كلية العلوم

القسم : **الرياضيات**

السنة : **الثانية**

المادة : لغات البرمجة٢

المحاضرة : الرابعة/نظري/

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

# التوابع (الدوال) Functions

## المقدمة Introduction

Divide and conquer

( في البرمجة نستخدم مبدأ "فرق تسد")

– Construct a program from smaller pieces or components

نبني البرنامج من أقسام و أجزاء صغيرة

– Each piece more manageable than the original program

كل قسم (جزء) يمكن إدارته بشكل أفضل ويمكن صيانته وتعديله

كل قسم يمكن استخدامه كمكون جاهز في أكثر من مكان وإعطاءه مهمة للقيام بها

## مكتبة التوابع الرياضية
## Math Library Functions

- Perform common mathematical calculations

لانجاز الحسابات الرياضية العامة(القياسية) نستدعي الملف الرأسي <math.h>

– Include the header file `<math.h>`

- Functions called by writing( يُستدعى التابع بكتابة اسمه والوسطاء)
  - functionName (argument);

or

  - functionName(argument1, argument2, …);
- Example

```
cout << sqrt( 900.0 );
```

  - sqrt (square root) function The preceding statement would print 30
  - All functions in math library return a `double`

كل التوابع الرياضية بشكل عام تعيد قيمة من النوع double

## أمثلة عن التوابع الرياضية

| Method | Description | Example |
|---|---|---|
| `ceil( x )` | rounds *x* to the smallest integer not less than *x* | `ceil( 9.2 )` is 10.0<br>`ceil( -9.8 )` is -9.0 |
| `cos( x )` | trigonometric cosine of *x* (*x* in radians) | `cos( 0.0 )` is 1.0 |
| `exp( x )` | exponential function *ex* | `exp( 1.0 )` is 2.71828<br>`exp( 2.0 )` is 7.38906 |
| `fabs(f x )` | absolute value of *x* | `fabs( 5.1 )` is 5.1<br>`fabs( 0.0 )` is 0.0<br>`fabs( -8.76 )` is 8.76 |
| `loor( x )` | rounds *x* to the largest integer not greater than *x* | `floor( 9.2 )` is 9.0<br>`floor( -9.8 )` is -10.0 |
| `fmod( x, y )` | remainder of *x*/*y* as a floating-point number | `fmod( 13.657, 2.333 )` is 1.992 |
| `log( x )` | natural logarithm of *x* (base *e*) | `log( 2.718282 )` is 1.0<br>`log( 7.389056 )` is 2.0 |
| `log10( x )` | logarithm of *x* (base 10) | `log10( 10.0 )` is 1.0<br>`log10( 100.0 )` is 2.0 |
| `pow( x, y )` | *x* raised to power *y* (*xy*) | `pow( 2, 7 )` is 128<br>`pow( 9, .5 )` is 3 |
| `sin( x )` | trigonometric sine of *x* (*x* in radians) | `sin( 0.0 )` is 0 |
| `sqrt( x )` | square root of *x* | `sqrt( 900.0 )` is 30.0<br>`sqrt( 9.0 )` is 3.0 |
| `tan( x )` | trigonometric tangent of *x* (*x* in radians) | `tan( 0.0 )` is 0 |

## Math Library Functions

- Function arguments can be

  الوسيط يمكن أن يكون من النوع:

  - Constants ثابت
    - `sqrt( 4 );`
  - Variables متحول
    - `sqrt( x );`
  - Expressions تعبير
    - `sqrt( 3 + 6x );`
    - `sqrt( sqrt( x ) ) ;`

# Functions

- أهمية التوابع Functions
  - جعل البرنامج قياسياً وقابلاً للمراجعة Modularize a program
  - إعادة استخدام البرمجيات Software reusability
    - نستدعي التابع أكثر من مرة ومن أماكن مختلفة Call function multiple times
- المتحولات المحلية Local variables
  - Known only in the function in which they are defined
  - All variables declared in function definitions are local variables

    كل المتحولات التي يُصرح عنها في جسم التابع تُعتبر محلية وخاصة بالتابع
- Parameters
  - Local variables passed to function when called

  Provide outside information

  يتم تمرير المتحولات المحلية عند مناداة التابع وتزود خارجياً بالمعلومات



التوابع :

- التابع هو عبارة عن وحدة مستقلة من كود البرنامج المصممة لانجاز مهمة محددة.
- يتم كتابة برامج C++ بأن يتم الجمع بين نوعين من التوابع :
  1- التوابع المكتوبة من قبل المبرمج
  2- التوابع الجاهزة المتوفرة في المكتبة المعيارية للغة C++
- يمكن أن يقوم التابع بأحد المهمتين:
  أ- حساب/ إرجاع قيمة معينة مثال التابع المسؤول عن إدخال قيمة،جمع عددين، القيمة الصغرى ، القيمة الكبرى.
  ب- القيام بعمل ما دون حساب أي قيمة
- كل تابع له :
  أ- اسم

  اسم اختياري للتابع مراعياً مجموعة من القواعد كأن ألّا يبدأ برقم وألّا يحوي على فراغ...

  ب- وسطاء

  وهي عبارة عن دخل هذا التابع أي المتحولات التي سيُطبق عليها عمليات لتنفيذ مهمته.

  لكل وسيط :
  - نمط
  - اسم
  - قيمة ابتدائية ( اختياري )

  ت- نمط

  وهو عبارة عن نمط القيمة المعادة من هذا التابع

  ث- جسم التابع

  وهي عبارة عن مجموعة التعليمات التي تقوم بالمهمة المسندة للتابع مجموعة مابين القوسين {}

## التوابع المكتوبة من قبل المبرمج

لكتابة برنامج يحوي على توابع معرفة من قبل المبرمج يجب القيام بما يلي :

1- التصريح عن تابع function declaration:

- أي إخبار المترجم بنموذج التابع (prototype) حيث يتضمن النموذج نمط التابع، اسمه ، وسطاءه دون ذكر جسم التابع
- يتم ذكر نموذج التابع في أي مكان من البرنامج ولكن قبل استخدام التابع مثلاً قبل التابع ()main أو في منطقة التصريح عن المتحولات ضمن ()main.

ويتم ذلك عن طريق ذكر prototype الخاص بالتابع يأخذ الشكل العام التالي:

```
return_value_type fun_name(para_type para_name , para_type  para_name,... );
```

حيث :

return_value_typ : نمط القيمة المعادة من التابع

- في حال كان التابع يحسب/ يرجع قيمة يتم ذكر نمط هذه القيمة
- وفي حال كان التابع فقط يقوم بعمل ما دون إرجاع أي قيمة فيتم ذكر نمط القيمة المعادة على أنه void
- في حال عدم ذكر أي نمط للتابع فيأخذ المترجم النمط الافتراضي هو int

fun_name : اسم التابع

() : مابين () يتم ذكر وسطاء التابع

- para_type هو نمط الوسيط ، para_name هو اسم الوسيط
- يتم الفصل بين كل وسيطين بفاصلة (,)
- ليس بالضرورة ذكر أسماء الوسطاء وإنما يجب ذكر أنماط الوسطاء حتى وإن لم نذكر الاسم

; : الفاصلة المنقوطة للدلالة على أنه نموذج التابع ( دون جسم التابع)

أمثلة :

- نموذج تابع الجمع :

الاسم : sum

الوسطاء : يقوم بإجراء مهمة الجمع على عددين صحيحين وبالتالي لإكمال مهمته يحتاج إلى عددين صحيحين أي بمعنى آخر يحتاج إلى وسيطين كل منهما عدد صحيح

النمط : يقوم بإعادة نتيجة عملية الجمع على عددين صحيحين وبالتالي النتيجة هي عدد صحيح وبالتالي نمط هذا التابع هو عدد صحيح

وبالتالي يكون نموذج تابع الجمع هو :

```
int sum (int num1 , int num2 );
```

- نموذج تابع طباعة عدد صحيح :

الاسم : printNumber

الوسطاء : يقوم بإجراء مهمة الطباعة لعدد صحيح وبالتالي لإكمال مهمته يحتاج إلى عدد صحيح أي بمعنى آخر يحتاج إلى وسيط واحد هو عدد صحيح

النمط : لا يقوم بإعادة أي نتيجة وإنما فقط الطباعة وبالتالي نحدد نمط هذا التابع void أي لايوجد أي قيمة معادة

وبالتالي يكون نموذج تابع الطباعة هو :

```
void printNumber (int num );
```

- نموذج تابع طباعة عبارة "hello world" :

الاسم : print
الوسطاء : يقوم بإجراء مهمة الطباعة لعبارة محدّدة ولا يطبع غيرها وبالتالي لإكمال مهمته لا يحتاج إلى أيّ وسيط
النمط : لا يقوم بإعادة أي نتيجة وإنما فقط الطباعة وبالتالي نحدد نمط هذا التابع void أي لايوجد أي قيمة معادة
وبالتالي يكون نموذج تابع الطباعة هو :

```
void print ( );
```

2- التعريف عن تابع function definition
أي كتابة جسم التابع حيث يتبع البرنامج نفس أسلوب تعريف التابع main أي كما يلي:

```
Return_value_type fun_name(para_type para_name , para_type  para_name,...)
{

//statements

}
```

- يبدأ كتابة نموذج التابع ولكن دون الفاصلة المنقوطة أي ذكر نمط القيمة المعادة ، الاسم ، القوسين () وبينهما وسطاء التابع
- بعدها يتم كتابة قوسين المجموعة {} وضمن هذين القوسين يتم التعريف عن المتحولات المستخدمة والتعليمات التي ستنجز المهمة المسؤول عنها التابع.
- في نهاية جسم التابع نميز حالتين :
  أ- في حال كان التابع بحسب/ يرجع قيمة تكون آخر تعليمة من جسم التابع التعليمة return حيث تسبب هذه التعليمة إعادة قيمة المتحول التالي لها كقيمة معادة من التابع بشرط أن يكون نمط هذا المتحول من نمط التابع. ولا يمر المترجم على أي تعليمة في التابع تأتي بعد تعليمة return.

  ب- أما في حال كان التابع فقط يقوم بعمل ما دون إرجاع أي قيمة يتم الاستغناء عن تعليمة return.
- يتم ذكر تعريف التابع قبل أو بعد استخدام التابع ولكن خارج تعريف أي تابع مثلا قبل أو بعد تعريف التابع main()

أمثلة :

- تعريف تابع الجمع :

```
int sum (int num1 , int num2 ){
    int result = num1 + num2;
    return result;
}
```

- تعريف تابع الطباعة:

```
void printNumber (int num ){
    cout<<num<<"\n";
}
```

- تعريف تابع طباعة عبارة "Hello world":

```
void print()
{
  cout<<"Hello world\n";
}
```

3- استدعاء تابع مصرح عنه calling a function

- أي تنفيذ تعليمات هذا التابع بشكل فعلي ضمن تابع main أو أي تابع آخر
- الشكل العام لتعليمة الاستدعاء:

يتم ذكر اسم التابع والوسطاء التي تحمل القيم الفعلية التي ستتأثر عليها المهمة في حال كان التابع لا يعيد أي قيمة للتابع المستدعي يتم ذكر تعليمة الاستدعاء فقط:

```
fun_name(para1 , para2, ....);
```

أما في حال كان التابع يرجع قيمة يتم إسناد تعليمة الاستدعاء إلى متحول يكون نمطه من نمط التابع ويتم تخزين القيمة المعادة من التابع ضمنه

```
variable = fun_name(para1 , para2, ....);
```

وعندها يقوم البرنامج بالبحث عن هذا التابع واتباع تعليماته وعند الانتهاء من تنفيذ تعليماته يعود إلى السطر التالي لهذا السطر في main()

- يتم الاستدعاء ضمن جسم تابع main() أو جسم أي تابع آخر.
- يمكن استدعاء تابع عدد غير محدد من المرات.

مثال : استدعاء تابع الجمع من أجل قيم مختلفة وفي كل مرة استدعاء تابع الطباعة لطباعة النتيجة.

الاستدعاء الأول
يكون فيه الوسطاء قيم فعلية

```
int result = 0;
 result = sum(6, 100);
 cout<<"Sum = ";
 printNumber(result);
```

الاستدعاء الثاني
يكون فيه الوسطاء متحولات تعتمد قيمها على دخل المستخدم

```
int x = 0;
int y = 0;
int z = 0;

cout<<"Enter num1\n";
cin>> x;

cout<<"Enter num2\n";
cin>> y;

 z = sum(x, y);
 cout<<"Sum = ";
 printNumber(z);
```

الاستدعاء الرابع
يمكن أن يكون فيه وسطاء هي متحولات تعتمد قيمها على استدعاءات لتوابع

```
int x = 0;
int y = 0;
int z = 0;

cout<<"Enter num1\n";
cin>> x;

cout<<"Enter num2\n";
cin>> y;

 z = sum(x, y);
 z = sum(z,10);
 cout<<"Sum += 10 : ";
 printNumber(z);
```

الاستدعاء الخامس :

يمكن أن يكون فيه وسطاء هي عمليات ناتجها من نمط الوسيط المطلوب

```
int x = 0;
int y = 0;
int z = 0;

cout<<"Enter num1\n";
cin>> x;

cout<<"Enter num2\n";
cin>> y;

 z = sum(x+x, y);

 cout<<"Sum += 10 : ";
 printNumber(z);
```

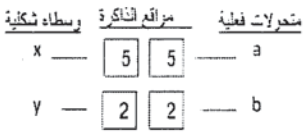


```
6.  //دالة تعيد قيمة هي مجموع عددين صحيحين
7  int sum(int x,int y)          // x,y وسطاء شكلية
8  {                             // مواقع تخزين مؤقتة
9     int s=0;//متحول محلي
10    s=x+y;
11   return s;
12
13
14 }
15 main()
16 {
17    int a,b; //متحولات فعلية
18
19
20             //الإرسال بقيمة
21       تمرير القيم إلى الدالة يكون
22       إما قيم ثابتة
23       أو متحول
24       أو تعبير حسابي
25    */
26
27    cout<<"a=  ";cin>>a;
28    cout<<"b=   ";cin>>b;
29    cout<<"a=22,b=30 "<<sum(22,30)<<endl; //استدعاء الدالة من خلال قيم ثابتة
30    cout<<sum(a,b)<<endl; //استدعاء الدالة من خلال قيم متحولات
31    cout<<sum(a+10,b)<<endl;//استدعاء الدالة من خلال تعبير حسابي
32 }
```

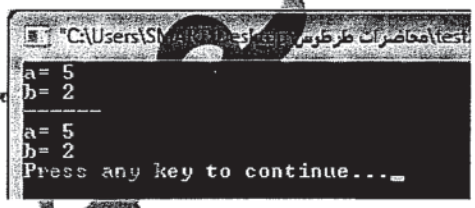
## تمرير وسطاء التوابع بالقيمة وبالعنوان
## References and Reference Parameters


## تمرير المتحولات بالقيمة (الارسال بقيمة)

```
5
6   //في حال لدينا تابع يعيد أكثر من قيمة
7   //دالة تبديل بين عددين//
8
9   void swp(int x ,int y)  //تمرير المتحولات بقيمة
10  {
11
12  int z;
13   z=x;
14   x=y;
15   y=z;
16     //التابع لا يعيد قيمة
17  }
18  main()
19  {
20     int a,b;
21     cout<<"a= ";cin>>a;     //الارسال بقيمة على شكل متحول
22     cout<<"b= ";cin>>b;
23     swp(a,b);
24     cout<<"a= "<<a<<endl; //بعد الاستدعاء
25     cout<<"b= "<<b<<endl;
```

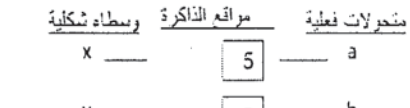


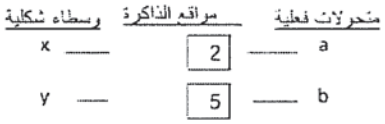


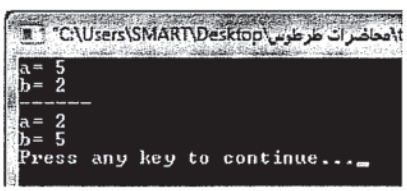



## تمرير المتحولات مرجعيا(الارسال بمرجع او العنوان)

```
6   //في حال لدينا تابع يعيد أكثر من قيمة
7   //دالة تبديل بين عددين//
8
9   void swp(int &x ,int &y)  //تمرير المتحولات مرجعيا
10  {                         //الارسال بمرجع
11
12  int z;
13   z=x;
14   x=y;
15   y=z;
16     //التابع لا يعيد قيمة
17  }
18  main()          //المتحولات الفعلية والوسطاء الشكلية
19                  //تشير إلى نفس الموقع في الذاكرة//
20  {
21     int a,b;
22     cout<<"a= ";cin>>a;
23     cout<<"b= ";cin>>b;
24     swp(a,b);
25     cout<<"------"<<endl;
26     cout<<"a= "<<a<<endl; //طباعة بعد الاستدعاء
27     cout<<"b= "<<b<<endl;
28  }
```



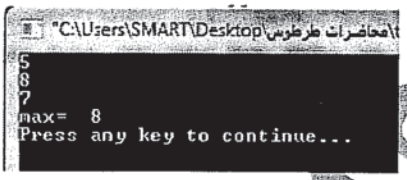




المتحولات الفعلية التي مررت إلى الدالة وكذلك الوسطاء الشكلية تشير إلى مكان واحد ضمن الذاكرة

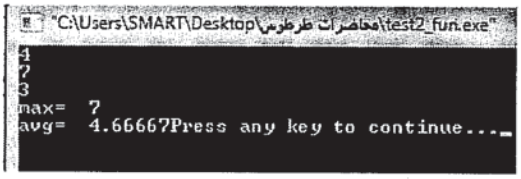
## دالة تعيد القيمة الأكبر لثلاث أعداد

```
64
65 void max(int x,int y,int z,int &m)
66 {
67     //int m;
68     m=x;
69     if (y<x && z<x)
70     m=x;
71      else
72     if (y> x && y>z)
73     m=y;
74       else
75       m=z;
76    // return m;
77 }
78 main()
79 {
80     int x,y,z,l;
81     cin>>x>>y>>z;
82     max(x,y,z,l);
83     cout<<"max=  "<<l<<endl;
84    // cout<<"avg=  "<<avg(x,y,z);
85 }
```

```
64
65 int max(int x,int y,int z)
66 {
67     int m;
68     m=x;
69     if (y<x && z<x)
70     m=x;
71      else
72     if (y> x && y>z)
73     m=y;
74       else
75       m=z;
76     return m;
77 }
78 main()
79 {
80     int x,y,z,l;
81     cin>>x>>y>>z;
82     max(x,y,z);
83     cout<<"max=  "<<max(x,y,z)<<endl;
84    // cout<<"avg=  "<<avg(x,y,z);
85 }
```




```
56
57 float avg(int x,int y,int z) //دالة تعيد المتوسط الحسابي للأعداد
58 {
59    return float (x+y+z)/3;
60 }
61 void max(int x,int y,int z,int &m) //دالة تعيد العدد الكبير للأعداد
62 {
63     //int m;                    //الرسال بمرجع
64     m=x;
65     if (y<x && z<x)
66     m=x;
67      else
68     if (y> x && y>z)
69     m=y;
70       else
71       m=z;
72    // return m;
73 }
74 main()
75 {
76     int x,y,z,l;
77     cin>>x>>y>>z;
78     max(x,y,z,l);
79     cout<<"max=  "<<l<<endl;
80     cout<<"avg=  "<<avg(x,y,z);
81 }
```



# Recursion العودية

- Recursive functions التوابع العودية
  - Functions that call themselves التوابع العودية تنادي نفسها
  - Can only solve a base case تنتهي بالوصول إلى الحالة الابتدائية
- If not base case طالما لم يتم الوصول إلى الحالة الابتدائية
  - Break problem into smaller problem(s) تقسم المسألة الى مسائل اصغر

```
6
7  //دالة لطباعة عبارة نصية بشكل عودي
8  void f(int n)
9  {
10     if (n<1)          //شرط ابتدائي للخروج من دالة العودية
11     return; //ايقاف عملية الاستدعاء
12     else
13     cout<<"hello"<<endl;
14     f(n-1);    //استدعاء ذاتي للتابع نفسه
15 }
16 main()
17 {
18     int n;
19     cout<<"n=  ";
20     cin>>n;
21     f(n);
22 }
```

```
"C:\Users\SMART\Desktop\محاضرات طرطوس
n=  5
hello
hello
hello
hello
hello
Press any key to continue...
```

# العودية لحساب العاملي Recursion

- Example: factorial

$$n! = n * (n-1) * (n-2) * \ldots * 1$$

  - Recursive relationship ( n! = n * ( n – 1 )! )

$$5! = 5 * 4!$$

$$4! = 4 * 3! \ldots$$

  - Base case (1! = 0! = 1)



```
38
39 //العاملي باستخدام الطريقة التكرارية
40 int fact(int n)
41 {
42    int f=1;
43    for (int i=1;i<=n; i++)
44    {
45       f=f*i;
46    }
47       return (f);
48
49 }
50 int main()
51
52    int n,m,f1,f2;
53    cin>>n;
54    cin>>m;
55    f1=fact(n);
56    f2=fact(m);
57    cout<<f1<<endl;
58    cout<<f2<<endl;
59 }
```

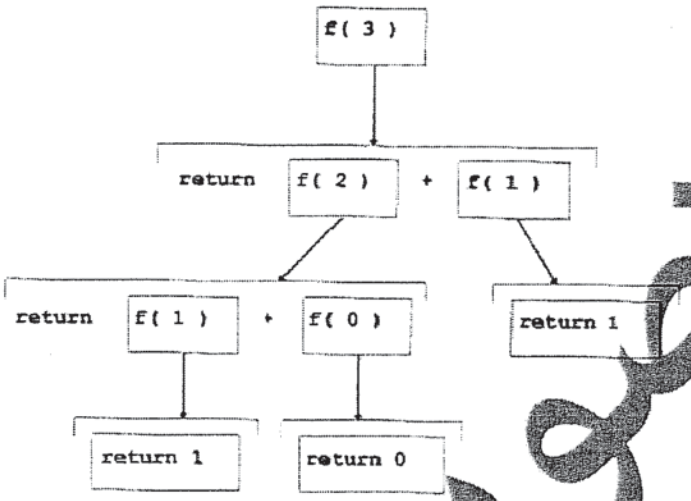
```
62
63 //العاملي باستخدام التعاودية
64 int fact( int n)
65 {
66   if (n==0||n==1)
67   return 1;
68   else
69   return n*fact(n-1);
70 }
71 main()
72 {
73   int n;
74   cin>>n;
75   cout<<fact(n)<<endl;
76 }
```

```
/*
5*fact(4)
 4*fact(3)
  3*fact(2)
   2*fact(1)
      1
```

## استخدام العودية في حساب سلاسل فيبوناشي
## Example Using Recursion: Fibonacci Series

- Fibonacci series: 0, 1, 1, 2, 3, 5, 8...
  - Each number sum of two previous ones
    كل عدد يساوي مجموع العددين السابقين من السلسة
  - Example of a recursive formula:
    - *fib(n) = fib(n-1) + fib(n-2)*

## Example Using Recursion: Fibonacci Series




**مثال:** إيجاد قيمة سلسلة فيبوناتشي عند حد ما حيث تعطى سلسلة فيبوناتشي كالتالي:

0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 ,....

1- باستخدام الحلقات التكرارية

```
81  //برنامج لحساب فيبوناتشي باستخدام التكرار//
82  main()
83  {
84    int n,f,f0,f1;
85    cin>>n;
86    f0=0;f1=1;f=1;
87    for(int i=2;i<=n;i++)
88    {
89      f=f1+f0;
90      f0=f1;
91      f1=f;
92    }
93    cout<<f<<endl;
94  }
```

```
98  //برنامج يعتمد على تابع فيبوناتشي بشكل تكراري//
99  int fib(int n)
100 {
101   int f,f0,f1;
102   f=1;f0=0;f1=1;
103   for (int i=2;i<=n;i++)
104   {
105     f=f1+f0;
106     f0=f1;
107     f1=f;
108   }
109   return f;
110 }
111 main()
112 {
113     int n;
114     cin>>n;
115     cout<<fib(n)<<endl;
116 }
```

```
//0 1 2 3 4 5 6 7  8  9 .....
//0 1 1 2 3 5 8 13 21 34 .....
```

```
                    //0 1 2 3 4 5 6 7  8  9 ......
                    //0 1 1 2 3 5 8 13 21 34 ......
122
123   //    برنامج يستدعي تابع فيبوناتشي بشكل عودي
124 int fib(int);
125 main()
126 {
127 int n,f;
128 cout<<"n=";cin>>n;
129 f=fib(n);
130  cout<<f;
131   }
132 int fib(int n)
133 {
134 if(n==0)
135 return 0;
136 else
137 if(n==1)
138 return 1;
139 else
140 return fib(n-1 )+fib(n-2);
141 }
```

```
"C:\Users\SMART\Desktop\ات طرطوس
n=3
2
Press any key to continue...
```

```
"C:\Users\SMART\Desktop\اضرات طرطوس
n=6
8
Press any key to continue...
```

```
if(n==0||n==1)  //اصغر قيمة تعطيها السلسلة
return n;
```

fib(2)+fib(1)

fib(1)+fib(0)1

```
22
23   //  1....n      مجموع الأعداد من
24   // 1+2+3+ ...+ (n-1)+n
25    int sum(int n)
26    {
27     if (n==1)          //اصغر قيمة يصل لها المجموع
28     return n;
29     else
30     return n+sum(n-1);
31    }                  // 3+sum(2)
32    main()             // 2+sum(1)
33    {                  //     |
34        int n;         //   1
35        cin>>n;
36        cout <<sum(n)<<endl;
37    }
```

```
"C:\Users\SMART\Desktop\ضرات طرطوس
3
6
Press any key to continue...
```

```
41
42  //مجموع اعداد صحيحة ضمن مجال معين//
43  // x....y  / 4....6
44   int sum(int x,int y)
45   {
46       if (x==y)     // الحالة التي تعطي أقل قيمة
47       return x;     // y أو x التابع يعيد اما
48       else
49       return y+sum(x,y-1);    //y من اخر قيمة
50   }
51   main()
52   {
53       int x;int y;      //x=4    y=6
54       cout<<"x=  ";
55       cin>>x;
56       cout<<"y=";
57       cin>>y;
58       cout<<sum(x,y)<<endl;
59   }
```

```
/*
6+sum(4,5)
 5+sum(4,4)
      |
      4
*/
```