lesson 74 Practice 17 (Functions)

فى هذا الدرس سنتعرف على وظيفة **return** فى **function**

تعنى **return 0;** أن تتوقف **function** أو إذا كان داخل **main** أن يتوقف البرنامج عن العمل

لكن ما وظيفة **return** داخل **function**

return فى **function** تقوم بإرجاع القيمة الموجودة بعد **return** فى نفس المكان الموجود فيه **function** فى **main**

تقوم بإرجاع القيمة إلى المكان الذى قمت باستدعاء **function** فيه

عند استدعاء **function** فى اى مكان تعنى أن تجعل قيمة **function** تساوى القيمة الموجودة بعد **return**

مثال :

```
#include <stdio.h>
int fun() {
  return 5;
}

int main() {
  int x= fun();
  // fun(); قم بحجز متغير يساوى
  // 5 نذهب إلى تنفيذ fun نجدها أنها تقوم بإرجاع
  printf("%d", x);
  // 5 هنا أصبحت تساوى
}
```

output:

5

(قم بتجربة الكود بنفسك واضغط هنا)

مثال أخر :

```
#include <stdio.h>
int fun() {
  return 5;
}
//أجعل قيمة fun عند استدعائها فى اى مكان تساوى 5
int main() {
  printf("%d", fun());
  // هنا أصبحت تساوى 5
}
output:
5
```

(قم بتجربة الكود بنفسك واضغط هنا)

إذاً **return** فى **function** تقوم بإرجاع القيمة الموجودة بعد **return** فى نفس المكان الموجود فيه **function** فى **main**

- إذاً **int function :** أنه لابد من وجود **return** لرقم، أى أنها تقوم بإرجاع رقم

- تعنى **void function** : أنها لا ترجع شئ،لسنا بحاجة إلى وضع **return**، لكن إذا كتبت سيتم كتابتها كالتالى **return ;**

فى **function** نستطيع تعريف متغيرات بين أقواس

```
#include <stdio.h>
int fun(int a, int b) {
  // قمنا بتعريفهم بين القوسين لأننا سوف نستبدلهم بالقيم الموجودة
```

```
  //main الموجودة فى  fun بين قوسين استدعاء
  return a + b;
}

int main() {
  int x=3, y=5;

  int z = fun(x, y);
  //x,y  وقم بوضع قيم المتغيرين الموجودة فى  fun قم بالذهاب إلى
  // fun بدلاً من قيم المتغيرين الموجودين فى
  printf("%d", z);
}
```

output:

8

(قم بتجربة الكود بنفسك واضغط هنا)

لكن لابد عند وضع متغيرات عند استدعاء function أن يكون نفس نوع data type المعرف داخل أقواس function

مثال :

```
#include <stdio.h>
int fun(int a, int b) {
  //الموجود هنا بين الأقواس وعدد المتغيرات data type  نوع
  //function عند استدعاء  data type أن يكون هو نفس العدد ونوع
  // وإدخال لها قيم
  return a + b;
}
int main() {
  int x=3, y=5;
```

```
  int z = fun(x, y);
  //function لابد أن يكون عدد المتغيرات هنا هو عدد المتغيرات الموجودة فى تعريف
  //data type لابد أن يكون نفس نوع
  printf("%d", z);
}
```

(قم بتجربة الكود بنفسك واضغط هنا)

ملحوظة :

إذا تشابه اسماء المتغيرات فى **main** و **function** فهذا ليس معناه انهم نفس المتغير، لكن هذا متغير له مكان يعمل به سواء كان داخل **function** أو سواء كان داخل **main** كل متغير له مجال scope خاص به وليس له علاقة بالآخر ، لكن إذا تم تعريفهم فى **Global Variable** يكون معرف لكل **funcation**

مثال أخر :

```
#include <stdio.h>
// ترتيب العمليات التي تحدث
int fun(int y) {
  //3-// y نقوم باستبدال قيمة x بدل من قيمة
  if (y > 0)
  //4-// 0 هل قيمة y حالياً أكبر من
  //  3 تساوى main نعم لأن قيمة x المستدعاة من
    return 10;
  //5-// 10 نقوم بإرجاع
  return-10;
}
int main() {
  int x = 3;
```

```
  int z = fun(x);
  //1-// fun(x); تساوى z هنا
  //2-// x محملة بقيمة  fun(int y) نذهب إلى
  printf("%d", z);
}
output:
10
```

(قم بتجربة الكود بنفسك واضغط هنا)

مثال أخر:

```
#include <stdio.h>
// ترتيب العمليات التي تحدث
int fun(int y) {
  //3-// y بدل من قيمة x نقوم باستبدال قيمة
  if (y > 0)
  //4-// 0 حالياً أكبر من y هل قيمة
  //  0 تساوى main المستدعاة من  x لا لأن قيمة
    return 10;
  return-10;
  //5-// -10 نقوم بإرجاع

}
int main() {
  int x = 0;
  int z = fun(x);
  //1-// fun(x); تساوى z هنا
  //2-// x محملة بقيمة  fun(int y) نذهب إلى
```

```
  printf("%d", z);
}
output:
-10
```

(قم بتجربة الكود بنفسك واضغط هنا)

لذلك **function** تقوم بعمل **return** لرقم واحد فقط

وعندما تقوم بعمل إرجاع **return** سيضع الرقم الموجود جانب **return** فقط مكان كلمة **function** المستدعاة فيه