جامعة حماه

المعهد التقاني للحاسوب

السنة الأولى

برمجة ٢

محاضرة٣

قسم البرمجيات

عملي

إعداد:

م. رفا البنات

م. أريج فياض

مفاهيم أساسية:

- **يتضمن التعريف أو التصريح عن التوابع مايلي :**

١- كلمتان مفتاحيتان هما void و static حيث كلمة void تشير الى أن التابع لا يعيد أي قيمة أما كلمة static فتشير الى مفاهيم متعلقة بالبرمجة غرضية التوجه سنتحدث عنها لاحقا.

٣- نمط الإعادة احد الأنماط(int ,string ,float,double.....)في حال كان التابع يعيد قيمة ولم نستخدم void.

٤- اسم التابع متبوع بقوسين من الشكل () وبداخل القوسين وسطاء التابع مع أنماطهم .

٥- الكود الخاص بالتابع محصور بين قوسين من الشكل {}، ونسميه جسم التابع.

- **القيم المعادة :**

إذا أردنا للتابع ان يعيد قيمة نقوم بما يلي :

١- تحديد نوع القيم المعادة وذلك ضمن التصريح عن التابع كما ذكرنا في الخطوة ٣ سابقاً.

٢- استخدام كلمة return لإنهاء تنفيذ التابع ونقل القيمة المعادة منه(إذا كان يعيد قيمة ولم نستخدم void)الى الكود الذي تم فيه استدعاء هذا التابع.

- **الوسطاء (البارمترات):**

لتزويد التابع بالوسطاء علينا أن نحدد ما يلي :

١-لائحة الوسطاء التي يتضمنها التابع أثناء التصريح بالإضافة إلى نوع كل وسيط على حدى ونفصل بينها بفاصلة عادية .

٢- لائحة وسطاء مطابقة للائحة في التصريح وذلك عند استدعاء التابع حيث نفصل بين الوسطاء بفاصلة ولكن بدون تحديد نوع الوسيط .

ملاحظات:

١- عند استدعاء تابع له وسطاء يجب أن تكون الوسطاء معرفة من قبل أن يتم الاستدعاء.

مثال :صرحنا عن x و y قبل أن نمررهم إلى التابع وليكن اسمه max

```
int x = int.Parse(Console.ReadLine());

int y = int.Parse(Console.ReadLine());

int z = max(x, y);
```

٢- نمط المعطيات void يعني أن التابع لايعيد أي قيمة ولا تجتمع return مع void .

٣- نكتب التابع خارج دالة ال main ويمكن كتابة أكثرمن تابع .

٤- يمكن أن يتم استدعاء التابع في توابع أخرى سواء تابع ال main او توابع أخرى قمنا بكتابتها نحن.

- **استدعاء تابع ضمن تابع:**

- **اكتب تابع يقوم باختبار عددين مدخلين اذا كانا صديقين أم لا.**

مع العلم أن : إذا كان العددين صديقين فيجب أن يكون مجموع قواسم العدد الأول تساوي العدد الثاني و مجموع قواسم العدد الثاني تساوي العدد الأول .

هنا نحتاج تابعين ، التابع الأول : يقوم بإيجاد مجموع قواسم عدد

التابع الثاني : يقوم باختبار العددين وتحديد إن كانا صديقين أم لا .

التابع الأول:

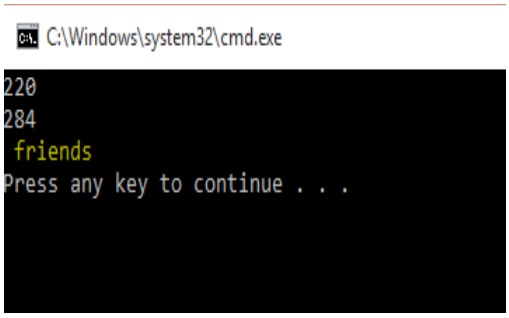
```
static int sum(int x)
        {
            int s = 0;
            for (int i = 1; i < x; i++)
            {
                if (x % i == 0)
                { s = s + i; }
            } return s;
        }
```

التابع الثاني:

```
static void friends(int x, int y)
        {
            if (sum(x) == y && sum(y) == x)
            {
             Console.WriteLine("friends");
             }
            else
              Console.WriteLine("not friends");
        }
```

نقوم باستدعاء تابع الfriendsضمن ال main

```
static void Main(string[] args)
        {
    int a = int.Parse(Console.ReadLine());
    int b = int.Parse(Console.ReadLine());
      friends(a, b);
      Console.ReadKey();
        }
```



- **تطابق الوسطاء (البارمترات):**

عند استدعاء تابع فإنه علينا مطابقة البارمترات المحددة في سطر التصريح عن التابع مع البارمترات الممررة إليه ضمن سطر استدعاء التابع وبالتالي إذا كان لدينا تابع من الشكل :

Void my_function (string name, double number) {}

فإنه لا يمكننا استدعاء هذا التابع بالشكل التالي ونمرر له الوسيط الأول رقمي وهو بالحقيقة نمطه string ومررنا الوسيط الثاني string علما ان هو double. وهذا غير مطابق لسطر التصريح.

My_function(255,"hi");

- **طرق تمرير الوسطاء للتابع :**

١- تمرير الوسطاء بالقيمة:

إن جميع التوابع التي تعرفنا عليها لحد الان قمنا بتمرير الوسطاء اليها بالقيمة وهذا يعني أن استخدام هذا الوسيط في التابع والتعديل عليه لن يؤثرعلى القيمة الفعلية الموجودة في ال main لهذا البارمتر .

مثال 1 :

```
static void show(int val)
{
    val = val * 2; // قمنا بمضاعفة العدد الممر الى التابع
    Console.WriteLine(val);
}

static void Main(string[] args)
{
    int my_number = 5;
    Console.WriteLine(my_number);// سوف يطبع ٥
    show(my_number); // سوف يستدعي التابع على القيمة الممرة اليه التي تساوي ٥ ويضاعفها
    Console.WriteLine(my_number);// سوف يطبع ٥

}
```

نلاحظ هنا اننا قمنا بتمرير القيمة ٥ الى التابع وضاعفها وأصبحت ١٠ لكننا عند طباعتها مرة ثانية تظهر ٥ وليس ١٠ لأننا هنا قمنا بتمرير الوسيط بالقيمة أي قمنا بتمرير نسخة عن المتحول الى التابع وليس المتحول نفسه واي تغيير في قيمة هذا المتحول ضمن التابع لن يؤثر على قيمته الفعلية المخزنة بالذاكرة .

مثال 2:

```
{
    static int sum(int x, int y) {
        int z = x + y;
        x = 20;
        return z;
    }
    static void Main(string[] args)
    {
        int a = 30;
        int b = 40;
        Console.WriteLine(a);//30 ستطبع
        Console.WriteLine(sum(a, b));// 70طبع
        Console.WriteLine(a); //  ٣٠  ستطبع
        Console.WriteLine(sum(a,b));// 70
        Console.ReadKey();
    }
}
```

```
C:\Windows\system32\cmd.exe
30
70
30
70
Press any key to continue . . .
```

٢- التمرير بالمرجع :

هنا لن يتم تمرير نسخة عن المتحول وإنما سيتم تمرير المتحول نفسه مباشرة وأي تغيير عليه ضمن التابع سيؤدي إلى تغيير قيمته الفعلية المخزنة في الذاكرة.

وهذا مايقصد بالتمرير بالمرجع أو العنوان يعني أنه سيتم تمرير عنوان المتحول الى التابع والتغيير عليه حكما سيؤدي الى تغييره فعليا في الذاكرة لأنه يعلم العنوان.

مثال:

```
static void show( ref int val)
{
    val = val * 2; // قمنا بمضاعفة العدد الممرر الى التابع
    Console.WriteLine(val);
}

static void Main(string[] args)
{
    int my_number = 5;
    Console.WriteLine( my_number);//  سوف يطبع٥
    show(ref my_number); // سوف يستدعي التابع على القيمة الممرة اليه بالمرجع التي تساوي ٥ ويضاعفها
    Console.WriteLine(my_number);// سوف يطبع ١٠

}
```

نلاحظ هنا انه تمت تعديل قيمة المتغيير my_number الى ١٠ بعد استدعاء التابع show وذلك لأننا مررنا الى التابع عنوان المتغير my_number .

مثال :

```
    static int sum( ref int x, ref int y) {
        int z = x + y;
        x = 20;
        return z;
    }
    static void Main(string[] args)
    {
        int a = 30;
        int b = 40;
        Console.WriteLine(a);//30 ستطبع
        Console.WriteLine(sum(ref a, ref b));// 70ستطبع
        Console.WriteLine(a); //  ٢٠  ستطبع
        Console.WriteLine(sum(ref a, ref b));// 60
        Console.ReadKey();
    }
}
```

```
file:///c:/users/hp/docun
30
70
20
60
```

ملاحظات:

١ * لكي نتمكن من التمرير بالمرجع او العنوان يجب ان نضع الكلمة المحجوزة ref لكي يفهم انه متحول ممر بالمرجع في سطر التصريح ونضعها أيضا في سطر الاستدعاء.

٢* لا يمكن أن نمرر متحول ثابت(const) بالمرجع .

٣* يجب علينا ان نعطي قيمة ابتدائية للمتحول المرجعي قبل تمرريه الى التابع.

٣-تمرير الوسطاء كوسطاء خرج :

عندما نريد من تابع ان يعيد لنا قيمتين لن نتمكن من ان نكتب به return مرتين لذلك عند التصريح عن التابع سنمرر له مع وسطائه وسيطين للخرج لكي يضع لنا القيم التي نريد ان يعيدها التابع وعند الاستدعاء نمررهم أيضا كوسيطي خرج.

مثال:

- اكتب تابع لجمع وطرح عددين.

```
static void process(int x, int y,out int sum,out int sub)
        {
            sum = x + y;
            sub = x - y;
        }//method
        static void Main(string[] args)
        {
            Console.WriteLine("enter the first number:");
            int x = int.Parse(Console.ReadLine());
            Console.WriteLine("enter the second number:");
            int y = int.Parse(Console.ReadLine());
            int sum, sub;
            process(x, y,out sum, out sub);
            Console.Write("sum=");
            Console.WriteLine(sum);
            Console.Write("sub=");
            Console.WriteLine(sub);
            Console.ReadKey();
        }//main
    }
```

```
enter the first number:
10
enter the second number:
20
sum=30
sub=-10
```

نلاحظ عند التصريح صرحنا عن متحولين إضافيين للخرج لكي يعيد لنا ناتج الجمع وناتج الطرح وجعل التابع من نمط void.

وقبل استدعاء التابع قمنا فقط بالتصريح عن متحولين وتمرريهم للتابع process كمتحولات خرج ويعيدوا لنا ناتج الجمع وناتج الطرح.