```
        r1 = 1;           // output another value through r1
    }

    // since r2 is inout we can use it as an input value
    // and also output a value through it
    r2 = r2 * r2 * r2;

    return true;
}
```

إن هذا التابع مماثل تقريباً لأي تابع بلغة C++ فيما عدا وجود الكلمات المحجوزة in وout وinout.

- **in**: يعني أنه ستنسخ قيمة الوسيط الفعلي إلى الوسيط الشكلي قبل بدء تنفيذ التابع. ليس من الضروري ذكر in قبل اسم الوسيط صراحة لأن الوسيط هو in افتراضياً. مثلاً، إن كلا التابعين التاليين متكافئين:

```
float square(in float x)
{
    return x * x;
}
```

(بدون ذكر in بشكل صريح):

```
float square(float x)
{
    return x * x;
}
```

- **out**: يعني أنه سيُنسخ الوسيط الشكلي إلى الوسيط الفعلي عندما يعود التابع. يفيدنا هذا الأمر من أجل إعادة القيم من خلال الوسطاء. إن الكلمة المحجوزة out ضرورية لأن لغة HLSL لا تسمح بتمرير الوسطاء مرجعياً كما لا تسمح بتمرير المؤشرات. نلاحظ أنه لما يكون الوسيط المحدد على أنه out، لا يُنسخ الوسيط الفعلي إلى الوسيط الشكلي قبل بدء التابع، وبالتالي لا يمكننا افتراض وجود قيمة أولية فيه ذات معنى قبل أن نقوم بإسناد قيمة ما إليه. بكلام آخر: يستخدم وسيط ذو out فقط من أجل إخراج المعطيات ولا يستخدم أبداً كدخل.

```
void square(in float x, out float y)
{
    y = x * x;
}
```

هنا ندخل العدد الذي نريد تربيعه من خلال الوسيط x ونأخذ القيمة المعادة (مربع x) من خلال الوسيط y.