



وزارة التعليم العالي والبحث العلمي
الجامعة التقنية الشمالية
المعهد الاداري التقني/نينوى



الحقبة التعليمية

القسم العلمي: تقنيات
انظمة الحاسوب

اسم المقرر: خوارزميات

المستوى: الاول

الفصل الدراسي: الثاني



معلومات عامة

اسم المقرر:	خوارزميات
القسم:	تقنيات أنظمة الحاسوب
الكلية: المعهد	الاداري التقني / نينوى
المستوى	الاول
الفصل الدراسي:	الثاني
عدد الساعات الاسبوعية:	نظري 2 عملي
عدد الوحدات الدراسية:	2
الرمز:	CST102
نوع المادة	نظري نظري عملي كلهما
هل يتوفر نظير للمقرر في الاقسام الاخرى	
اسم المقرر النظير	
القسم	
رمز المقرر النظير	
معلومات تدريسي المادة	
اسم مدرس (مدرسي) المقرر:	حاضر ابراهيم عبد
اللقب العلمي:	مدرس مساعد
سنة الحصول على اللقب	2005
الشهادة :	ماجستير
سنة الحصول على الشهادة	2004
عدد سنوات الخبرة (تدريس)	20

الوصف العام للمقرر

يهدف المقرر الى تقديم المفاهيم الاساسية اللازمة لتصميم بنى المعطيات والخوارزميات المناسبة لحل المسائل البرمجية، وتقدير ادائها. ولهذا يعرض في هذا المقرر عدة نقاط منها مفاهيم اساسية في خوارزميات، دراسة تعقيد الخوارزميات، الخوارزميات العودية الاساسية، الخوارزميات التراجعية، بنى المعطيات، خوارزميات الفرز الداخلي. يتوقع بعد انتهاء الطالب لهذا المساق ان يكون قادرا على التعامل مع المفاهيم الاساسية لحل المسائل البرمجية عن طريق تصميم خوارزميات الحل بشكل فعال. والقدرة على تحليل وتوصيف الخوارزميات المصممة وحساب درجة تعقيدها كذلك معرفة بنى المعطيات المناسبة لحل المسائل البرمجية والخوارزميات المتميزة والتي تستخدم هذه البنى.

الاهداف العامة

- 1- تنمية التفكير المنطقي المنظم لدى الطالب
2. معرفة الطالب بكيفية بناء خوارزميات لحل مشكلة بسيطة.
3. تدريب الطالب على تحويل الخوارزميات الى برنامج.
4. تدريب الطالب لحل المشاكل باستخدام برمجة الحاسب

الأهداف الخاصة

استيعاب الطالب للمفاهيم الاساسية اللازمة لتصميم بنى المعطيات والخوارزميات المناسبة لحل المسائل البرمجية، وتقدير ادائها. ولهذا يعرض في هذا المقرر عدة نقاط منها مفاهيم اساسية في خوارزميات، دراسة تعقيد الخوارزميات، الخوارزميات العودية الاساسية، الخوارزميات التراجعية، بنى المعطيات، خوارزميات الفرز الداخلي. يتوقع بعد انتهاء الطالب لهذا المساق ان يكون قادرا على التعامل مع المفاهيم الاساسية لحل المسائل البرمجية عن طريق تصميم خوارزميات الحل بشكل فعال. والقدرة على تحليل وتوصيف الخوارزميات المصممة وحساب درجة تعقيدها كذلك معرفة بنى المعطيات المناسبة لحل المسائل البرمجية والخوارزميات المتميزة والتي تستخدم هذه البنى.

الأهداف السلوكية او نواتج التعلم

- اكساب الطالب مهارة تحديد وتحليل المشكلة.
- اختيار الاسلوب المنطقي في تحديد المدخلات والمخرجات الخاصة بالمشكلة.
- كتابة الخوارزمية لهذه المشكلة مع تحويلها الى المخطط الانسيابي الخاص بها.
- أمثلة أهداف تدريسية:
 - بعد الانتهاء من الدرس (المحاضرة) سيكون الطالب قادرا على ان:
 - فهم اسلوب التحليل المنطقي لحل المشاكل التي تواجهه .
 - اختيار الاسلوب المنطقي الافضل.
 - القدرة على كتابة البرنامج الخاص باي مشكلة.
 - تنفيذ البرنامج على الحاسوب من خلال الاسلوب المنطقي البرمجي الذي اكتسبه.

المتطلبات السابقة

- اذكر أي متطلبات سابقة قد يحتاجها الطالب قبل التسجيل في المقرر، مثل مواد دراسية سابقة أو مهارات معينة.

الأهداف السلوكية او مخرجات التعليم الأساسية		
ت	تفصيل الهدف السلوكي او مخرج التعليم	آلية التقييم
1	يكتسب الطالب مهارة تحديد وتحليل المشاكل التي تواجهه في الحياة العملية بطريقة منطقية.	-الاختبارات النظرية - اختبارات قصيرة - اختبار عملي -تصحيح الواجبات
2	القدرة على حل هذه المشاكل بأسلوب برمجي يستطيع تحويله الى برنامج ينفذ على الحاسوب.	-الاختبارات النظرية - اختبارات قصيرة - اختبار عملي -تصحيح الواجبات
3	اكتساب مهارة بناء الانظمة التطبيقية الكبيرة من خلال الاسلوب المنطقي الخوارزمي والاسلوب البرمجي الحاسوبي.	-الاختبارات النظرية - اختبارات قصيرة - اختبار عملي -تصحيح الواجبات
4	اكتساب مهارة تحويل المهارات اليدوية الى مهارات الكترونية من خلال الاسلوب المنطقي الخوارزمي الذي يسمى بالاثمنة.	-الاختبارات النظرية - اختبارات قصيرة - اختبار عملي -تصحيح الواجبات

أساليب التدريس (حدد مجموعة متنوعة من أساليب التدريس لتناسب احتياجات الطلاب ومحتوى المقرر)

مبررات الاختيار	الاسلوب او الطريقة
كون المقرر يتميز بالجانب العملي من حيث اسلوب التعامل مع الحاسوب	1. اسلوب المحاضرة الحضورية في المخبر الحاسوبي.
	.2
	.3
	.4
	.5
	.6

طرائق القياس:

الاختبارات النظرية	1
اختبارات قصير	2
اعمال المختبر	3
اختبارات عملية (تطبيق على الحاسوب)	4
تصحيح الواجبات	5

تفاصيل المادة الدراسية:-

الموضوع الفرعي	الموضوع	الاسبوع
مبادئ أولية عن البرمجة/ تعريف البرنامج/ لغات البرمجة/ (لغات البرمجة العليا High level language ولغات البرمجة الدنيا low level language)- تعريف برنامج المستخدم (User program)- البرامج التطبيقية (Application programs)- برمجيات المنظومة (Operating system)	مبادئ أولية عن البرمجة	الاول
فهم المشكلة/ تقسيم المشكلة / عملية حل المشاكل ، أنواع البيانات والمتغيرات المستخدمة في لغات البرمجة وتعريفها في البرنامج/ الثوابت والمتغيرات Numeric / Constants & variable الحرفية String والرقمية	حل المشكلة Problem Solving	الثاني
المخططات الانسيابية (Flow chart)- فوائد المخططات الانسيابية- الأشكال المستخدمة في رسم المخططات الانسيابية، أنواع المخططات الانسيابية- المخططات الانسيابية البسيطة (Simple flow chart)- المخططات الانسيابية المتفرعة (Branched flow chart)- المخططات الانسيابية ذات الحلقات (Loop flow chart)	المخططات الانسيابية (Flow chart)	الثالث - الخامس
الخوارزميات Algorithms/تعريف الخوارزمية/ تصميم الخوارزمية (Algorithm design)، أنواع الخوارزميات المتسلسلة (Sequential) والشرطية (Conditional) ، والمتكررة (Repetition)	الخوارزميات Algorithms	السادس - الثامن
صفات البرنامج الجيد -مراحل تطور البرنامج (program development stages)، كتابة البرنامج، التنفيذ وإيجاد الأخطاء (Implementation & debugged) - أنواع الأخطاء البرمجية (Types of errors)/الأخطاء الإملائية و القواعدية (Syntax errors)/الأخطاء التنفيذية (Run time errors)/الأخطاء المعنوية (Semantic errors)، الاختبار Testing، التوثيق والصيانة Documentation & maintenance	صفات البرنامج الجيد	التاسع - العاشر
التصميم من الأعلى إلى الأدنى (Top-down design) ، التصميم من الأدنى إلى الأعلى Bottom-up design	تصميم البرامج	الحادي عشر

<p>دورة حياة العملية (البرنامج) داخل الحاسوب (Process life cycle) التنفيذ أولاً/الاستعداد (Ready)/ثانياً/الانتظار (waiting) ثالثاً رابعاً الانتهاء (Complete) (Running)</p>	<p>دورة حياة العملية (البرنامج) داخل الحاسوب (Process life cycle)</p>	<p>الثاني عشر</p>
<p>البرامج الفرعية Subprograms ، الروتينات الفرعية Subroutines</p>	<p>الروتينات الفرعية Subroutines</p>	<p>الثالث عشر</p>
<p>مقدمة لأسلوب البرمجة المهيكلة / التراكيب المستخدمة في البرمجة المهيكلة تراكيب السلسلة Sequence / تركيب الاختيار IF-THEN –else تركيب التكرار Do-While</p>	<p>مقدمة لأسلوب البرمجة المهيكلة</p>	<p>الرابع عشر والخامس عشر</p>

المحتوى العلمي

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 ، أجره ساعة العمل) .

II. المعلومات (Information)

هي المعرفة التي تكونت نتيجة لتحليل البيانات المدخلة وإجراء العمليات المطلوبة عليها والمعلومات الناتجة تكون أكثر معنى من البيانات وتساعد متخذي القرارات في تحقيق أغراض معينة. ومن الأمثلة على ذلك (معدل الطالب ، دخل الموظف الأسبوعي) .

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 واتباع قواعد خاصة.

البرنامج

```
def fibonacci1(n):
    if n == 1 or n == 2:
        return 1
    return fibonacci1(n-1) + fibonacci1(n-2)
def fibonacci2(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 يمكنه 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 المستوى فإن اللغات المنخفضة المستوى تكون مُعدّة، 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 مُعقّداً 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 المستوى 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
nwn

mov ecx,res
mov edx,1
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)
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 للمبرمجين عند استخدامها في كتابة البرامج، حيث تُسمى الأوامر المكتوبة بها بالكود البرمجي، وتكون الرموز والمفردات المستخدمة في كتابة الكود قريبة من مفردات اللغة الإنجليزية، كما تُوفر اللغات العالية المستوى إمكانية إضافة الكثير من التعليقات، والشروح ضمن البرنامج الذي يعمل عليه المُبرمج؛ الأمر الذي يجعل تلك البرامج أسهل، ويزيد من قابليتها للقراءة، والتعديل من المُستخدم نفسه، أو حتى من المُستخدمين 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 account,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(account=1;account<=number;account++)
            factorial_number=factorial_number*account;
    }
    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 أحياناً بالتبادل مع نظام التشغيل ، لكن برنامج النظام يعني جميع البرامج المتعلقة بتنسيق عمليات الكمبيوتر. تتضمن برامج النظام نظام التشغيل ولكنها تتضمن أيضاً 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 آلية كلياً تقوم لاحقاً بحل بعض المشاكل المحددة وهي أنظمة تستجيب للبيانات المدخلة، وخلال وقت معقول، تحسب الرد الصحيح أو تقدر تقريبياً حلاً صحيحاً بما فيه الكفاية. إضافة إلى ذلك، يقضي الأشخاص في علم الحاسوب وقتاً كبيراً بشكل مُدهش في تقصي وإصلاح المشكلات في برامجهم (التنقيح البرمجي).

هناك عدة خطوات للعمل هي :

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

ولعمل الخطوات السابقة يجب الاخذ بنظر الاعتبار ما يأتي :

ا. فهم المشكلة ؟

ii. تقسيم المشكلة ؟

iii. حل المشكلة ؟

أنواع البيانات والمتغيرات المستخدمة في البرامج

1. Constant

$X = 7, h = 3.1, w = 'A', K = \text{"Hello World"}$

2. Variables

int a

float m

double f

Dim matrix (7,3) As integer

Dim StudentName As String

المخططات الانسيابية

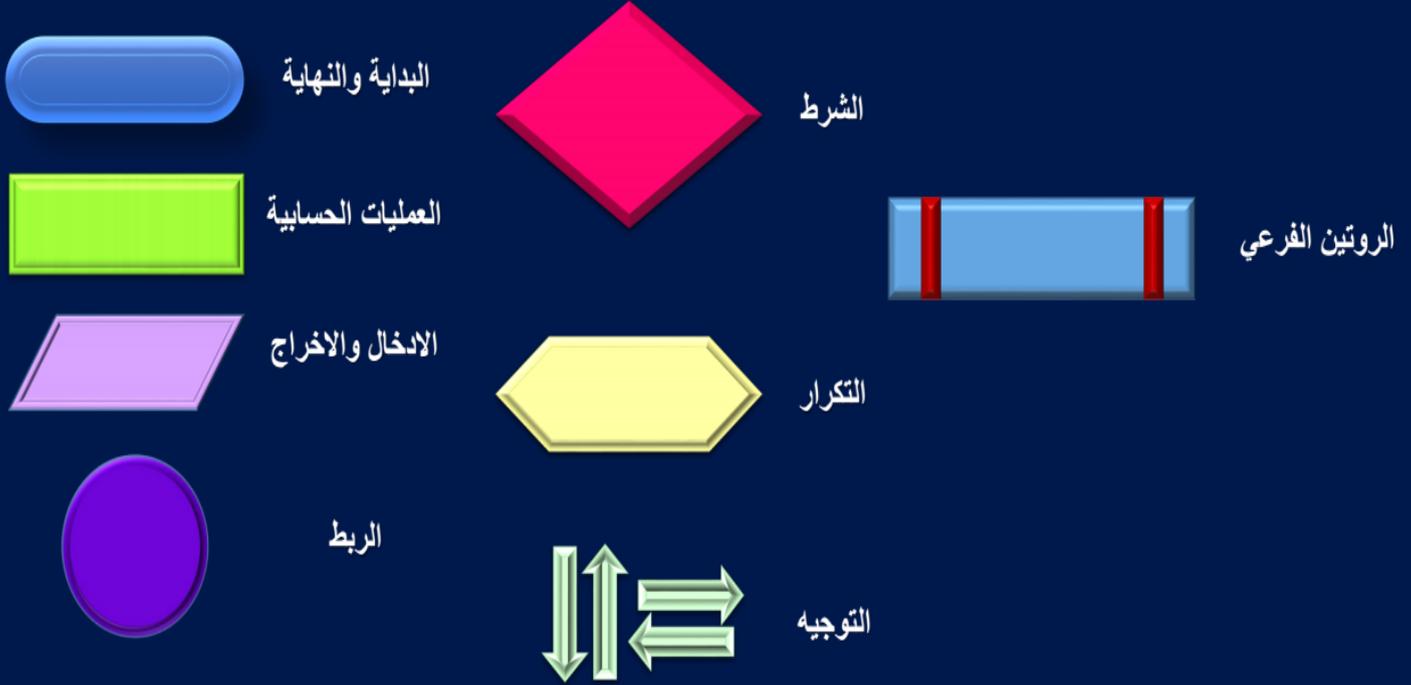
هو وصفا تصويريا أو تمثيل مصور للخوارزمية يوضح حل المسألة من البداية إلى النهاية بشكل أكثر وضوحا وأسهل فهما مع اخفاء التفاصيل لإعطاء الصورة العامة للحل، والمخطط الانسيابي يتكون نتيجة استخدام مجموعة من الاشكال كل شكل يوضح عملية معينة وتربط هذه الاشكال بخطوط وأسهم توضح اتجاه سير تنفيذ العمليات.

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

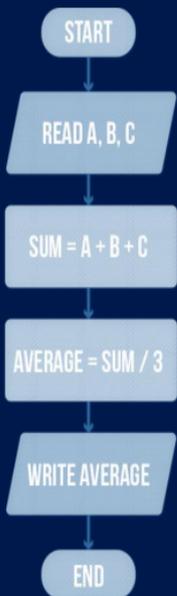
فوائد استخدام المخططات الانسيابية

1. تمكن المبرمج من الالمام الكامل بالمسألة 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 .
2. توضح سير العمليات وتسلسل تنفيذها .
3. تعتبر المخططات الانسيابية وسيلة مناسبة و مساعدة في كتابة ومتابعة البرامج ذات التفرعات الكثيرة .
4. تساعد المبرمج وبسهولة على تعديل برنامج ما ، فبمجرد النظر الى المخطط الانسيابي نظرة سريعة ، يدرك ماهية المسألة وإمكانية التعديل .
5. 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 .
6. تعتبر المخططات الانسيابية من الوسائل و الأدوات الهامة لتوثيق البرنامج .

الأشكال المستخدمة في المخططات الانسيابية

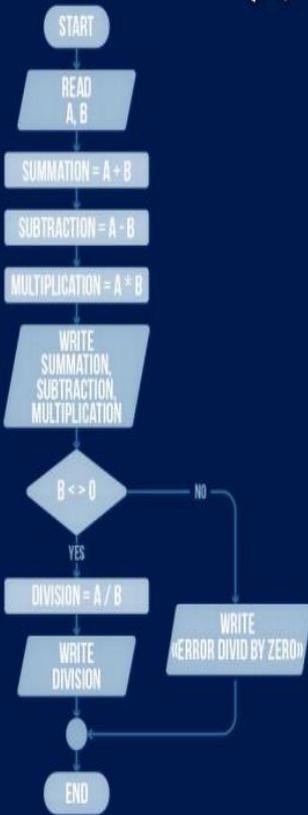


أنواع المخططات الانسيابية



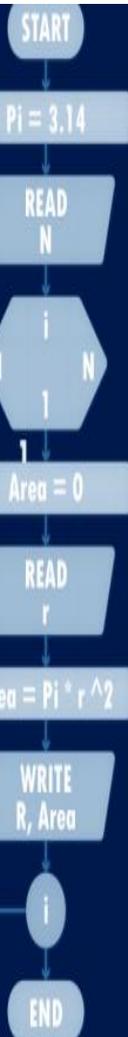
1. المخططات الانسيابية البسيطة (Simple flow chart)

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



II. المخططات الانسيابية المتفرعة (Branched flow chart)

يتضمن هذا النوع اتخاذ القرارات أو مفاضلة بين خيارين أو أكثر



III. المخططات الانسيابية ذات الحلقات (Loop flow chart)

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

أمثلة

١. ارسم المخطط الانسيابي لإيجاد مربع ومكعب العدد X .
٢. ارسم المخطط الانسيابي لقراءة قيمتين معلومتين واختيار أكبرهما.
٣. ارسم المخطط الانسيابي لترتيب قيمتين ترتيباً تصاعدياً.
٤. ارسم المخطط الانسيابي لحساب قيمة Y طبقاً للمعادلات الآتية : 
٥. ارسم المخطط الانسيابي لطباعة الأعداد الطبيعية المحصورة بين ١ إلى ١٠٠ ومربعاتها.
٦. ارسم المخطط الانسيابي لحساب معدل طالب في ٥ مواد دراسية وطباعة المعدل إذا كان الطالب ناجح .
٧. ارسم المخطط الانسيابي لحساب حاصل ضرب الحدود التالية : (5,10,15,20,.....,500).
٨. ارسم المخطط الانسيابي لطباعة X^2 لخمسة قيم لـ X .
٩. ارسم المخطط الانسيابي لإيجاد حاصل ضرب N من القيم .
١٠. ارسم المخطط الانسيابي لحساب عدد الأعداد الموجبة وعدد الأعداد السالبة وعدد الأصفار لـ ١٠٠ عدد .
١١. ارسم المخطط الانسيابي لقراءة أسم ومعدل ٢٠ طالب وإيجاد عدد الطلبة الناجحين والراسبين ونسبة النجاح الكلية .
١٢. ارسم المخطط الانسيابي لاختبار العدد X فيما إذا كان فردياً أم زوجياً .
١٣. ارسم المخطط الانسيابي لحساب عدد الأعداد الزوجية و عدد الأعداد الفردية لمجموعة تتكون من ١٠ أعداد .

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

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 وتنتظر 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.

• تعريف الخوارزمية Algorithm

هي مجموعة من الخطوات الرياضية والمنطقية المتسلسلة اللازمة لحل مشكلة ما ويجب ان تتوافر فيها الشروط الآتية :-

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

Characteristic of Algorithm خصائص الخوارزمية

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).

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

الحل:

- I. البداية -
- II. ادخل الأعداد X_1, X_2, X_3, X_4
- III. احسب المعدل $AV = (X_1 + X_2 + X_3 + X_4) / 4$
- IV. اطبع AV
- V. النهاية

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

تصميم الخوارزمية Algorithms design

تمتاز الخوارزميات بما يلي :-

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

أنواع الخوارزميات:

يمكن تصنيف الخوارزميات إلى:

Step 1

Step 2

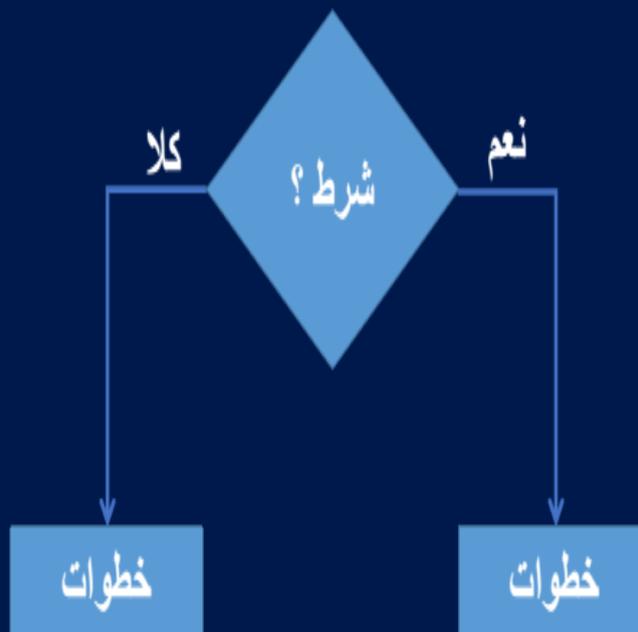
Step 3

1. الخوارزمية المتسلسلة المستقيمة Sequential

وهي الخوارزمية التي تحتوي على خطوات متسلسلة بحيث لا يؤدي تنفيذ أي منها إلى تفرع خارج التسلسل.

2. الخوارزمية الشرطية أو المتفرعة Conditional

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



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 . احسب معدل الطالب $AV = (D1 + D2 + D3 + D4 + D5) / 5$

٤ . إذا كان $AV \geq 50$ أطلع ناجح والا اطلع راسب

٥ . النهاية

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

الحل :

١ . البداية

٢ . ادخل قيمة العدد A

٣ . إذا كان $A \geq 0$ أطلع "العدد موجب" وإلا أطلع "العدد سالب"

٤ . النهاية

الخوارزمية المتكررة Repetition

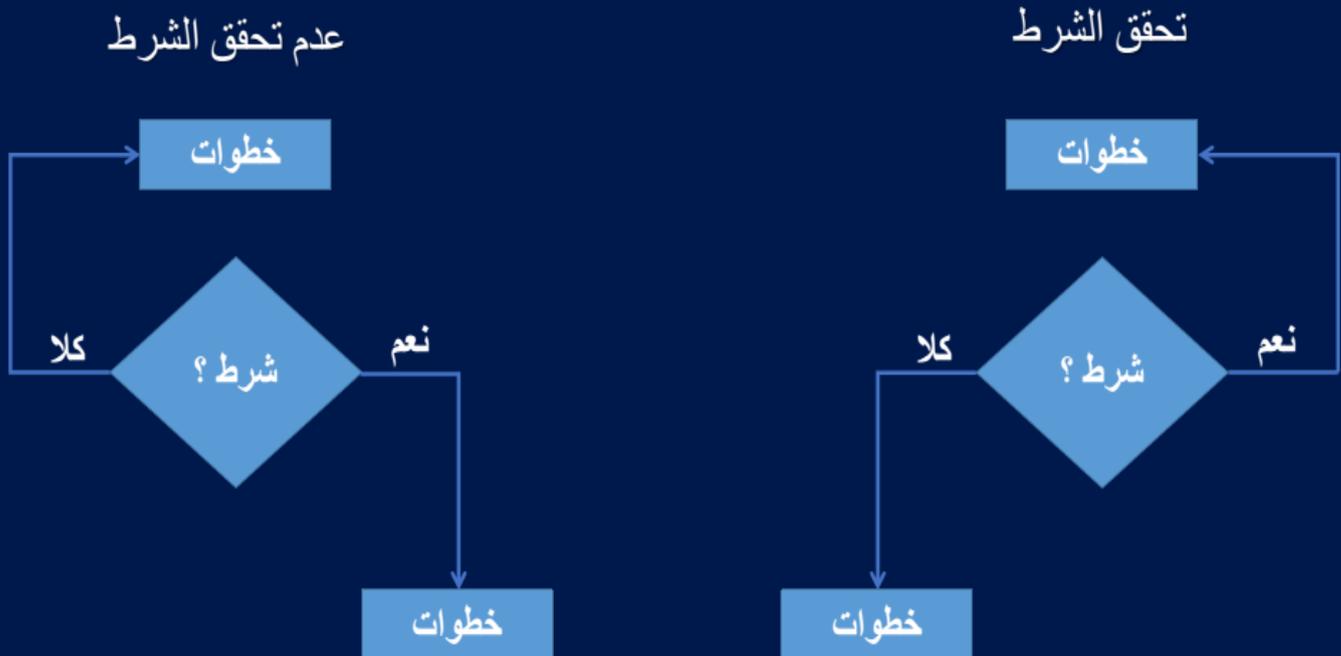
التكرار: عند حل بعض المسائل لابد من إعادة تنفيذ مجموعة من الخطوات عدة مرات وهذا التنفيذ المتكرر لهذه الخطوات هو الحلقة التكرارية Loop .

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

I. تحقق الشرط

II. عدم 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 .

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

الحل :

١ . البداية

٢ . افرض $I=2$

٣ . أطلع I

٤ . إذا كان $I \geq 37$ اذهب إلى الخطوة ٦

٥ . اجعل $I=I+2$ اذهب الى الخطوة ٣

٦ . النهاية

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

الحل :

١. البداية
٢. $K=10$ افرض
٣. $SUM = 0$ افرض
٤. احسب المجموع $SUM = SUM + K$
٥. اجعل $K=K+2$
٦. إذا كان $K \leq 20$ اذهب إلى الخطوة ٤
٧. أطلع 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 يأخذ 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 والفعال وتحقيقه باستخدام الخوارزميات و بناء المعطيات المناسبة.

مراحل تطور البرنامج : Program Development Stages

- I. توصيف المتطلبات Requirements Specification : فهم المدخلات و المخرجات.
- II. التصميم Design : تحديد العمليات الرئيسية التي تطبق و افترض وجود أجهزة معالجة لتنفيذ هذه العمليات.
- III. التحليل Analysis : المفاضلة بين الخوارزميات المتوفرة لحل نفس المسألة تبعاً لمقاييس مفاضلة منفق 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 معينة 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;
}
```

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، يعرف الخطأ المنطقي على أنه خطأ في البرنامج يجعله يعمل بطريقة غير صحيحة، ولكنه 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 وهكذا.

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.

أنواع الأخطاء البرمجية : Types of errors

ضمن عملية تطوير البرنامج application development يواجه المبرمجون عدة انواع من الأخطاء.
I. الأخطاء الإملائية والقواعدية Syntax Errors هي الأخطاء الأكثر شيوعا وانتشارا ولكنها سهلة العلاج، وهي التي تظهر اثناء كتابة البرنامج عندما لا يتم استخدام اللغة البرمجية بالشكل الصحيح.

فالأخطاء الإملائية تحدث غالبا بسبب وجود خطأ إملائي 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 الأخطاء 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.

II. الأخطاء 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 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.

III. الأخطاء 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 يظهر كنتيجة غير صحيحة وليس كخطأ واضح.

وهذه الأخطاء 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:

$$N = 0.5 * a * b$$

$$N = a^2$$

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.

اسباب الخطأ في البرنامج :

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

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

ادوات التنقيح التصحيح :

عند اكتشاف خطأ في البرنامج يجب على المبرمج ان يتتبع الخطأ في البرنامج ليكتشف موقعه ومن ثم تصحيحه وأكثر الاساليب شيوعاً في تتبع الخطأ هي وضع عبارات Print في مواقع متعددة في البرنامج حيث تطبع قيم المتغيرات التي هي موضع الشك ويستمر الفحص لحين اكتشاف الأخطاء.

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 والزمينية 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.

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.

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 نعتاد 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.

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

تعتبر طريقتنا 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.

التصميم من أعلى إلى الأسفل Top down :

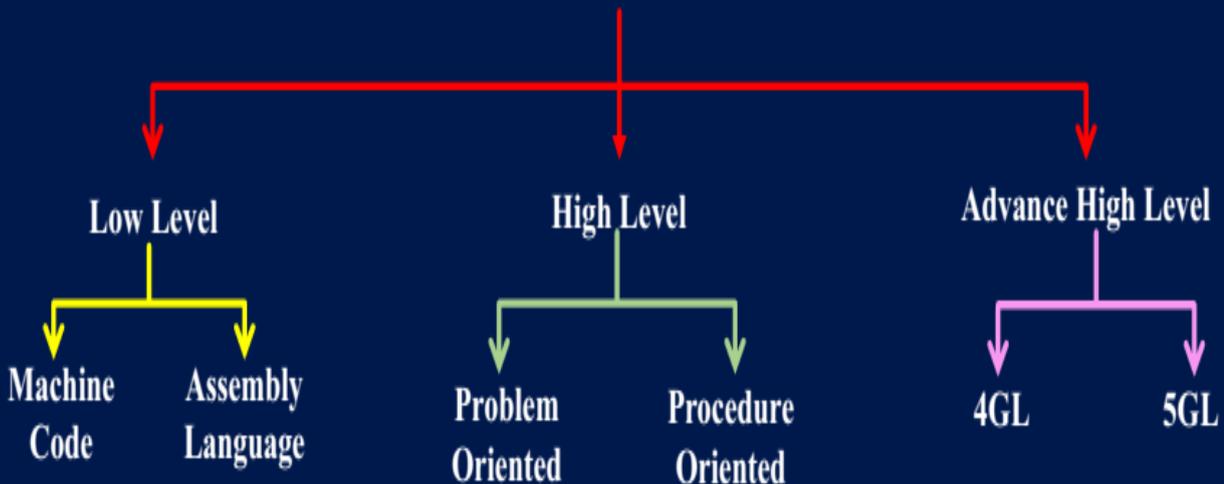
● إن طريقة التصميم من أعلى لأسفل (وتعرف أيضا بالتصميم المتدرج) هي أساسا عبارة عن تحليل النظام بهدف اكتساب المعرفة بالنظم الفرعية التركيبية الخاصة به. تصاغ النظرة العامة للنظام في طريقة التصميم من أعلى لأسفل بحيث لا تفصل أي أنظمة فرعية عن المستوى الأول. تتم بعد ذلك تنقية النظام الفرعي بتفصيل أكبر وأحيانا 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

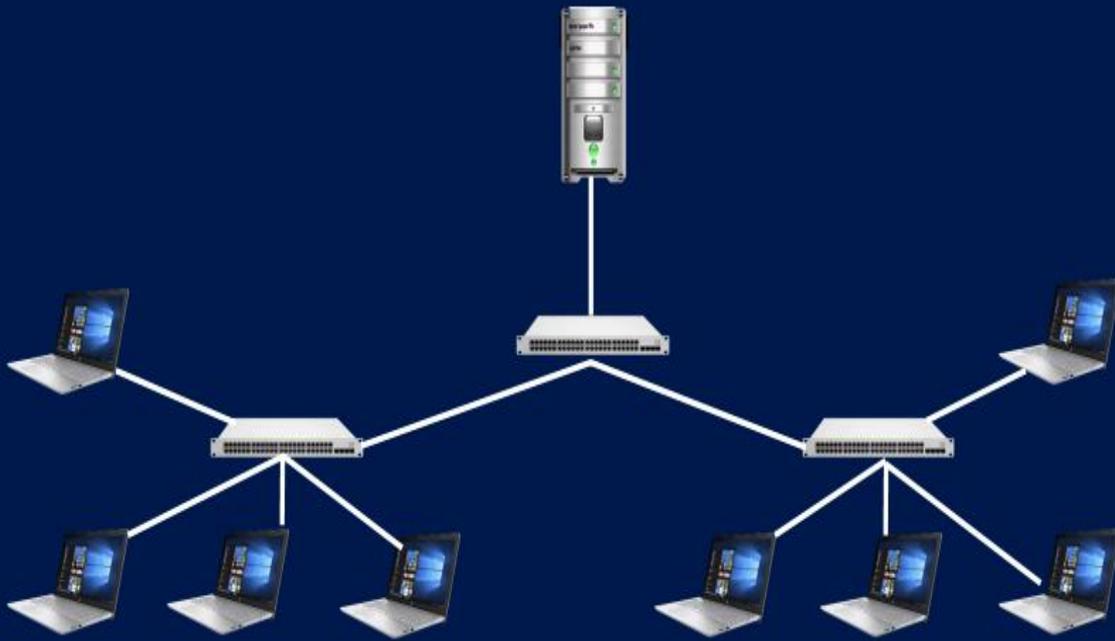
Computer Language



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:

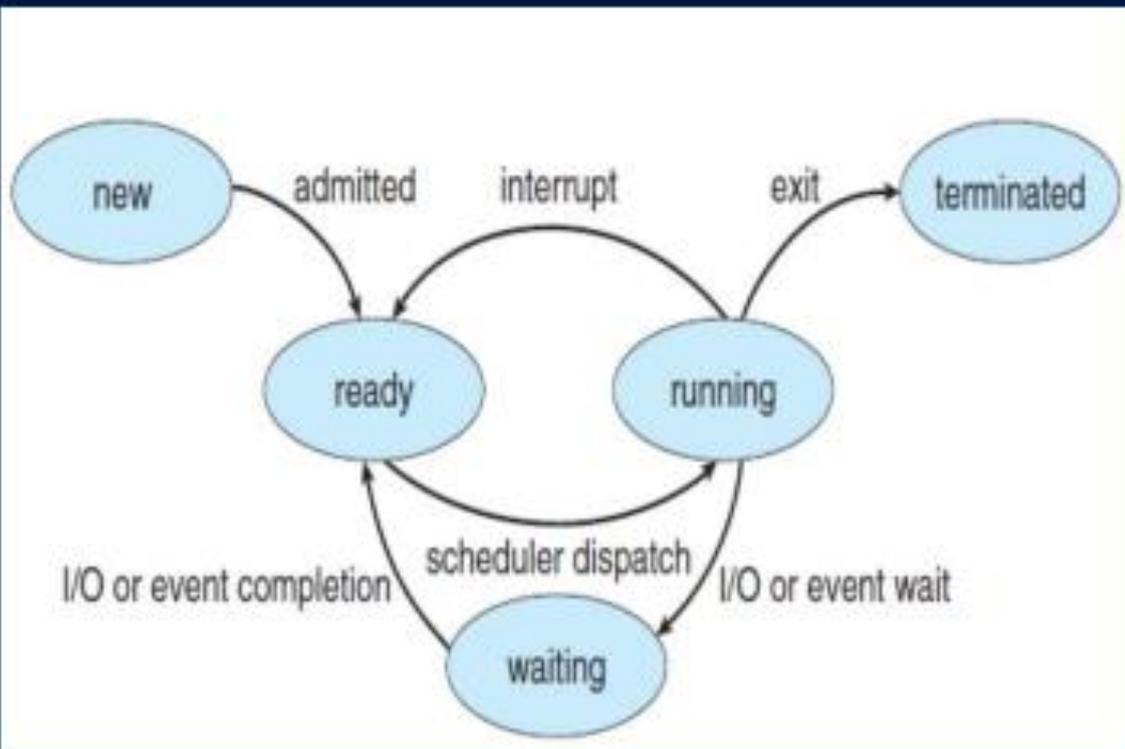
هي بناء البرنامج منذ لحظة ورود الفكرة في الذهن الى الخروج للعميل وخدمة ما بعد البيع. فإن دورة الحياة للنظام البرمجي طويلة وتتم بمراحل مختلفة وعديدة ليصل الى ايدي العملاء إذ إن كتابة الاكواد والشفرات ليست هي النقطة الرئيسية في دورة حياة البرنامج بل ممكن ان تكون هي اقلها وقتاً و مجهوداً.
تغير العملية حالتها 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 .

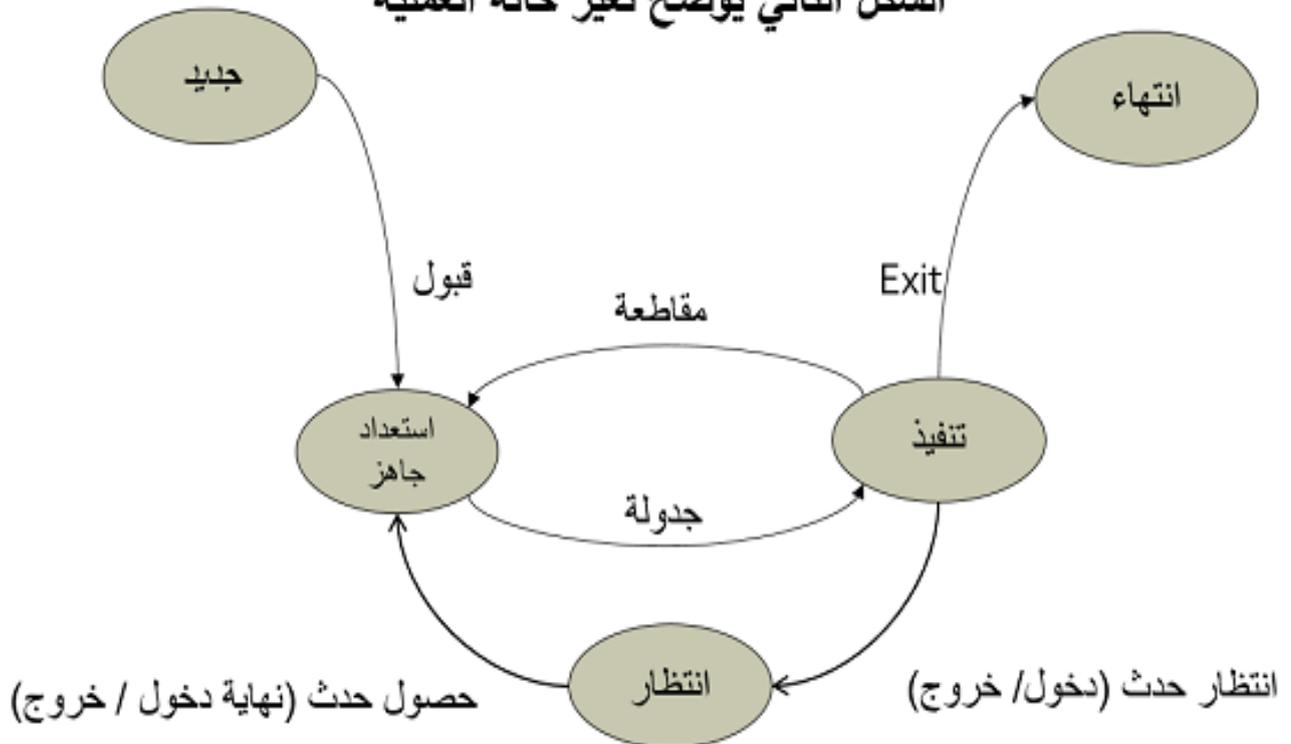
State Description

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.



الشكل التالي يوضح تغير حالة العملية



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) كأى 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 والانهاء الغير الطبيعي(يقرر نظام التشغيل اخراج العملية (Process) من النظام وسحب المعالج (C.P.U) منها لغرض منحه الى عملية اخرى تنتظر التنفيذ .

V. 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 الأمرية (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 . اجراءات.

2. Functions . توابع.

3. Methods . منهجيات.

4. Routines . روتينات.

5. Subroutines . روتينات فرعية.

ملاحظة:

توصيف حل المشكلة خطوة بخطوة تعد من مساوي 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 تجربها داخل البرنامج كثيرا و تكررهما 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 باسم 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 تلافى 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 (Visual Basic) VB للمبرمجين .
- ٢ . **البرامج الفرعية** و هي البرامج الخاصة بالمشروع المراد عمله و هي تشمل الدوال و البرامج الفرعية.

الفرق بين هذين النوعين هو ان الوحدات النمطية Module يمكن 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 (Name يكون كالتالي:

1. Name من خلال ذكر اسمه فقط.

2. Call Name() من خلال كتابة Call ثم اسمه يليه اقواس هلالية.

3. Name(3) من خلال اسمه مع المعاملات.

الاسبوع الرابع عشر - الخامس عشر: البرمجة الهيكلية

البرمجة الهيكلية، معروفة ايضاً باسم البرمجة المعيارية (Modular programming) ، احد اساليب كتابة الكود، هدفها جعل عملية كتابة وقراءة الكود اكثر سلاسة وسهولة. عن طريق تقسيم عملية كتابة الكود الى شكل يقرب الى الشكل الهرمي الى عناصر مستقلة، كل عنصر منها مستقل عن العناصر الأخرى وله خصائصه المميزة له. لا داعي للقلق اذا وجدت هذه الجمل صعبة الاستيعاب. فهي مجرد جمل نظرية مجردة على كل حال. دعك من هذه الجمل وتعالى معي نتعمق في هذا المفهوم بشكل عملي مبسط أكثر

سنتعرف فيما يلي على عناصر البرمجة الهيكلية

Block

البلوك هو عبارة عن مجموعة من الأوامر المتتالية – Sequential commands – التي يتم تنفيذها واحدة تلو الأخرى، من السطر الأول الى السطر الأخير. اذا تناولنا هذا العنصر من وجهة نظر برمجية، من لغة ++C ، فسيكون المثال على نوعية الكود في البلوك كالتالي:

```
Variable_X = Array[5];
```

```
Variable_X ++;
```

```
Sum = Variable_X + Variable_Y;
```

```
Output = Dummy_Function (Sum);
```

يبدو من الفقرة السابقة انها عبارة عن مجموعة من الأوامر، كاسترداد عنصر من مصفوفة، او جمع وطرح بعد المتغيرات. لذلك، المهم ان يتم تنفيذها بشكل متسلسل لأن -غالباً- ما يكون كل سطر معتمد على ما قبله.

Selection

باختصار، Selection، هي عملية التحكم في التدفق. او من وجهة نظر برمجية عملية، هي IF STATEMENT ، والتي يتم بناءً عليها اختيار في اي اتجاه سيذهب الكود بتاعك عند ظهور حدث معين.

Iteration

وهي عملية التكرار، او Loop ، وهي اللف حول مجموعة من العناصر لاختيار عنصر أو اكثر. عملية التكرار لها العديد من الأشكال في البرمجة الهيكلية، كال For Loop, While loop, While True loop وغيرهم. لكل منهم وظيفة معينة يتم اختيارها بناءً على المعطيات المقدمة.

Nesting

بالعربية، تسمى التداخل، وهي عبارة عن تداخل احد عمليات التحكم في التدفق -Flow Control- – فمثلاً، اكثر من جملة if متداخلين في نفس البلوك تسمى بالتداخل. التداخل لها تأثير كبير على كفاءة الخوارزميات، يمكنك الاطلاع على كفاءة الخوارزميات بشكل عام من هنا. ليست بالضرورة ان تكون جملة IF ، ربما تكون دالة داخل دالة، او loop بداخل اخرى.

انواع البرمجة المهيكله

- 1_ المهيكله المستقيمه:- ونقصد ب المستقيمه اي انها تحتوي ع خطوات متسلسله لا يمكن ان يؤدي اي منها ال تفرع خارج التسلسل اي بمعنى
- خطوة (1)
- خطوة (2)
- خطوة (3) .

مثال:- امتحن طالب سادس 3 مواد فيرنا رياضيات احياء صع الحوارر منه المناسبه للحصول على المتوسط الحسابي للدرجات؟

الحل:-

المعطيات:- x, y, z

مطلوب:- $R = (X + Y + Z)$

10_start □

20_read □

30_r(x+y+z)/3 □

40_print r □

50_end □

2_المهيكله الشرطية

وهي التي تحتوي على خطوات وأوامر شرطية اي بمعنى شرط التنفيذ اي اذا شرط التنفيذ تكون الاجابة بنعم تنفذ الخطوات متسلسلة..

If $R \geq 50$ then print end

```
10_      start
20_      read x,y,z
30_      r(x+y+z)/3
40_if  r>50 go to 50
50_      print R
60_      end.
```

سؤال:- عددان A B اطبع العدد الاكبر.....

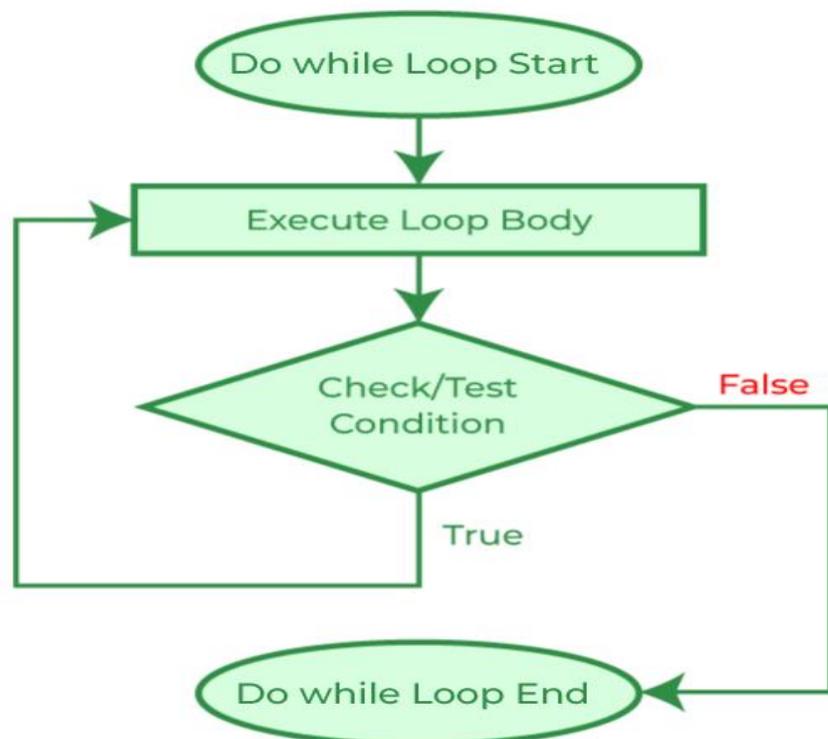
```
10_ start
20_ read a,b
30_ a=max
40_ if a>b then go to 50
50_ print a
60_ end
```

3_ المهيكلة المتكرره

ويتم في هذا النوع من الهياكل تكرار تنفيذ مجموعه من الخطوات حسب ما تم ذكره في السؤال

Syntax of do...while Loop in C

```
do {  
    // body of do-while loop  
} while (condition);
```



```
// C Program to print multiplication table using do...while
```

```
// loop
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int N = 5, i = 1;
```

```
do {
```

```
printf("%d\t\t%d\t=\t%d\n", N, i, N
```

```
* i);
```

```
} while (i++ < 10);
```

```
return 0;
```

```
}
```

مثال:- اكتب حوار برمجة لتابع قسمه مجموعة اول خمس اعداد
زوجية على مجموع اول خمسة اعداد فردية 10-1
؟

المعطيات:- الاعداد الفردية $s=2$ $s=s+2$
الاعداد الزوجية $z=0$ $z=z+2$
المطلوب:- جمع الاعداد الفردية $m=m+2$
الاعداد الزوجية $n=n+z$

```
10_start  
20_s=1  
30_s=s+2  
40_if s>=10 then go to 30  
50_m=m+s  
60_z=0  
70_z=z+2  
80_if z>+10 then go to 70  
90_n=n+z  
100_x=M/n  
110_printx  
120_end
```

سؤال:- لديك خمسة عشر عددا استخرج الاعداد السالبة والموجبة والتي تساوي صفر؟؟

الحل:- المعطيات خمسة عشر عددا L=15

1_ الاعداد السالبة L=0

2_ الاعداد الموجبة N=0

3_ التي تساوي صفر Z=0

```
10_start
20_L=0,m=0,n=0,z=0
30_if L<=15 then go to 20
40_read A
50_if A>0 Then go to 40
60_m=m+a
70_n=n+a
80_z=z+a
90_print m,n,z
110_end
```

سؤال:- اجمع الاعداد معدومه تبادلي 2- ورتابه 0.2 ل 12 عدد؟؟؟؟

الحل:-

```
10_start
20_s=0
30_for I=-2 to 12 stop 0.2
40_read a
50_s=s+a
60_print s
70_end
```

المخطط الانسيابي:- عبارة عن خارطة نصف بسلسل الخطوات المتبعة لكل مساله وذلك باستعمال اسكال مربعي عليها

س:- عدد فوائد خرائط السير والمخططات الانسيابية؟؟؟

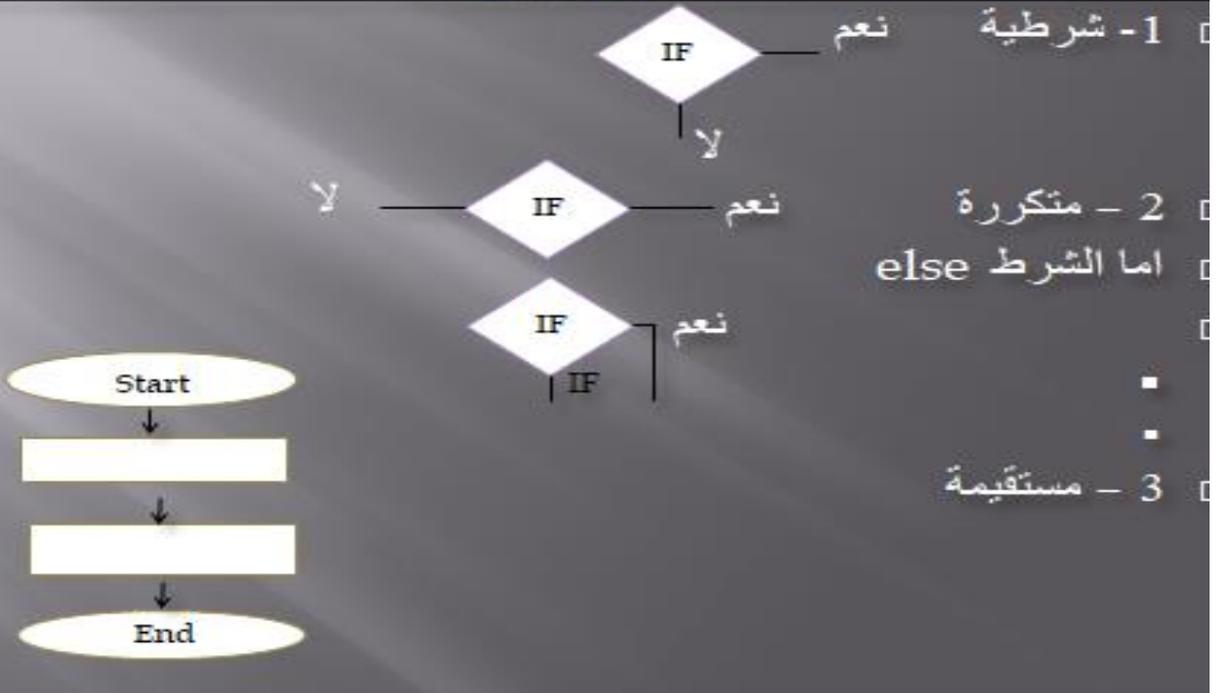
- 1_ تعطي خرائط السير صورة واضحة وبسيطة عن طبيعه الخطوات المتبعة لحل المسألة
- 2_ تعتبر خرائط السير ضرورية في حالة المسائل التي تحتوي ع شروط او ابدال
- 3_ تسهل خرائط السير البرامج او النظام من قبل المستخدمين
- 4_ تعتبر خرائط السير احد الخطوات المهمة لتوثيق البرامج
- 5_ تسهيل عملية مراجعه البرامج من اجل التعديل او بأستئناف الاخطاء الموجودة لتصحيحها

$$F(x) = \begin{cases} x^2 + x + 1 & x \geq 0 \quad \square \\ x - x^2 - 1 & x < 0 \end{cases}$$

```
10 - Start
20 - Read F ( x )
30 - IF F ( x ) > = 0 Then Go to
40 - Print F ( x ) = X2 + X + 1
50 - Print X - X2 - 1
60 - End
```

```
نعم If x > 0 Then Print X2 + X + 1
لا X - X2
```

س / عدد انواع الحواررمات موضح ذلك بمحطظ انسنادى لكل حواررمه ؟



سؤال:- عدد مراحل تطور البرنامج؟؟

1_ يجب ان يكون البرنامج قليل امتغيرات

2_ يستطيع البرنامج اكتشاف الاخطاء في البيانات المخصصه له

3_ في حالة وجود اخطاء في البرنامج يتم حله بسهولة

صفات تطور البرنامج...؟

الصفات:-

- 1_ ان يكون البرنامج سهلاً
- 2_ سهولة القراءة والفهم ونعني بذلك سهولة وجود الاخطاء
- 3_ سهولة الاختيار اي اختيار البرنامج بيانات معينه

.....

اما مراحل التطور كالاتي...

- 1_ تحليل وتعريف....
- 2_ التصميم....
- 3_ التنفيذ وايجاد الاخطاء....
- 4_ الاختبار

س:- عدد طرق تصميم البرنامج وشرح الطريقة المناسبة واذكر السبب؟؟

الحل:-

- 1_ من الاسفل الى الاعلى
 - 2_ من الاعلى الى الاسفل
 - 3_ طريقه الدمج
- افضل طريقه هي الدمج لعدده اسباب..
- 1_ سرعه الحصول على النتائج
 - 2_ تتم في عمليات التكرار التي تحصل في بعض البرامج
 - 3_ اكتشاف الاخطاء التي تحصل وحلها بسرعه
 - 4_ لا تحتاج على تسلسل في العمليات

س:- عرف البرنامج الفرعي والبرنامج الرئيسي؟؟؟

- ان البرنامج الذي يحتوي على مجموعه من البرنامج الفرعية يدعى main program
- اي برنامج رئيسي وكل فرع من هذه الفروع تؤدي عملية معينه
- س:- ماهي الغاية من استخدام البرنامج الفرعي داخل البرنامج الرئيسي؟؟؟
- الحل:-
- ان هذه الاسلوب يوفر امكانية الاستدعاء للبرنامج الفرعي الواحد لاكثر من مره داخل البرنامج وبالتالي يوفر مساحه تخزين....

المصادر الأساسية:

- Introduction in Data structure and algorithms,tomas Itch, tcharls lyrson,2016
- Grokking algorithms :An illustrated guid for programs , 2015.
- Introduction in the design and analysis of algorithms, Anany Levitin, 2015
- المصادر المقترحة:
- (يذكر هنا بعض المصادر المقترحة التي لم تدخل ضمن مصادر بناء الحقيبة لإعطاء الفرصة للمتعلم لإغناء المعلومات و للمزيد من المعرفة)
- روابط مقترحة ذات صلة:
- يذكر هنا بعض الروابط ذات الصلة بالمواضيع الخاصة بالمحتوى لزيادة المعرفة او المهارة كان يدر بعض روابط اليوتيوب لمحاضرات ضمن نفس المواضيع او تقدم شرح مفصل لبعض الفقرات التي لم يتم تغطيتها في المحاضرة ويفضل. ان تكون بصيغة رمز الاستجابة السريع QRC



•
•