنواع المتغيرات المرسلة للدالة أو القيمة المعادة منها.

تعريف دالة المعاودة (Recursion Function):
هي عبارة عن دالة مثل أي دالة (برنامج فرعي يقوم بعملية ما على مدخلات ويعود بناتج معين) ولكنها تتميز عن بقية الدوال بأنها تستدعي نفسها عدد من المرات حتى يتحقق شرط ما ، ثم تتوقف وتعود بالقيمة .

* ملاحظة :
لابد من وجود شرط(قيمة معينة) عند تحققه تعود الدالة بالقيمة ، حتى لا تكون الدالة تستدعي نفسها عدد لا نهائي من المرات.
أي لابد من شرط واحد على الأقل إذا تحقق تتوقف الدالة عن استدعاء نفسها وتعود بالقيمة.

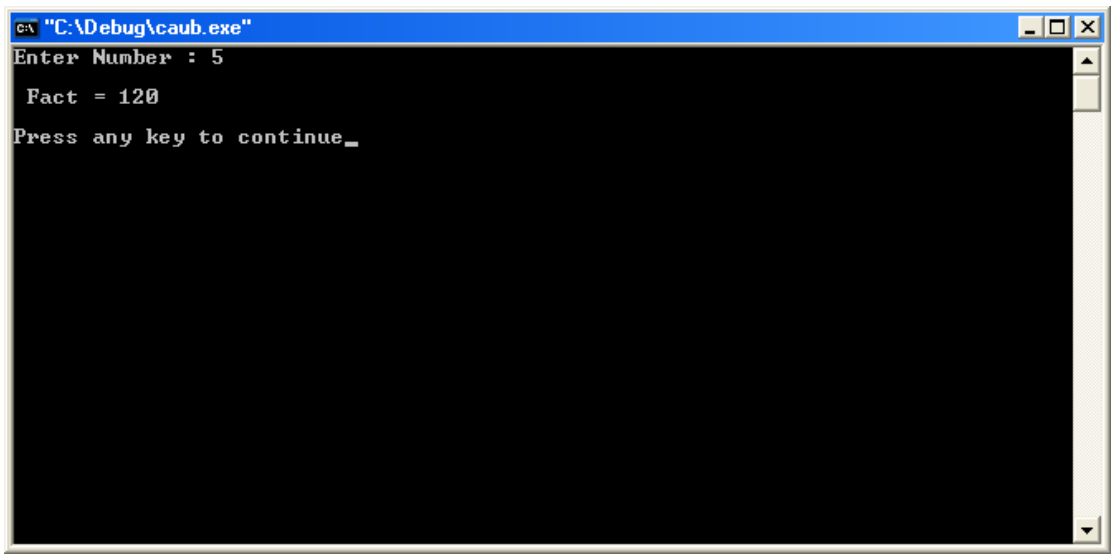
* مثال يوضح المقصود من هذا النوع من الدوال:

دالة المضروب (أو المعاملي) هي من أشهر الأمثلة على هذا النوع .
نحن نعرف المضروب بأنه التالي:
$N! = N (n-1)! (n-2)! \ldots$
مثلاً:
مضروب الخمسة هو :
$5! = 5 * 4 * 3 * 2 * 1$

*** وعندما نريد كتابة أي نوع من هذه الدوال لا بد أولا أن نفكر متى ستنتهي الدالة من استدعاء نفسها والعودة بالقيمة ؟
*** متى ما أجبنا على هذا السؤال سهل علينا كتابة البرنامج بكل سهولة.
نعود على مثالنا فنجد أن مضروب الصفر هو دائما واحد أي أن ($0! = 1$)
$5! = 5 * 4!$
$4! = 4 * 3!$
$3! = 3 * 2!$
$2! = 2 * 1!$
$1! = 1 * 0!$
$0! = 1$

أي عندما نكتب البرنامج سنجعل الدالة تكرر نفسها إلى أن تصل إلى مضروب الصفر فتعيد لمكان استدعاء الدالة القيمة واحد ثم تتم عملية ضرب القيم (5*4*3*2*1) ثم العودة بالناتج وهو (120) وهو ناتج مضروب الخمسة وهذا ما سوف يحصل بالضبط في البرنامج التالي:

```
/*******برنامج المضروب******/

#include <iostream.h>



int Fact( int N );



int main()
{
        int N;

        cout << "Enter Number : ";
        cin  >> N;

        cout << "\n Fact = " << Fact(N) << "\n\n";

        return  0;
}

int Fact( int N )
{

        if( N == 0 )
                return 1;
        else
                return N * Fact( N-1 );

}
```



هذا البرنامج البسيط يقوم بحل المضروب وإليك الشرح:
في بداية البرنامج عرفنا دالة اسمها Fact تعود بعدد صحيح ويمرر لها عدد صحيح.
في الدالة main يطالب المستخدم بإدخال عدد ثم يتم استدعاء الدالة Fact ممرر لها العدد N وتعود بقيمة المضروب ليتم طباعته.
جسم الدالة Fact يحتوي على عبارة if_else الشرطية حيث يقول (إذا كان العدد الممرر للدالة يساوي الصفر فعد بالقيمة واحد وهذا هو القيمة التي تحدد نهاية استدعاء الدالة كما قلت (0! = 1) . وإذا كان غير ذلك فقم بضرب العدد الممر للدالة (N) بالعدد الناتج من استدعاء الدالة مرة أخرى بالعبارة (Fact(N-1)) حيث ستعود الدالة هذه المرة بالقيمة (4 * Fact(4-1)) أي سيتم ضرب (5 * 4 * Fact(3)) ثم يتم استدعاء الدالة مرة أخرى – لأن الشرط مازال غير محقق – فتعود الدالة هذه المرة بالقيمة (3 * Fact(3-1)) أي سيتم ضرب القيمة (5 * 4 * 3 * Fact(2)) ثم يتم استدعاء الدالة مرة أخرى – لأن الشرط مازال غير محقق – فتعود الدالة هذه المرة بالقيمة (2 * Fact(2-1)) أي سيتم ضرب القيمة (5 * 4 * 3 * 2 * Fact(1)) ثم يتم استدعاء الدالة مرة أخرى – لأن الشرط مازال غير محقق – فتعود الدالة هذه المرة بالقيمة واحد (1) لماذا ؟
لأن Fact(1-1) يؤدي إلى Fact(0) وهنا يكون الشرط متحقق فتعود الدالة بالقيمة واحد (1) إلى السطر الذي بعد else لأنه تم استدعاء الدالة هناك فتعود بالقيمة (1) فيتم عملية الضرب (5 * 4 * 3 * 2 * 1) ويكون الناتج (120) وتعود بها الدالة إلى السطر الذي تم استدعائها فيه عند الدالة main لطباعة الناتج.

* مثال أخر وهو دالة القوة power حيث تأخذ هذه الدالة وسيطين الأول عدد وليكن (x) والثاني أس للعدد وليكن (n) ثم تقوم بعملية الرفع لأس والعودة بالناتج كالتالي x ^ n .

هنا لابد أن نبحث عن متى ستوقف الدالة أو بمعنى أصح ما هي القيم التي تنهي عملية الاستدعاء المتكرر (أسلوب المعاودة) .

فنجد التالي:

1 – إذا كانت n = 0 فأن x = 1 .

2 – إذا كانت n < 0 فإن 1 / x^n .

3 – إذا كانت n > 0 فإن x * x^n-1 .

ولذلك عندما نكتب الدالة بأسلوب المعاودة ننظر إلى الحالات في الحالة الأولى يكفي أن نكتب التالي

```
if( n == 0 )
        return 1;
```

أما في الحالة الثانية فيكفي أن نكتب التالي:

```
else if( n < 0 )
return 1 / Power( X,abs(n) );
```

و استخدمنا هنا أسلوب المعاودة حيث يتم استدعاء الدالة مرة أخرى متى ما كان العدد المرسل للدالة سالب (مع ملاحظة أنه في المقام ولذلك نأخذ له القيمة المطلقة لكي يكون عدد موجب – كما في علم الرياضيات -).

أما في الحالة الثالثة فيكفي أن تكتب التالي:

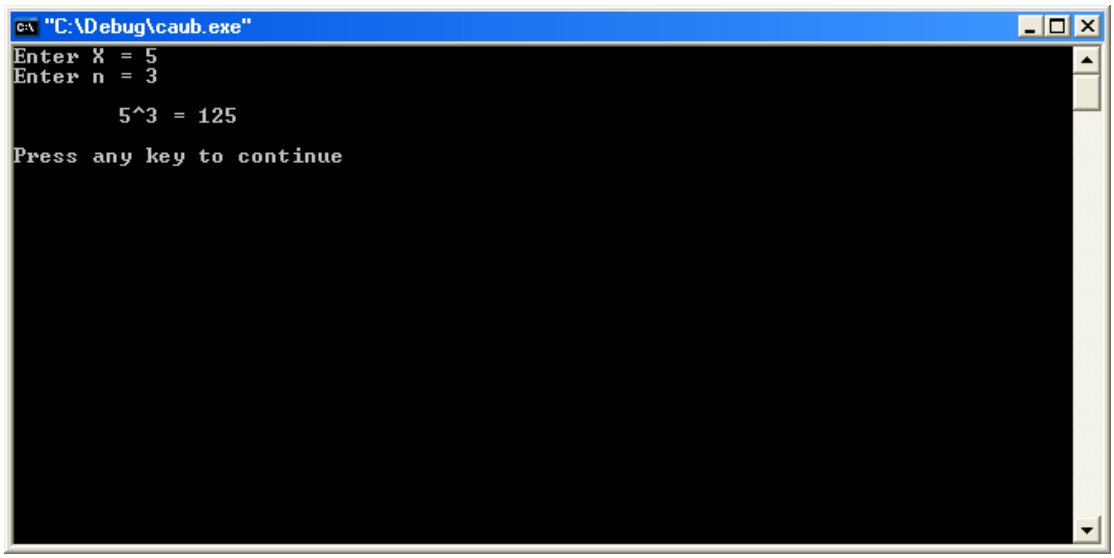
```
else
return  X * Power( X,n-1 );
```

هنا نقوم بضرب العدد المرسل لدالة بالقيمة المعادة من الدالة بعد إنقاص الأس بمقدار واحد وهكذا حتى يصل إلى شرط ينهي الدالة المستدعاة وذلك عندما يتحقق الشرط الأول فتعود بالقيمة واحد التي يتم ضربها بالعدد الممرر للدالة والعودة بالناتج إلى الدالة الرئيسية (main) وطباعة الناتج.

البرنامج

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

/*************************************/
/*** compute power(X,n) , X^n *******/
/*************************************/
```

```
double Power( double X, int n );


int main()
{
	double x;
	int n;

	cout << "Enter X = ";
	cin  >> x;
	cout << "Enter n = ";
	cin  >> n;

	cout << "\n\t" << x << "^" << n << " = " << Power( x,n )
<< "\n\n";

	return 0;
}


double Power( double X, int n )
{
	if( n == 0 )
		return 1;
	else if( n < 0 )
		return 1 / Power( X,abs(n) );
	else
		return X * Power( X,n-1 );
}
```



شرح البرنامج :
في البداية تم الإعلان عن الدالة Power( ) والتي تأخذ وسيطين هم العدد double والأس الصحيح int وتعود بالقيمة من النوع double .
وفي الدالة main() يطلب من المستخدم إدخال العدد والأس ثم يتم تمرير القيم للدالة واستدعائها لطباعة ناتج رفع العدد للأس.
وفي جسم الدالة تم شرحه في الأعلى فلداعي لأعادته.

*شرح الكلمة المحجوزة inline :
تأتي هذه الكلمة مع الدوال لتعمل كتلميح للمترجم فكأنها تقول " هذه دالة مدرجة دالة بسيطة لاتعمل لها استدعاء أيه المترجم".
فيقوم المترجم بعدم عمل استدعاء للدالة وقد يتجاهل التلميح ويعمل استدعاء ولكن ما يحدث هو أن المترجم لا يعمل استدعاء بل كأن تعليمات الدالة الموجود في جسمها كتبت في المكان الذي استدعيت منه هذه الدالة.

-أي أقصد أن المكان الذي نستدعي فيه الدالة كأننا كتبنا هنا جسم الدالة.
أي كأن المترجم قام بنسخ الأوامر الموجودة في جسم الدالة وكتبها في مكان استدعاء الدالة.

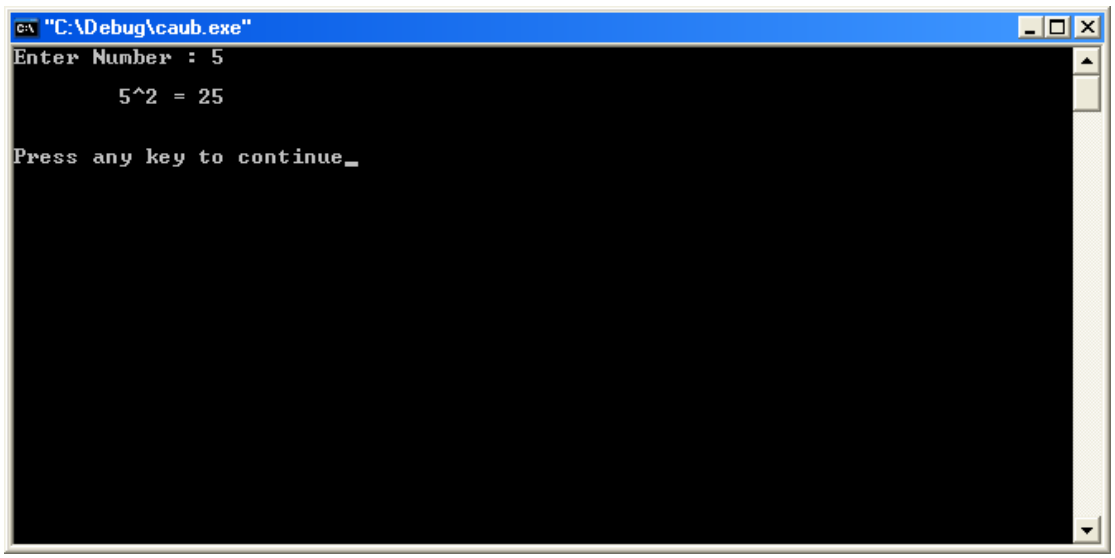
-متى تستخدم الكلمة المحجوزة inline :
ليس هنالك قاعدة عامة يمكن أن نسير عليها ولكن نقول كلما كانت الدالة صغيرة وبسيطة وتحتوي على عدد قليل جدا من التعليمات فيفضل استخدام الكلمة المحجوزة inline .

-كيف يتم استخدامها:

تكتب الكلمة المحجوزة inline قبل نوع خرج الدالة دائما لتدل على أن هذا إجراء مدرج.

-مثال : سنستخدم مثال تكرر كثيرا وهو مربع عدد فدالة مربع عدد تحتوي على عدد على جملة واحدة فقط فمن الأفضل كتابته كإجراء مدرج لكي نستفيد من خاصية الكلمة المحجوزة inline :

```
#include <iostream.h>


inline int DoubleNumber( int n );

int main()
{
	int N;


	cout << "Enter Number : ";
	cin  >> N;
	cout << "\n\t" << N << "^2 = " <<  DoubleNumber( N );
	cout << "\n\n\n";

	return 0;
}


inline int DoubleNumber( int n )
{

	return n * n;
}
```



شرح البرنامج تم الإعلان عن الدالة DoubleNumber(int n) أنها دالة مدرجة باستخدام الكلمة المحجوزة inline ثم نوع خرجها وهو int ويتم تمرير متغير صحيح لهذه الدالة.ثم تقوم بحساب مربع عدد وإعادة الناتج إلى المكان الذي استدعيت عنده.
هنا ما يحدث هو عند الاستدعاء كأننا نسخنا الأمر الموجود في جسم الدالة وكتبناه مكان استدعاء الدالة أي لا يحدث استدعاء فعلي للدالة.

*الخاتمة:
الحمد لله الذي سهل علينا هذا العمل شرحا وتمارينا وكتابة وأسأله سبحانه أن يجعل هذا العمل خالصا لوجه الكريم.
كما أن هذا هو جهد المقل فمكان من صواب فالفضل لله سبحانه وتعالى ومكان من خطأ فم نفسي والشيطان .
لاتنسونا من دعائكم في ظهر الغيب.

كتبه:
السهم الناري (أخوكم خالد).