الـخو ارزميـا ت وحل الـمشكلـة

حيدر محمدعلي محمدرضا آل طعمة

المعهد التقني كربلاء
مركز الحاسبة الالكترونية

2022 - 2021

مفاهيم اساسية في البرمجة

I. البيانات (Data) :

مجموعة من الحقائق الضرورية التي تعبر عن مواقف وأفعال معينة سواء أكان ذلك التعبير بأرقام أو رموز أو كلمات أو ما شابه ذلك ولا تفيد هذه البيانات في شيء وهي على صورتها الحالية.

ومن الأمثلة على ذلك (درجات طالب ، أجرة ساعة العمل) .

II. المعلومات (Information)

هي المعرفة التي تكونت نتيجة لتحليل البيانات المدخلة واجراء العمليات المطلوبة عليها والمعلومات الناتجة تكون اكثر معنى من البيانات وتساعدِ متخذي القرارات في تحقيق أغراض معينة.

ومن الامثلة على ذلك (معدل الطالب ، دخل الموظف الأسبوعي).

مفاهيم اساسية في البرمجة

III. معالجة البيانات (Data processing)

هي إخضاع البيانات المدخلة للتحليل وإجراء مجموعة من العمليات الحسابية والمنطقية عليها باستخدام وسائل معينة بغرض الحصول على معلومات مفيدة .

IV. البرمجة (Programming)

هي مجموعة من الأوامر والتعليمات سطور برمجية (Code) يتم كتابتها على إحدى منصات البرمجة لتتحول إلى لغة الآلة بغرض تنفيذ عمليات معالجة على البيانات المتاحة وإظهار نتائج بغرض الاستفادة منها؛ ويتم ذلك باستخدام إحدى لغات البرمجة وباتباع قواعد خاصة.

البرنامج

```
def fibonacci1(n):
   if n ==1 or n == 2:
      return 1
   return fibonacci1(n-1) + fibonacci1(n-2)
def fibonacc2(n):
        a,b =1,1
        for i in range (n-1):
          a,b = b, a+b
        return a
def fibonacci3():
     a,b = 1,1
     while True:
            yield a
            a,b = b, a+b
n = 0
for i in fibonacci3():
           if n >= 1001:
             break;
           print (n," : ",i)
           n += 1
```

- كتابة الايعازات أو الأوامر التي ترشد الكمبيوتر إلى كيفية معالجة البيانات.
- سلسلة من الايعازات التي يتم إدخالها إلى الكمبيوتر ، مما يمكّنه من إجراء عمليات منطقية وحسابية محددة على البيانات.

أنواع لغات البرمجة

1- لغات منخفضة المستوى Low Level Language

هي لغات البرمجة التي تُكتب عبر نظام العد الثنائي (Binary Number System) المتمثل بالأعداد 0 و 1 ، بحيث يفهمها جهاز الحاسوب، وعلى عكس اللغات عالية المُستوى فإن اللغات المُنخفضة المُستوى تكون مُعقّدة، وغير قابلة للقراءة من قبل المستخدمين، وينطوي هذا النوع من اللغات على فرعين من اللغات، وهما الآتي :

- لغة الآلة : (Machine Language) وهي اللغة التي تتمكن أجهزة الحواسيب فقط من فهمها، ويتم تمثيل هذه البرامج بالنظام الثنائي، وتُعتبر عملية كتابة هذه البرامج أمراً معقداً للغاية على المُبرمجين، كما تختلف لغة الآلة من جهاز حاسوبٍ لآخر، حيث يتم إنشاء البرامج في المعالجات الموجودة على الأجهزة، بالتالي لا يمكن استخدام البرنامج المكتوب بلغة الآلة عبر جهاز يحتوي على مُعالج من نوع PowerPC على جهاز آخر يحتوي على مُعالج Intel .

- **EXAMPLE :**

Machine Instruction	Machine Operation
00000000	Stop Program
00000001	Turn bulb fully on
00000010	Turn bulb fully off
00000100	Dim bulb by 10%
00001000	Brighten bulb by 10%
00010000	If bulb is fully on, skip over next instruction
00100000	If bulb is fully off, skip over next instruction
01000000	Go to start of program (address 0)

أنواع لغات البرمجة

- لغة التجميع : (Assembly Language) تُعتبر لغة التجميع أسهل نسبياً من لغة الآلة؛ وذلك لاحتوائها على بعض مفردات اللغة الإنجليزية؛ ككلمة add، وكلمة sub على سبيل المثال، مما يجعل قراءة برامجها وفهمها أسهل مقارنةً ببرامج لغة الآلة، وتعمل برامج هذا المستوى من لغات البرمجة بمثابة مُترجم يعمل على تحويل برامج اللغات العالية المُستوى التي يكتبها الإنسان إلى لغة الآلة التي تفهمها الأجهزة، وكما هو الحال في لغة الآلة فإن لغات التجميع هي لغات غير محمولة؛ أي أنه لا يُمكن نقل برنامج مكتوب بلغة التجميع من جهاز كمبيوتر لآخر.

- EXAMPLE :
- The following example divides 8 with 2. The **dividend 8** is stored in the **16 bit AX register** and the **divisor 2** is stored in the **8 bit BL register**.

```
section .text
global main ;must be declared for using gcc
main: ;tell linker entry point
mov ax,'8'
sub ax, '0'
mov bl, '2'
sub bl, '0'
div bl
add ax, '0'
mov [res], ax
mov ecx,msg
mov edx, len
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)
int 0x80 ;call kernel
nwln
mov ecx,res
mov edx, 1
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)
int 0x80 ;call kernel
mov eax,1 ;system call number (sys_exit)
int 0x80 ;call kernel
section .data
msg db "The result is:", 0xA,0xD
len equ $- msg
segment .bss
res resb 1
```

When the above code is compiled and executed, it produces following result :

```
The result is :  4
```

أنواع لغات البرمجة

2- لغات عالية المستوى High Level Language

تستخدم تنسيقاً مألوفاً للمُبرمجين عند استخدامها في كتابة البرامج، حيث تُسمى الأوامر المكتوبة بها بالكود البرمجيّ، وتكون الرموز والمفردات المستخدمة في كتابة الكود قريبة من مفردات اللغة الإنجليزية، كما تُوفر اللغات العالية المستوى إمكانية أضافة الكثير من التعليقات، والشروح ضمن البرنامج الذي يعمل عليه المُبرمج؛ الأمر الذي يجعل تلك البرامج أسهل، ويزيد من قابليتها للقراءة، والتعديل من المُستخدِم نفسه، أو حتى من المُستخدِمين الآخرين.

لذلك تُعدّ كتابة البرامج عبر هذا النوع من اللغات أسهل وأسرع من كتابتها باللغات البرمجية ذات المستوى المنخفض، ولا تعتمد اللغات العالية المُستوى على نوع جهاز الحاسوب؛ فهي مُصممة للعمل على أجهزة الحاسوب المُختلفة، وبعض النظر عن نوعها، أو نظامها تشغيلها، ومن الأمثلة على هذا النوع من لغات البرمجة؛ لغة C#، ولغة Java، ولغة JavaScript، ولغة Python، ولغة SQL.

- **Example :**

  Write program to find factorial for number between 1 and 170

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
using namespace std;
void fact();
double factorial_number;
int acount,number;
int main()
{
 system("cls");
 fact();
 system("pause");
}
```

```
void fact()
{
 cout<<"Enter Your Number To Find Factorial [ 1-170 ] : ";
 cin>>number;
 if (number <= -1 || number == 0) factorial_number=1;
    else
              {
               factorial_number=1;
               for(acount=1;acount<=number;acount++)
               factorial_number=factorial_number*acount;
              }
    cout<<"\n\nFactorial [ "<<number<<" ] is ===>
"<<factorial_number<<"\n"<<endl;
}
```

أنواع البرامج

- **برنامج المستخدم** : برنامج كمبيوتر مكتوب من قبل الشخص الذي يستخدمه أو من قبل موظفي المنظمة التي ستستخدمه.

- **برامج التطبيقات** : برامج مكتوبة لحل مشاكل معينة أو لإنتاج تقارير محددة أو لتحديث ملفات معينة، هذه البرامج تقوم بعمل مفيد نيابة عن مستخدم الكمبيوتر، وعادة ما يستخدمه المستخدمون النهائيون، ومن الأمثلة على ذلك معالجات الكلمات ومشغلات الوسائط وبرامج المحاسبة.

- **برامج أنظمة التشغيل** : مجموعة من البرامج التي تقع بين برامج التطبيقات وأجهزة الكمبيوتر. برنامج نظام التشغيل هو وسيط بين الأجهزة وبرامج التطبيقات ، ويستخدم مصطلح برنامج النظام أحيانًا بالتبادل مع نظام التشغيل ، لكن برنامج النظام يعني جميع البرامج المتعلقة بتنسيق عمليات الكمبيوتر. تتضمن برامج النظام نظام التشغيل ولكنها تتضمن أيضًا برامج BIOS وبرامج التشغيل وبرامج الخدمة.

حل المشكلة

- حل المشكلات هو جزء من عملية أكبر تضم تحديد المشكلة وإزالة التكرار والتحليل والفحص والإصلاح وخطوات أخرى. البرمجة الخطية وغير الخطية ونظرية الأرتال والمحاكاة كلها أدوات أخرى لحل المشكلات. يتضمن جزء كبير من علم الحاسوب تصميم أنظمة آليّة كليًّا تقوم لاحقًا بحل بعض المشاكل المحددة وهي أنظمة تستجيب للبيانات المُدخلة، وخلال وقت معقول، تحسب الرد الصحيح أو تقدر تقريبيًا حلًا صحيحًا بما فيه الكفاية. إضافة إلى ذلك، يقضي الأشخاص في علم الحاسوب وقتًا كبيرًا بشكل مُدهش في تقصّي وإصلاح المشكلات في برامجهم (التنقيح البرمجي).

هناك عدة خطوات للعمل هي :

i. تعريف المشكلة
ii. تحديد السبب أو الأسباب الرئيسيـة للمشكلة
iii. تطوير الحلول البديلة
iv. تحديد الحل الافضل
v. تنفيذ الحل
vi. تقييم النتيجة

ولعمل الخطوات السابقة يجب الاخذ بنظر الاعتبار ما يأتي :

I. فهم المشكلة ؟

II. تقسيم المشكلة ؟

III. حل المشكلة ؟

أنواع البيانات والمتغيرات المستخدمة في البرامج

1. Constant

   X = 7, h = 3.1 , w = ‘A’ , K = “Hello World”

2. Variables

   int a

   float m

   double f

   Dim matrix (7,3) As integer

   Dim StudentName As String

المخططات الانسيابية

هو وصفا تصويريا أو تمثيل مصور للخوارزمية يوضح حل المسألة من البداية إلى النهاية بشكل أكثر وضوحا وأسهل فهما مع اخفاء التفاصيل لإعطاء الصورة العامة للحل، والمخطط الانسيابي يتكون نتيجة استخدام مجموعة من الاشكال كل شكل يوضح عملية معينة وتربط هذه الاشكال بخطوط وأسهم توضح اتجاه سير تنفيذ العمليات.

ويطلق على المخططات الانسيابية تسميات أخرى مثل (خرائط سير العمليات أو خرائط التتابع).

فوائد استخدام المخططات الانسيابية

I. تمكن المبرمج من الالمام الكامل بالمسالة المراد حلها والسيطرة على جميع اجزائها بحيث تساعده على اكتشاف الأخطاء المنطقية (logic error) والتي تعتبر من أهم الأخطاء التي تجهد المبرمج ومن ثم تصحيحها .

II. توضح سير العمليات وتسلسل تنفيذها .

III. تعتبر المخططات الانسيابية وسيلة مناسبة و مساعدة في كتابة ومتابعة البرامج ذات التفرعات الكثيرة .

IV. تساعد المبرمج وبسهولة على تعديل برنامج ما ، فبمجرد النظر الى المخطط الانسيابي نظرة سريعة ،يدرك ماهيه المسألة وإمكانية التعديل .

V. يعتبر الاحتفاظ برسوم المخططات الانسيابية لحلول مسائل معينة أمرا مهما إذ يعتبر مرجعا مهما يمكن استخدامه لحل مسائل أخرى مشابهة دون الحاجة الى الرجوع الى المبرمج الأول باعتبار أن الحلول الأولى قد صيغت في خطوات واضحة بسيطة و مفهومة .

VI. تعتبر المخططات الانسيابية من الوسائل و الأدوات الهامة لتوثيق البرنامج .

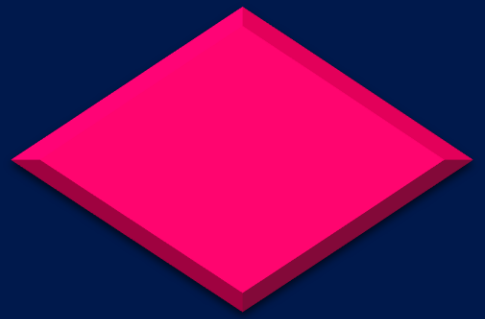
الاشكال المستخدمة في المخططات الانسيابية

البداية والنهاية

العمليات الحسابية

الادخال والاخراج

الربط



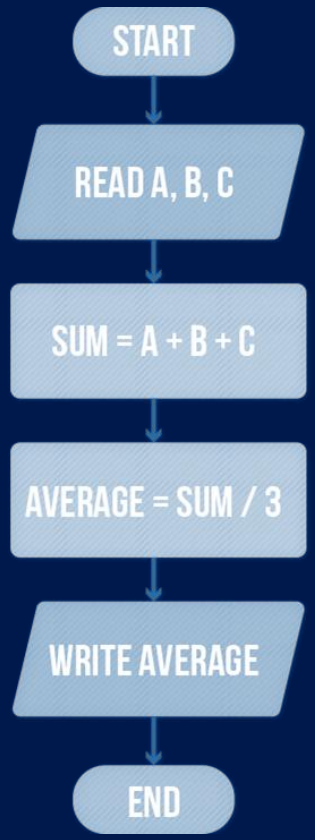
الشرط

الروتين الفرعي

التكرار

التوجيه

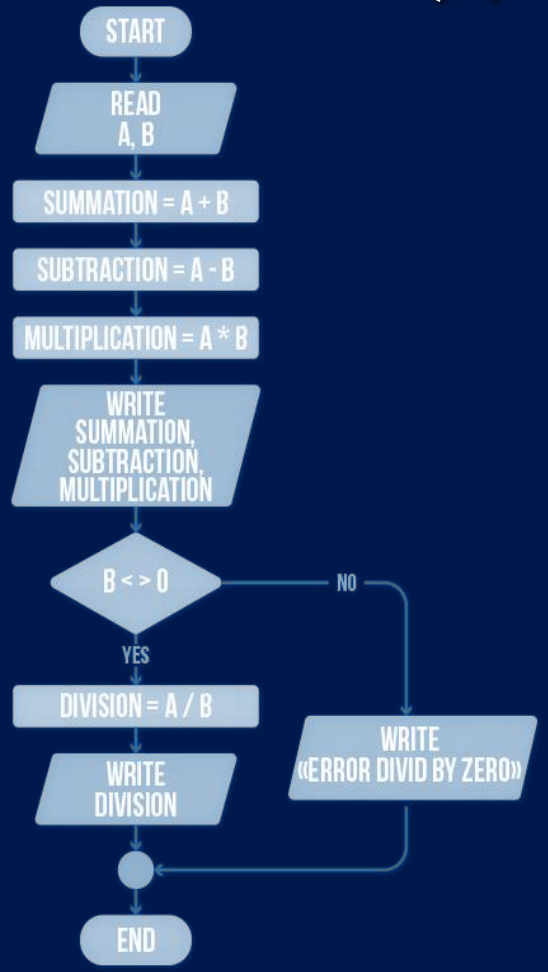
أنواع المخططات الانسيابية



I. المخططات الانسيابية البسيطة (Simple flow chart)

يخلو هذا النوع من التفرعات و التكرارات وأنما يحتوي على مجموعة أوامر وأحداث متسلسلة

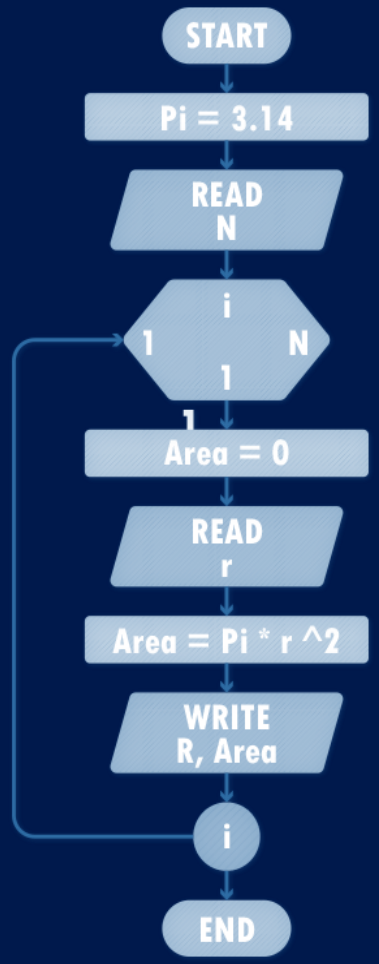
أنواع المخططات الانسيابية



II. المخططات الانسيابية المتفرعة (Branched flow chart)

يتضمن هذا النوع أتخاذ القرارات أو مفاضلة بين خيارين أو اكثر

أنواع المخططات الانسيابية



III. المخططات الانسيابية ذات الحلقات (Loop flow chart)

نحتاج لهذا النوع من المخططات لإعادة تنفيذ عملية او مجموعة من العمليات في مسألة معينة (البرنامج) عدداً محدداً أو غير محدد من المرات.

أمثلة

I. ارسم المخطط الانسيابي لإيجاد مربع ومكعب العدد X.

II. ارسم المخطط الانسيابي لقراءة قيمتين معلومتين واختيار أكبرهما.

III. ارسم المخطط الانسيابي لترتيب قيمتين ترتيبا تصاعديا.

IV. ارسم المخطط الانسيابي لحساب قيمة Y طبقا للمعادلات الأتية :

$$Y=\begin{cases} X^2+1 & X>0 \\ 5 & X=0 \\ 2X+5 & X<0 \end{cases}$$

V. ارسم المخطط الانسيابي لطباعه الأعداد الطبيعية المحصورة بين ١ الى ١٠٠ ومربعاتها.

VI. ارسم المخطط الانسيابي لحساب معدل طالب في ٥ مواد دراسية وطباعة المعدل أذا كان الطالب ناجح .

VII. ارسم المخطط الانسيابي لحساب حاصل ضرب الحدود التالية : (5,10,15,20,.....,500).

VIII. ارسم المخطط الانسيابي لطباعة X^2 لخمسة قيم لـ X .

IX. ارسم المخطط الانسيابي لإيجاد حاصل ضرب N من القيم .

X. ارسم المخطط الانسيابي لحساب عدد الأعداد الموجبة وعدد الأعداد السالبة وعدد الأصفار لـ ١٠٠ عدد.

XI. ارسم المخطط الانسيابي لقراءة أسم ومعدل ٢٠ طالب وإيجاد عدد الطلبة الناجحين والراسبين ونسبة النجاح الكلية.

XII. ارسم المخطط الانسيابي لاختبار العدد X فيما أذا كان فرديا أم زوجيا .

XIII. ارسم المخطط الانسيابي لحساب عدد الأعداد الزوجية و عدد الأعداد الفردية لمجموعة تتكون من ١٠ أعداد .

الخوارزمية

سميت الخوارزمية بهذا الاسم نسبة إلى العالم أبو جعفر محمد بن موسى الخوارزمي الذي ابتكرها في القرن التاسع الميلادي. الكلمة المنتشرة في اللغات اللاتينية والأوروبية هي " algorithm " وفي الأصل كان معناها يقتصر على خوارزمية لتراكيب ثلاثة فقط وهي : التسلسل والاختيار والتكرار.

التسلسل : تكون الخوارزمية عبارة عن مجموعة من التعليمات المتسلسلة، هذه التعليمات قد تكون إما بسيطة أو من النوعين التاليين.

الاختيار : بعض المشاكل لا يمكن حلها بتسلسل بسيط للتعليمات، وقد تحتاج إلى اختبار بعض الشروط وتنظر إلى نتيجة الاختبار، إذا كانت النتيجة صحيحة تتبع مسار يحوي تعليمات متسلسلة، وإذا كانت خاطئة تتبع مسار آخر مختلف من التعليمات. هذه الطريقة هي ما تسمى اتخاذ القرار أو الاختيار.

التكرار : عند حل بعض المشاكل لا بد من إعادة نفس تسلسل الخطوات عدد من المرات. وهذا ما يطلق عليه التكرار.

الخوارزمية

- **تعريف الخوارزمية Algorithm**

هي مجموعة من الخطوات الرياضية والمنطقية المتسلسلة اللازمة لحل مشكلة ما

ويجب ان تتوافر فيها الشروط الاتية :-

١. تحديد المخرجات Output
٢. تحديد المدخلات Input
٣. تحديد عمليات المعالجة Process

الخوارزمية

خصائص الخوارزمية Characteristic of Algorithm

١. يجب أن تكون لكل خوارزمية بداية ونهاية.
٢. يجب أن تكون كل خطوة من خطوات الخوارزمية واضحة تعبر عن العملية المقصودة من تلك الخطوة.
٣. يجب أن تكون الخوارزمية متسلسلة بحيث تؤدي الى عمل معين أو الوصول الى نتيجة أو نهاية.
٤. يجب أن تكون الخوارزمية كاملة بحيث تأخذ بنظر الاعتبار جميع الاحتمالات التي تواجه طريقة التنفيذ (أي أن تكون الخوارزمية عامة).

الخوارزمية

مثال : أكتب خوارزمية لحساب الوسط الحسابي (المعدل) لأربعة أعداد.

الحل:

I. البداية -
II. ادخل الأعداد X1,X2,X3,X4
III. احسب المعدل AV= (X1+X2+X3+X4)/4
IV. اطبع AV
V. النهاية

الخوارزمية

مثال : أكتب الخوارزمية التي تعطي نتيجة حل التعبير الرياضي الآتي :

$$R = X^2 + 2X$$

الحل:

I. البداية
II. ادخل قيمة X
III. احسب قيمة R = (X*X) + (2*X)
IV. اطبع قيمة R
V. النهاية

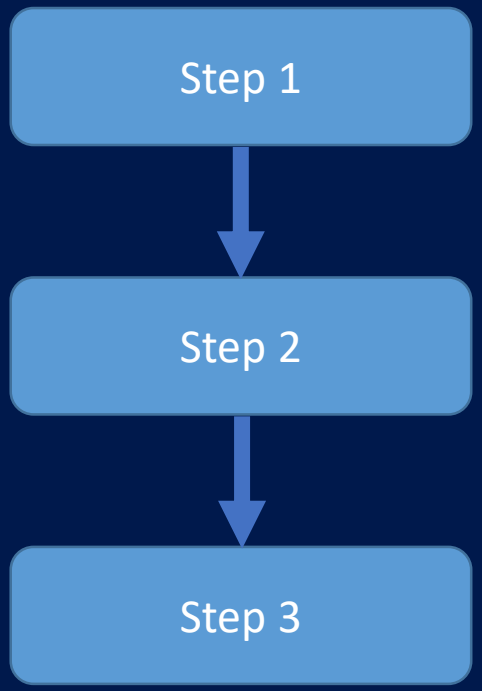
الخوارزمية

تصميم الخوارزمية Algorithms design

تمتاز الخوارزميات بما يلي :-

I. تكون أهداف هذه الخطوات واضحة.
II. تكون عبارة عن مجموعة من الخطوات المتسلسلة والمرقمة.
III. تكتب بلغة سلسة وبسيطة كما يفضل الحرص على الدقة في شرح طريقة الحل.

الخوارزمية



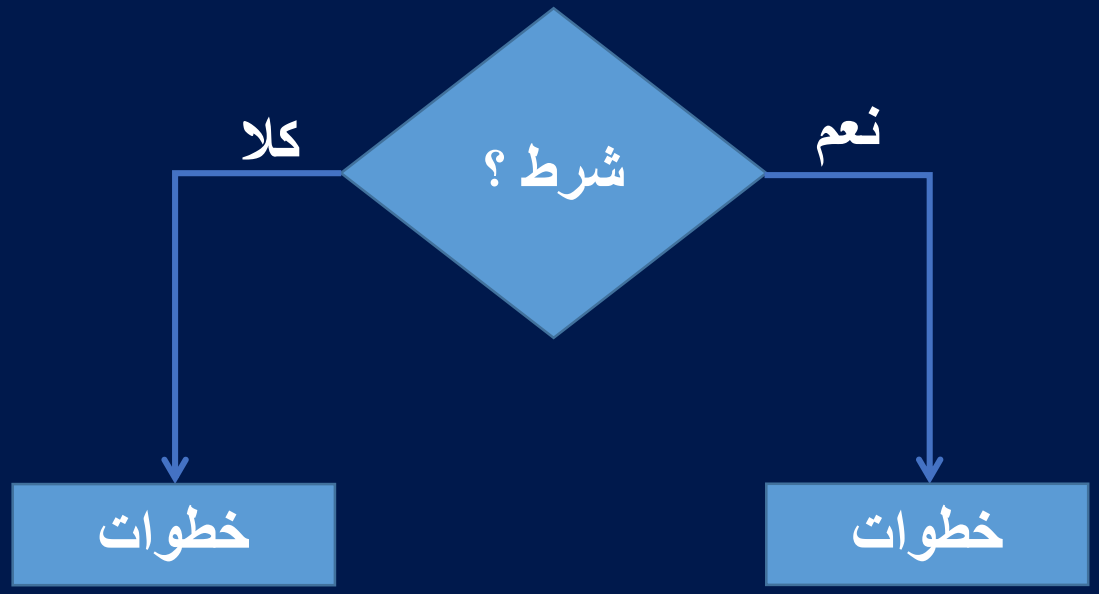
أنواع الخوارزميات:
يمكن تصنيف الخوارزميات إلى :

I. **الخوارزمية المتسلسلة المستقيمة Sequential**
وهي الخوارزمية التي تحتوي على خطوات متسلسلة بحيث لا يؤدي تنفيذ أي منها إلى تفرع خارج التسلسل.

الخوارزمية

II. الخوارزمية الشرطية أو المتفرعة Conditional

تكون هذه الخوارزمية في حالة كون المسألة ذات احتمالين للحل ولذلك تتفرع الخطوات وتحتوي هذه الخوارزمية على أوامر شرطية



الخوارزمية

مثال : اكتب خوارزمية لحساب معدل طالب في اربعة مواد دراسية وطباعة المعدل اذا كان الطالب ناجح .

الحل :

١. البداية
٢. ادخل درجات الطالب D1,D2,D3,D4,D5
٣. أحسب معدل الطالب AV=(D1+D2+D3+D4+D5)/5
٤. إذا كان AV>=50 أطبع ناجح والا اطبع راسب
٥. النهاية

الخوارزمية

مثال : اكتب خوارزمية لطباعة عبارة "العدد الموجب" إذا كان العدد A موجب و أطبع عبارة "العدد سالب" اذا كان سالب.

الحل :

١. البداية
٢. ادخل قيمة العدد A
٣. إذا كان A>=0 أطبع "العدد موجب" وإلا أطبع "العدد سالب"
٤. النهاية

الخوارزمية

الخوارزمية المتكررة Repetition

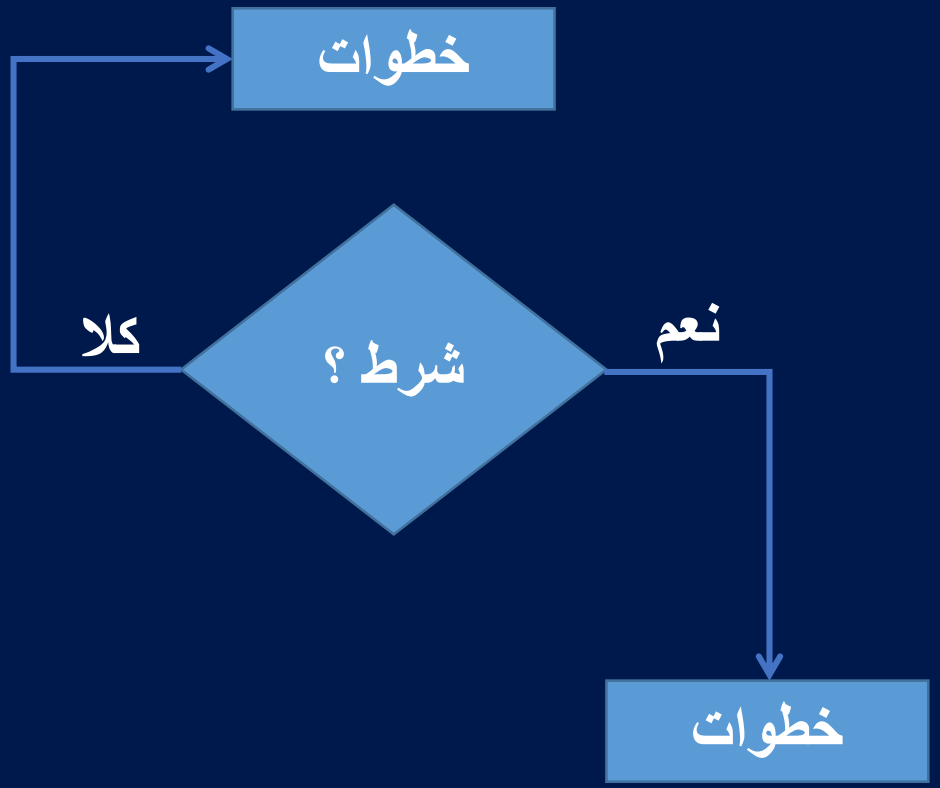
التكرار: عند حل بعض المسائل لابد من إعادة تنفيذ مجموعة من الخطوات عدة مرات وهذا التنفيذ المتكرر لهذه الخطوات هو الحلقة التكرارية Loop .

الخوارزمية المتكررة : هي التي تقوم بتنفيذ مجموعة من الخطوات عدة مرات استنادا إلى شرط معين
ولهذه الخوارزمية الحالتين التاليتين :

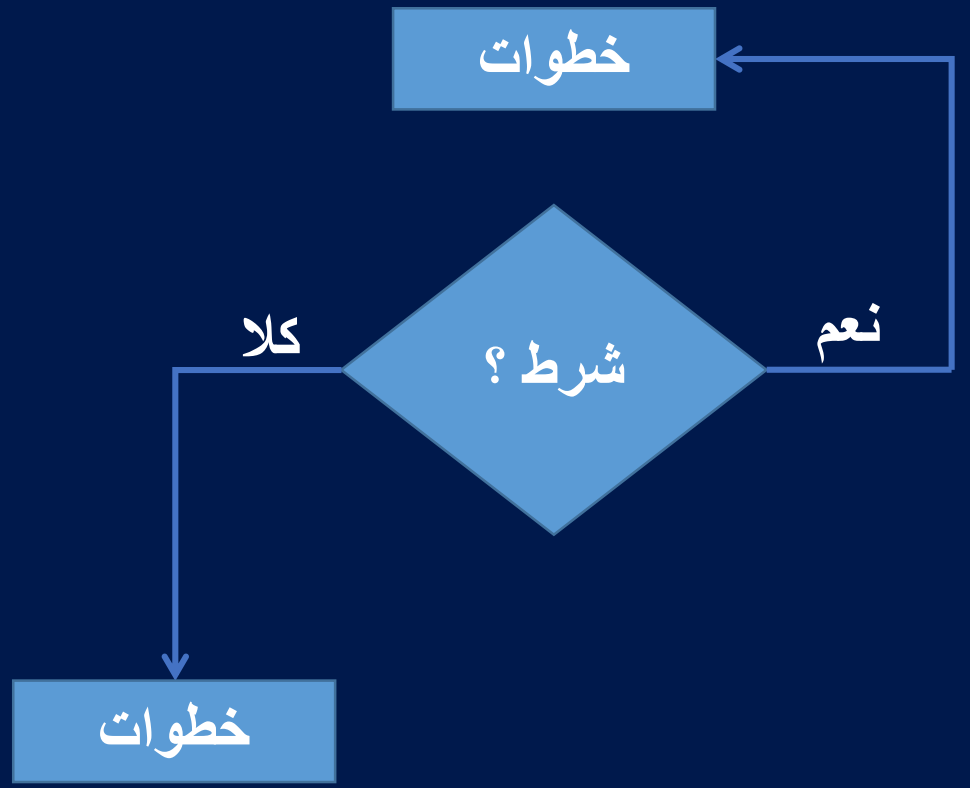
I. تحقق الشرط
II. عدم تحقق الشرط

الخوارزمية

عدم تحقق الشرط



تحقق الشرط



الخوارزمية

ملاحظة :

- العداد (Counter) : هو المتغير الذي نستطيع من خلاله الوصول الى العدد المحدد من التكرار لحل مشكلة معينة.

بعد اختيار متغير عددي معين نطلق عليه تسمية العداد نتبع هذه الخطوات :

I. إعطاء العداد قيمة ابتدائية .
II. إعطاء العداد مقدار زيادة معين.
III. إعطاء العداد قيمة نهائية.

الخوارزمية

مثال : اكتب خوارزمية لطباعة الأعداد الزوجية فقط المحصورة بين ٣٧ - ١

الحل :

١. البداية
٢. افرض I=2
٣. أطبع I
٤. إذا كان I>=37 اذهب إلى الخطوة ٦
٥. اجعل I=I+2 اذهب الى الخطوة ٣
٦. النهاية

الخوارزمية

مثال : اكتب خوارزمية لإيجاد مجموع الأعداد الزوجية المحصورة بين ٢٠ – ١٠

الحل :

١. البداية
٢. نفرض K=10
٣. نفرض SUM = 0
٤. احسب المجموع SUM =SUM + K
٥. اجعل K=K+2
٦. إذا كان K<=20 اذهب إلى الخطوة ٤
٧. أطبع SUM
٨. النهاية

البرنامج

صفات البرنامج الجيد : Qualities of a Good Program

يجب أن يُحقق البرنامج المواصفات التالية لنقول عنه أنه جيد:

I. يُنفذ بشكل صحيح: فلا يجب أن يقوم البرنامج بطباعة ناتج جمع عددين مثلاً بدل ضرب عددين.
II. فعال : يجب أن يستغرق البرنامج أقل وقت ممكن وفي بعض الأحيان يجب أن يُأخذ بعين الاعتبار استخدام أقل قدر ممكن من ذاكرة الحاسوب.
III. سهل القراءة و الفهم : يجب أن يكون سهل القراءة والفهم من قبل أي شخص آخر غير الذي قام بكتابة البرنامج.
IV. سهل الفحص و التدقيق : أي يمكن التأكد من صحة البرنامج و قيامه بالعمل المطلوب.
V. سهل التعديل : وذلك لوجود إصدارات جديدة و نسخ معدلة من البرامج بشكل دائم.

يمكن الحصول على البرنامج الصحيح والفعّال وتحقيقه باستخدام الخوارزميات و بناء المعطيات المناسبة.

البرنامج

مراحل تطور البرنامج : Program Development Stages

I. توصيف المتطلبات Requirements Specification : فهم المدخلات و المخرجات.
II. التصميم Design : تحديد العمليات الرئيسية التي تطبق وافتراض وجود أجهزة معالجة لتنفيذ هذه العمليات.
III. التحليل Analysis : المفاضلة بين الخوارزميات المتوفرة لحل نفس المسألة تبعا لمقاييس مفاضلة متفق عليها تعقيدات الوقت، تعقيدات الخزن لاختيار أفضلها.
IV. التحسين و الترميز Refinement & Coding : في هذه الخطوة يتم تحديد التمثيل البياني ثم كتابة إجراءات لكل عملية وتكوين نسخة متكاملة للبرنامج.
V. التحقق من الصلاحية Verification : تتضمن هذه الخطوة ثلاث جوانب مختلفة :

a. البرهنة على الصحة Proving : يجب إثبات ان البرنامج صحيح قبل استخدامه، حيث يتم استخدام طرق معينه للبرهنة على الصحة.
b. الاختبار Testing : هي عملية توليد نماذج بيانية يعمل عليها البرنامج حيث إن الهدف منها هو إعطاء إشارة على وجود أخطاء في البرنامج.
c. تشخيص الأخطاء Debugging : هي عملية تحديد مواقع الأخطاء في البرنامج و تصحيح الإيعازات التي سببت تلك الأخطاء.

البرنامج

- كتابة البرنامج

```
#include <iostream>
using namespace std;
void fact();
double factorial_number;
int acount,number;
int main()
{
  system("cls");
  fact();
  system("pause");
}
void fact()
{
  cout<<"Enter Your Number To Find Factorial [ 1-170 ] : "; cin>>number;
  if (number <= -1 || number == 0) factorial_number=1;
       else
                  {
                    factorial_number=1;
                    for(acount=1;acount<=number;acount++)
                      factorial_number=factorial_number*acount;
                  }
     cout<<"\n\nFactorial [ "<<number<<" ] is ===> "<<factorial_number<<"\n"<<endl;
}
```

البرنامج

التنفيذ و إيجاد الأخطاء : Implementation & Debugger

A. تجربة البرنامج و تنفيذه :

بعد الحصول على البرنامج، تتم تجربته للتأكد من صحته وذلك باستخدام عينة من المعطيات الاختبارية Test Data فإذا ثبت صحة طريقة الحل بمطابقة النتائج الخارجة من الحاسب مع النتائج التي تم الحصول عليها يدويا على سبيل المثال، يمكن تنفيذ البرنامج على المعطيات الحقيقية.

في علم الحاسوب، يعرّف الخطأ المنطقي على أنه خطأ في البرنامج يجعله يعمل بطريقة غير صحيحة، و لكنه لا يتوقف عن العمل بشكل مفاجئ. الخطأ المنطقي ينتج عنه مخرجات غير مقصودة أو غير مرغوب بها رغم أنها قد لا تُكتشف حينها.

على عكس البرنامج الذي يحتوي على خطأ قواعدي syntax error ، هي الأخطاء الناتجة عن كتابة الكود بطريقة خاطئة، لذلك فإن ال compiler لن يفهم الكود المكتوب، لأن اللغة لها قواعد يتبعها ال compiler فلن يستطيع فهم ما تريد لذلك يعطيك خطأ، مثال على ذلك تصور أنك بدل أن تكتب void كتبت vod أو بدل while كتبت whle و هكذا.

البرنامج

B. تصحيح الإخطاء الإملائية :

إحدى الطرق لإيجاد الأخطاء المنطقية هو أن يتم اختبار السطور البرمجية الخاصة بالعمليات الحسابية التي لم تعطي النتيجة المرغوب بها.
المصحح Debugger : و يسمى أيضا المنقح في علم البرمجة هو أداة لتشخيص و إيجاد و إزالة الأخطاء من أنظمة الحواسيب، خصوصا من البرمجيات.
هذا مثال على دالة Function مكتوبة بلغة سي C++ هدفها حساب معدل رقمين وهي تحتوي على خطأ منطقي. هنالك أقواس ناقصة في المعادلة، لذا فإن البرنامج يترجم بنجاح و يعمل و لكنه لا يعطي نتائج صحيحة بسبب أولوية العمليات الحسابية في لغة سي أي ان البرنامج يبدأ بتقديم القسمة قبل الجمع.

البرنامج

أنواع الأخطاء البرمجية : Types of errors

ضمن عملية تطوير البرنامج application development يواجه المبرمجون عدة انواع من الأخطاء.

I. الاخطاء الاملائية والقواعدية Syntax Errors هي الاخطاء الاكثر شيوعا وانتشارا ولكنها سهلة العلاج، وهي التي تظهر اثناء كتابة البرنامج عندما لا يتم استخدام اللغة البرمجية بالشكل الصحيح.

فالأخطاء الاملائية تحدث غالبا بسبب وجود خطأ املائي لغوي (طباعي)في كتابة الاوامر الخاصة باللغة البرمجية المستخدمة، مثل كتابة كلمة محجوزة بشكل خطأ او استخدام غير صحيح لدالة معينة (يعني خطأ في اسم الدالة.)

أما الاخطاء القواعدية هي التي تحدث نتيجة عدم فهم المبرمج لقواعد اللغة البرمجية المستخدمة وعدم التزامه وخروجه عن القواعد المتبعة في صياغة وتركيب الجملة مثلا نسيان اغلاق قوس وقد تم فتحه مسبقا او نسيان علامات الاقتباس في النصوص وغيرها.

البرنامج

II. الاخطاء التنفيذية اخطاء وقت التنفيذ Run Time Errors
هي الأخطاء التي تظهر بعد ترجمة البرنامج اثناء التنفيذ، فيحصل شيء غير متوقع وغير قادر على المعالجة من تلقاء نفسه، وبالتالي يتوقف العمل مثلا ا وجود نقص في المدخلات.

مثلا لو كتبت برنامج لقراءة البيانات من ملف معين فعند تشغيل البرنامج رغم ان البرنامج صحيح سيظهر خطأ إذا لم يكن الملف موجود أو إذا كان الملف موجود وتآلف. وغالبا يمكن حل هذه الأخطاء عن طريق إعادة صياغة البرنامج وتشغيل البرنامج من جديد.

البرنامج

III. الاخطاء المعنوية المنطقية Semantic Errors

في أي لغة برمجية بعد تصحيح الأخطاء الإملائية والقواعدية من البرنامج يمكن ان يظهر خطأ اخر، بعد تنفيذ البرنامج يعطي سلوك غير صحيح لتنفيذ البرنامج ويبين أن النتائج غير صحيحة وغير متوافقة مع ما صمم من اجله البرنامج. هذا النوع من الأخطاء صعب الاكتشاف وعندما تكتشفه فقد نصرف الكثير من الوقت والجهد لمعالجته وتصحيحه، وهو عادة ا يظهر كنتيجة غير صحيحة وليس كخطأ واضح.

وهذه الاخطاء تحدث عندما يرتكب المبرمج خطأ في بناء سلسلة التعليمات اللازمة لإعطاء النتيجة المطلوبة أي ان الخطأ يكون في منطق الحل.

مثل كتابة معادلة خاصة بمساحة المربع على أنها مساحة المثلث:

N = 0.5 * a * b

وصحتها كالتالي: N = a ^2

او محاولة قسمة عدد على صفر، او في حالة الحلقات التكرارية الغير المنتهية، او قراءة متغيرات معينة وهي ليست معرفة، او لدينا عملية رياضية معقدة تتطلب في أحد اجزاءها جمع رقمين ثم جاء المبرمج من دون قصد ووضع اشارة عملية الطرح بدل من الجمع، وغيرها من الحالات الاخرى.

البرنامج

اسباب الخطأ في البرنامج :

هناك عدة اسباب تؤدي الى فشل البرنامج وعدم تحقيق الهدف الذي ك تب من اجله وهي:

١ . عدم الفهم الكامل للمشكلة من خلال دراستها وتحليلها بشكل صحيح.
٢ . الخوارزمية غير صحيحة فإذا كانت المسالة معقدة نوعا ما فمن الافضل ان توضع الخوارزمية من قبل المبرمج وتدقق من قبل مبرمج اخر للتأكد من صحتها قبل تحويلها الى برنامج.
٣ . عدم الإلمام بلغة برمجية بشكل جيد والذي يسبب بعض الاخطاء.
٤ . الاخطاء المطبعية التي ترتكب اثناء ادخال البرنامج الى الحاسبة.
٥ . وجود اخطاء في مترجم اللغة مع ان هذا الخطأ نادر الحصول.

البرنامج

ادوات التنقيح التصحيح :
عند اكتشاف خطأ في البرنامج يجب على المبرمج ان يتتبع الخطأ في البرنامج ليكتشف موقعه ومن ثم تصحيحه وأكثر الاساليب شيوعا في تتبع الخطأ هي وضع عبارات Printفي مواقع متعددة في البرنامج حيث تطبع قيم المتغيرات التي هي موضع الشك ويستمر الفحص لحين اكتشاف الاخطاء.

البرنامج

التوثيق Documentation :

وهي مرحلة هامة من مراحل بناء النظام البرمجي حيث يتم توثيق البناء الداخلي للبرنامج؛ وذلك بغرض الصيانة والتطوير. يفضل أن يكون فريق خاص يهتم بعملية التوثيق لجميع المشاكل والحلول التي يمكن أن تظهر أثناء بناء البرمجية. وبدون التوثيق قد يصل المبرمج إلى مرحلة لا يعود بعدها قادرا على متابعة صيانتها وتطويرها؛ مما يزيد الكلفة المادية والزمنية الخاصة بهذا البرنامج إلى حدود غير متوقعة، أو بمعنى آخر الفشل في بناء برامج ذات جودة عالية ودورة حياة طويلة. وهناك أكثر من طريقة للتوثيق :

١ . توثيق المبرمج وهو ممكن أن يكون بإضافة تعليقات داخل الشفرة البرمجية.

٢ . توثيق المحلل بكتابة مستندات شرح لدورة البرنامج.

٣ . توثيق الاختبار للنظام وفيها يتم تسجيل نقاط الخلل في البرنامج.

البرنامج

الصيانة Maintenance :

ان هذه المرحلة هي المرحلة الأطول في حياة النظام البرمجي لبقاء النظام قادرا على مواكبة التطورات والمعدات الحديثة، جزء من هذه المرحلة يكون في تصحيح الأخطاء، والجزء الآخر يكون في التطوير وإضافة تقنيات جديدة.

البرنامج

تصميم البرنامج Program Design :

هو تطوير خوارزمية لحل مشكلة. فالخوارزميات هي مجموعة من الخطوات لحل مشكلة في فترة زمنية محدودة.
تكون الخوارزمية مستقلة عن أي لغة برمجة معينة ، لذلك يجب أن يتجنب تصميم البرنامج، الاعتماد على ميزات لغة كمبيوتر معينة.
التصميم هو أهم خطوة في كتابة أي برنامج. عندما تكتب برامجك الأولى ، تكون عادةً صغيرة بما يكفي بحيث لا تظهر أهمية التصميم بسهولة. كلما كبرت البرامج ، أصبحت أهمية التصميم أسهل في الفهم. فمن الجيد أن تعتاد على تصميم برامجك بدلاً من القرصنة منذ البداية ، على الرغم من أن مرحلة التصميم قد تبدو مضيعة للوقت بالنسبة لك. سيسمح التصميم الجيد بتعديل البرنامج بمرور الوقت.
معظم البرامج ليست ثابتة، أي أنهم يتغيرون لدمج الميزات الجديدة أو التنفيذ بكفاءة أكبر أو التغلب على السلوك غير الصحيح.
يقسم التصميم مشكلة كبيرة إلى عدة أجزاء مستقلة أو شبه مستقلة. تصبح هذه الأجزاء وحدات يمكن للمبرمج إعادة استخدامها في برامج أخرى أو استبدالها بوحدة مماثلة ولكنها أكثر كفاءة أو بوحدة تصحيحها. نظرًا لأن الوحدات صغيرة الحجم ، يسهل فهمها نسبيًا ، وبالتالي يسهل تصحيحها. تؤخذ بشكل فردي، فهي بسيطة ومباشرة للتنفيذ، فهي قوية بما يكفي لحل المشاكل الكبيرة والمعقدة.

البرنامج

هناك نوعين من التصميم : الأول التصميم من أعلى الى الاسفل والثاني التصميم من اسفل الى الأعلى

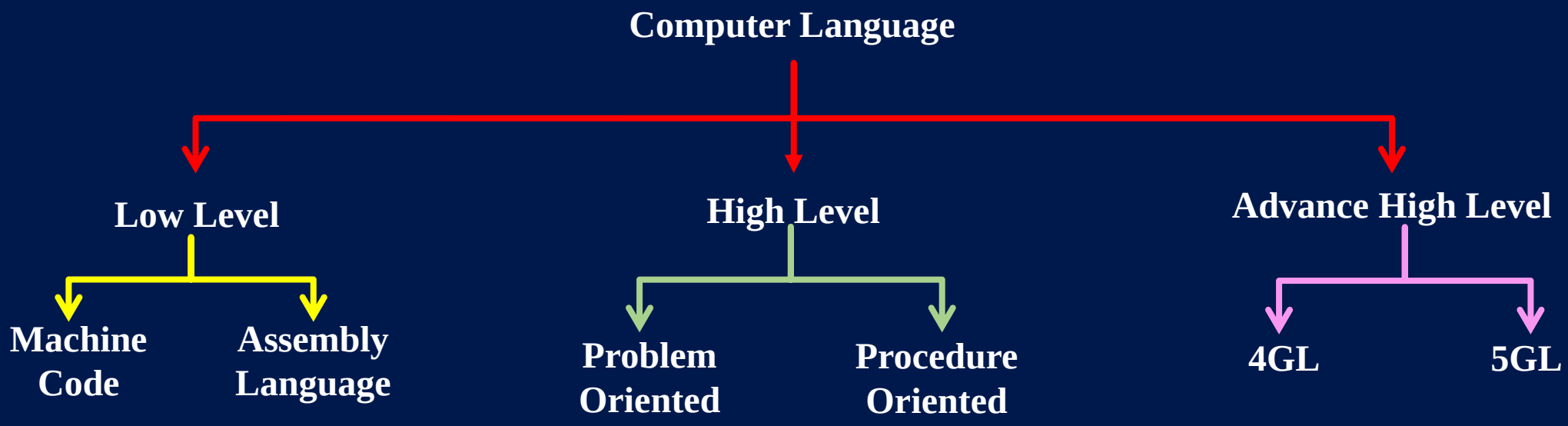
تعتبر طريقتا التصميم من أعلى لأسفل، ومن أسفل لأعلى من الاستراتيجيات التي تستخدم في معالجة المعلومات وترتيب المعرفة. تستخدم طريقة التصميم من أعلى إلى أسفل في كثير من الحالات كمرادف لمصطلح التحليل أو تقسيم المشكلة إلى وحدات صغيرة بينما تستخدم طريقة التصميم من أسفل إلى أعلى كمرادف لمفهوم التخليق أو الاصطناع.

البرنامج

التصميم من أعلى الى الأسفل Top down :

- إن طريقة التصميم من أعلى لأسفل) وتعرف أيضا بالتصميم المتدرج(هي أساسا عبارة عن تحليل النظام بهدف اكتساب المعرفة بالنظم الفرعية التركيبية الخاصة به. تصاغ النظرة العامة للنظام في طريقة التصميم من أعلى لأسفل بحيث لا تفصل أي أنظمة فرعية عن المستوى الأول. تتم بعد ذلك تنقية النظام الفرعي بتفصيل أكبر وأحيانا في مستويات فرعية إضافية إلى أن يتم تقليص كامل المواصفات إلى العناصر الأساسية.
- تعتبر هذه الطريقة أسلوبا برمجيا من حيث أنها عماد لغات البرمجة الإجرائية التقليدية التي يبدأ فيها التصميم عن طريق تحديد الأجزاء المعقدة ومن ثم تقسيمها إلى أجزاء أصغر تباعا.
- إن التقنية المتبعة في كتابة البرنامج باستخدام طرق التصميم من أعلى لأسفل هي كتابة إجراء يعمل على تسمية كل الوظائف الرئيسية التي يحتاجها. يقوم فريق البرمجة فيما بعد بالاطلاع على متطلبات كل من هذه الوظائف ويتم تكرار هذه العملية. ستقوم هذه الإجراءات الفرعية المجزأة في النهاية بأداء الأفعال بطريقة بسيطة جدا بحيث يمكن كتابة الشيفرة الخاصة بها بسهولة وإيجاز.
- يكون البرنامج جاهزا لاختباره عندما تتم برمجة مختلف الروتينات الفرعية الخاصة به. يمكن أن يكون العمل في المستوى المنخفض مستقلا عن طريق تعريف كيفية دمج التطبيق معا في المستوى الأعلى.

البرنامج

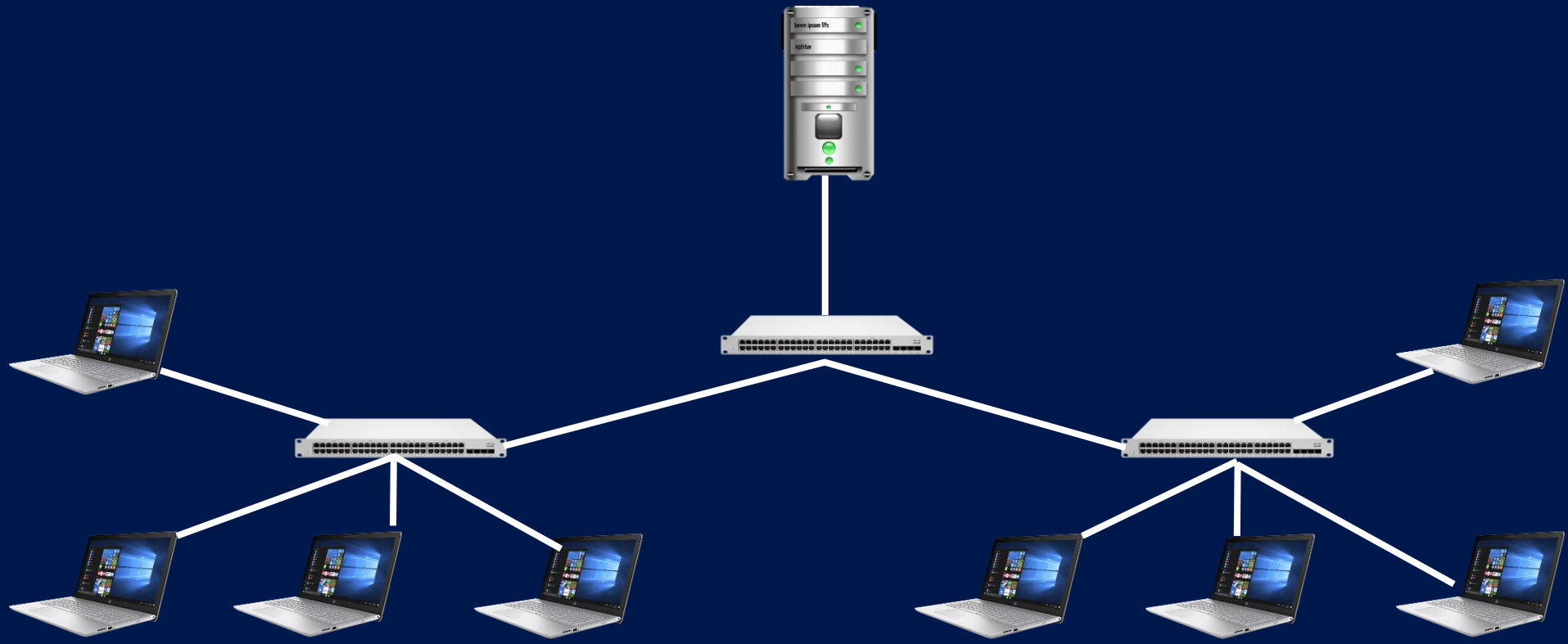


البرنامج

التصميم من الاسفل الى الاعلى Down Top :

- طريقة التصميم من الأسفل لأعلى فهي عبارة عن دمج أجزاء الأنظمة معا لإنشاء نظم أكثر ضخامة وبهذا يتكون نظام ناشئ هو عبارة عن تجميع للأنظمة الفرعية الخاصة بالأنظمة الأصلية.
- يتم في هذا التصميم تحديد العناصر الأساسية المنفردة للنظام أولا في تفصيل كبير، من ثم ربطها معا لتكون أنظمة فرعية أكبر حجما يتم ربطها بدورها مع بعضها على مستويات عدة أحيانا إلى أن يتم تشكيل نظام مكتمل عالي المستوى في النهاية .
- في هذه الطريقة يتم تحديد عناصر القاعدة المنفردة للنظام في تفصيل كبير حيث يتم ربط هذه العناصر معا لتكون أنظمة فرعية أكبر إلى أن يكتمل النظام.
- تشبه هذه الطريقة في عملها نموذج "البذرة" حيث تكون البدايات صغيرة ولكنها تنمو لتصبح في النهاية أكثر تعقيدا واكتمالا.

البرنامج



العملية او الاجراء (Process) : هو عبارة عن برنامج تحت التنفيذ .

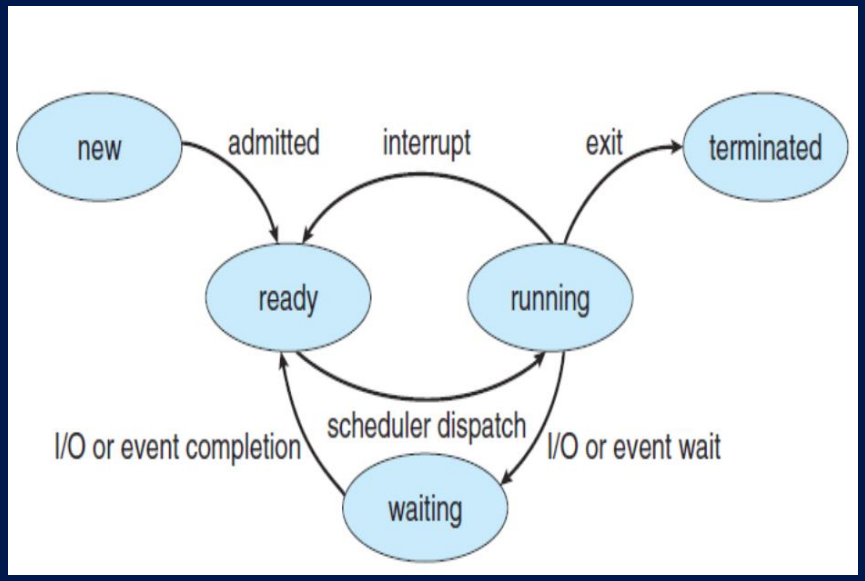
دورة حياة العملية داخل الحاسوب:
هي بناء البرنامج منذ لحظة ورود الفكرة في الذهن الى الخروج للعميل وخدمة ما بعد البيع. فإن دورة الحياة للنظام البرمجي طويلة وتمر بمراحل مختلفة وعديدة ليصل الى ايدي العملاء إذ إن كتابة الاكواد والشفرات ليست هي النقطة الرئيسية في دورة حياة البرنامج بل ممكن ان تكون هي اقلها وقتا و مجهوداً.
تغير العملية حالتها أثناء تنفيذها والتي تسمى دورة حياة العملية. تتكون دورة حياة العملية من خمس مراحل وهي:

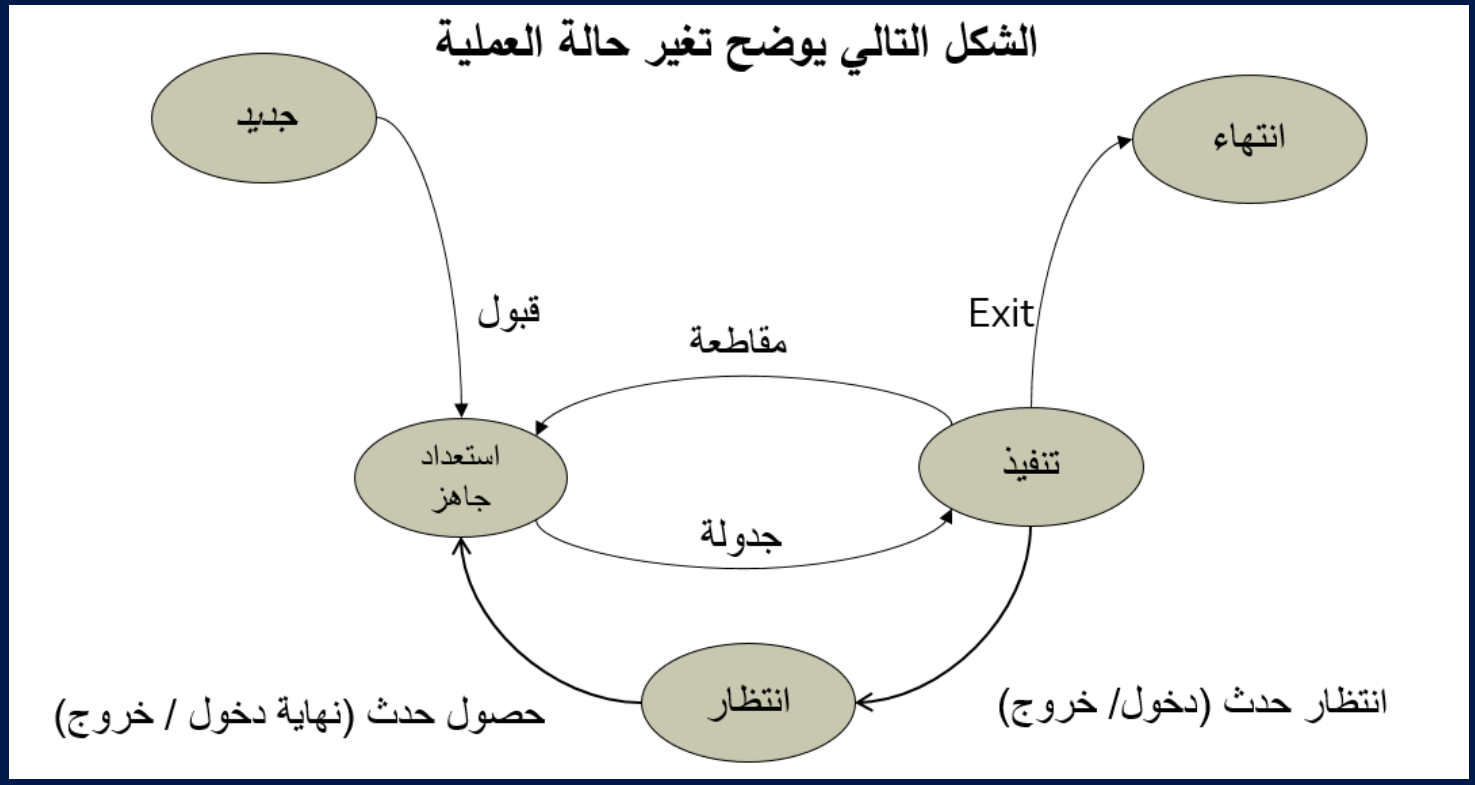
١ . جديد .
٢ . ادارة .
٣ . انتظار .
٤ . جاهز .
٥ . تم إنهاؤه .

State Description

1. New The process which is being created
2. Running Instructions being executed
3. Waiting The process is waiting for an event to get occur
4. Ready The process is waiting to be assigned to a processor
5. Terminated The process completed its execution

توجد هذه الإحصائيات في الغالب في جميع الأنظمة ولكن بعض أنظمة التشغيل تحدد حالات العملية بدقة أكبر. يمكن للمعالج إجراء عملية واحدة فقط في كل مرة ، ومع ذلك ، قد تكون العديد من العمليات جاهزة أو في انتظار.


new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting



I. جديد (New) :

العملية (Process) دخلت النظام أي موجودة داخل القرص المغناطيسي وتحتاج الى مصادر وهي نوعين :
أما مصادر مادية (hardware) مثل المعالج C.P.U، الطابعة، الذاكرة وغيرها، أو مصادر برمجية (software) كأي برنامج من نظام التشغيل او أي ملف اخر. ويقوم نظام التشغيل على جمع تلك المصادر ، وفي هذه المرحلة قد تكون اكثر من عملية (process) موجودة .

II. الاستعداد جاهز (Ready) :

العملية (Process) في هذه الحالة تكون قد حصلت على جميع المصادر المطلوبة لغرض التنفيذ ما عدا المعالج (C.P.U) والتي قام بجمعها نظام التشغيل وإن نظام التشغيل يختار مجموعة من العمليات من حالة جديد (New) الى حالة الاستعداد (Ready) ويتم الاختيار بالاعتماد على خوارزميات معينة.

III. التنفيذ (Running) :
يختار نظام التشغيل عملية واحدة معينة فقط من مجموعة العمليات في حالة الاستعداد جاهز (Ready) ويمنحها الى المعالج C.P.U لغرض التنفيذ .

IV. الانتهاء (Complete) :
انهاء العملية (Process) من تنفيذ جميع الايعازات ويعتبر هذا الانهاء طبيعي (Normal) أي قد انتهى تنفيذ العملية (Process) , او قد يكون الانهاء غير طبيعي عند حصول خطأ معين من قبل العملية (Process) وفي كلتا الحالتين الانهاء الطبيعي والإنهاء الغير الطبيعي(يقرر نظام التشغيل اخراج العملية (Process) من النظام وسحب المعالج (C.P.U) منها لغرض منحه الى عملية اخرى تنتظر التنفيذ .

٧. الانتظار (Waiting) :
في مرحلة التنفيذ قد لاتصل العملية (Process) الى مرحلة الانتهاء لأسباب منها طلب عمليات ادخال او اخراج (I/O Request) فهنا تدخل العملية في حالة انتظار و تنتقل من حالة الانتظار الى حالة الاستعداد الجاهز بعد زوال السبب اعلاه أي بعد اتمام عمليات الادخال والإخراج.

ملاحظة (Note) :
قد يحصل انتقال للعملية (Process) من حالة التنفيذ (Running) الى حالة الاستعداد الجاهز (Ready) عندما تكون هنالك اما مقاطعة (Interrupt) او نفاذ الفترة الزمنية T.S.E (Time Slice Expire) وفي هذه الحالة يتم سحب المعالج (C.P.U) من العملية (Process) .

الروتينيات الفرعية (Subroutine)

هو نمط مشتق من البرمجة الهيكلية يستند إلى مفهوم استدعاء الإجراء، وما الإجراء إلا سلسلة من الخطوات الحسابية التي يتعين فهمها، والتي يمكن استدعاؤها في أي وقت أثناء تنفيذ البرنامج، من قبل إجراءات أخرى أو من قبل الإجرائية ذاتها.

هذا النمط من انماط البرمجة يملك عدة أسماء فالتسمية الرئيسية لهذا النمط من انماط لغات البرمجة يدعى اللغة الإجرائية (**Procedural**) كونها تعتمد على الاجرائيات ويمكن ان تدعى باللغة الآمرية (**Imperative**) وذلك من كونها لغة أوامر بحيث تعتمد على تلقين الحاسب الحل خطوة بخطوة كما قد نجد تسمية البرمجة الهيكلية (**Structured**) بحيث تعتمد على مفهوم استدعاء الاجرائيات أو كما هي معروفة بالروتين.

تعتمد هذه اللغة عدة مبادئ منها تقسيم البرنامج إلى عدة اقسام جزئية لتسهيل القراءة و إعادة الاستخدام تسمى هذه الاجزاء بعدة أسماء:

1. Procedures . 1اجراءات.
2. Functions . 2توابع.
3. Methods . 3منهجيات.
4. Routines . 4روتينات.
5. Subroutines . 5روتينات فرعية.

ملاحظة:
توصيف حل المشكلة خطوة بخطوة تعد من مساوئ هذا النمط حيث يجب على المبرمج ان يقوم بحل المشكلة المتناولة بنفسه و ليس عن طريق الحاسب.
من هذا يتضح ان الروتينات الفرعية Subroutines هي شبه برنامج صغير اذا كان عندك سلسلة عمليات متشابهة سوف تجريها داخل البرنامج كثيرا و تكررها كثيرا فلا داعي لإعادة كتابتها كل مرة بل يكفيك ان تكتبها مرة واحدة و تطلق عليها اسم و كلما ذكرت هذا الاسم داخل البرنامج تتم تلك العمليات و يصبح هذا الاسم كأنه من أوامر اللغة.
مثال/ لنفرض ان هناك اوامر لرسم جدول بأبعاد و مساحة معينة و تحتاج لرسمه كثيرا فليس معنى هذا انك كل مرة تريد رسم الجدول ستعيد كتابة اوامره الكثيرة في كل مرة داخل البرنامج ، فقط جمع اوامر رسمه في روتين فرعي و تسميته بإسم و كلما ذكرت هذا الاسم يتم رسم الجدول.

بشكل عام الدوال و البرامج الفرعية عبارة عن برامج او اجزاء برامج ثانوية يتم استخدامها لأداء غرض معين و
من فوائدها:

١ . تقليل و تلافي التكرار في بناء البرامج مرة اخرى.
٢ . تقليل الوقت المطلوب لبناء البرامج و المشاريع.
٣ . التقليل من الذاكرة المطلوبة لشفرات وبيانات المشروع.

هناك نوعين من البرامج الفرعية من حيث البناء و التصريح هما:

١ . **الوحدات النمطية : Module**و هي تمثل احد الكيانات الاساسية التي تقدمها لغة VB (Visual Basic)للمبرمجين .

٢ . **البرامج الفرعية** و هي البرامج الخاصه بالمشروع المراد عمله و هي تشمل الدوال و البرامج الفرعية.

الفرق بين هذين النوعين هو ان الوحدات النمطية Moduleيمكن استخدامها في أي كيان بأي نموذج موجود ضمن
المشروع أما النوع الاخر فإنه يمكن استخدامه فقط ضمن الكيانات التابعة لنموذج واحد فقط.

الصيغة العامة لتعريف البرامج الفرعية هي:

```
[(public| private] subname [(argument list]
[(statement(s]
[(statement(s]
End sub
```

الروتينيات الفرعية (Subroutine)

و يتم استدعاء الدالة او الروتين إن كان اسم الروتين ( ) Nameيكون كالتالي:
Name . 1من خلال ذكر اسمه فقط.
Call Name() . 2من خلال كتابة Callثم اسمه يليه اقواس هلالية.
Name(3) . 3من خلال اسمه مع المعاملات.