lesson 14 Functions

لقد ذكرنا عندما درسنا أمر printf أن هذا الأمر هو إختصار لكلمة print function .
فى هذا الدرس نريد التعرف على معنى ال functions و لماذا نستخدمهم ؟

ال function :

عبارة عن كتلة من التعليمات البرمجية التى تؤدي مهمة محددة
مثال على ال function :

```
#include <stdio.h>
void greeting( ) {
  printf("Hello");
}
int main(  ) {
  greeting(  );
 return 0;
}
```

(قم بتجربة الكود بنفسك واضغط هنا)

الناتج : Hello

فى البرمجة أحياناً نستخدم مجموعة من الأكواد أكثر من مرة، إذا كان لدينا مثلاً 200 سطر برمجى ونريد استخدامه أكثر من مرة سنقوم بعمل function
ونقوم بتسميتها باسم يعبر عن وظيفتها وعند الحاجة اليها نقوم بكتابة اسمها ليتم استدعاؤها.

يتم تعريف ال function خارج ال main و يكون لها شكل معين مكون من 4 مكونات رئيسية :

1. نوع ال function
2. الاستدعاء (call)
3. ال return

4. ال arguments & parameters

لنأخذ مثال أكثر تفصيلا لشرح المكونات عليه .
لنفترض أننا نريد عمل function يتم إرسال رقمين إليها و وظيفتها أن تقوم بإرجاع مجموعهم، يمكن كتابتها على النحو التالى :

```
#include <stdio.h>
int add ( int x , int y ) {
  return x+y ;
}

int main(  ) {
 int x,y;
 scanf("%d%d",&x,&y);
 int total = add( x ,y );
 printf(" total = %d \n",total);
 return 0;
}
```

أولا نوع ال function :

فى هذا المثال (int add) هى function من نوع int، نعم ال functions تأخذ نفس أنواع ال data types مثل المتغيرات، و لكن لماذا ؟ هل يتم تخزين بيانات فيها مثل المتغيرات ؟

لا ، نوع ال function ليس للتخزين ولكن هو متعلق برقم 3 فى مكونات ال function و هو ال return حيث يجب أن تقوم ال function بعمل return لقيمة معينة من نفس نوعها .

ثانيا الاستدعاء (call) :
لكى يبدأ البرنامج فى تنفيذ ال function يجب أن يتم إستدعاؤها، هنا تم إستدعاء ال function add داخل ال main حيث سيتم تبديل اسم ال function الذى قام باستدعاؤها و يحل مكانه القيمة النهائية لها و هى ما تم عمل return له.
إذا ما هو ال return ؟

ثالثا الإرجاع (return):
عند إستدعاء ال function عن طريق كتابة اسمها فى مكان ما سيمر عليه البرنامج أثناء تنفيذه، يذهب البرنامج إلى هذه ال function ليبدأ تنفيذ الأوامر بداخلها و ينتظر أن يتم إرجاع قيمة للمكان الذى تم استدعاء ال function فيه، هذه القيمة يجب ان تكون من نفس نوع ال function، لو كانت ال function من نوع int يتم عمل return لبيانات من نوع int، لو كانت ال function من نوع float يتم عمل return لبيانات من نوع float .. و هكذا

هناك نوع واحد من البيانات الذى لا يحتاج إلى return و هو ال void.
ال **void** معناه أنه مكان فارخ لا يعبر عن أى بيانات، و نستخدم هذا النوع إذا كانت ال function ستقوم بمهمة محددة بدون إرجاع أى بيانات مثل مثال طباعة كلمة Hello .

إذا فى المثال السابق كانت ال function تقوم بعمل return لحاصل جمع ال x و y و حاصل الجمع ذلك سيكون قيمة من نوع int و هو نفس نوع ال function و سيتم إستقباله فى ال main على أنه int فلن تظهر مشاكل فى الكود .
و لكن كيف تم إرسال قيم ال x , y داخل ال function ؟

رابعا ال arguments & parameters :

لكى يتم تمرير البيانات إلى ال function عند إستدعائها يتم كتابة البيانات فى الأقواس الخاصة بالاستدعاء

```
total = add ( x , y )
```

من الممكن كتابة

```
total = add ( 5 , 9 )
```

أو أى أرقام أخرى، هذه الأرقام يتم إرسالها إلى ال function و يسموا بال arguments و تكون ال function مستعدة لإستقبالهم

```
int add ( int x , int y )
```

فنجد أن ال function تستقبل متغيرين من نوع int و هما ما أرسلناهم عند الاستدعاء، و هنا int x و int y يسموا بال parameters أى أن ما موجود داخل أقواس ال function عند تعريفها هو ما يحدد نوع و عدد البيانات التى يجب إرسالها إلى ال function لكى تعمل .

ملحوظة هامة : ال x و y داخل ال function هم متغيرين مختلفين تماما عن ال x و y داخل ال main الذان قمنا بعمل scanf لهما .

أى أن ال scope الخاص بال function i, هو scope منفصل تماما و مستقل بنفسه عن باقى البرنامج و المتغيرات فيه مستقلة بذاتها حتى و لو حملت نفس الاسماء من متغيرات خارج ال scope.

كان يمكن أن يكون الكود على هذا الشكل :

```
#include <stdio.h>
int add ( int num1 , int num2 ) {
  return num1 + num2 ;
}

int main(  ) {
```

```
 int x,y;
 scanf("%d%d",&x,&y);
 int total = add( x ,y );
 printf(" total = %d \n",total);
 return 0;
}
```

أى أن num1 و num2 سوف يأخذوا نفس قيم ال x و y و لكنهما متغيران مختلفان تماما عنهما .

ملحوظة : ليس من الضرورة أن يتم إرسال بيانات أو استقبال بيانات فى ال functions و لكن هذا على حسب البرنامج و الغرض منه، و لكن من الضرورى كتابة ال function على نفس الهيئة بنفس الأقواس حتى لو كانت فارغة سواء فى الإنشاء أو الإستدعاء.

تكلمنا عن تفاصيل ال functions و مكوناتها الرئيسية و لكن لنتعمق أكثر علينا أن نعرف أن هناك نوعان من ال **function** :

النوع الأول هو **Standard library functions** و هي ال functions الجاهزة للاستخدام بمجرد استدعائها مثل : **printf ( ) - scanf ( )**
هي functions جاهزة للاستخدام بمجرد استدعائها وتعريف المكتبة التي تحتوي عليها ، وهنا المكتبة هى **stdio.h** الخاصة بأوامر الطباعة و الأدخال

النوع الثاني نوع تم إنشائه من المستخدم مثال function للترحيب

```
#include <stdio.h>
void fun(  ) {
  printf("Hello");
  //هنا قمنا بتعريفها
 }
```

```
int main( ) {
  fun( );
  //هنا قمنا بالاستدعاء للتنفيذ وكلما احتجنا إليها قمنا بكتابتها مرة أخرى واستدعائها
}
```

تعتبر main هى أيضا function لكنها الرئيسية الذى يدخل اليها الكمبيوتر ويقوم بتنفيذ كل الاوامر الموجودة بها بالترتيب .