نظام الحاسب :- Computer System

هو مجموعة من الآلات و البرامج المصممة لتنفيذ مهمات معالجة البيانات , فالآلات هي الأجهزة الخاصة بالحاسب (Hardware) و البرامج التي توجه الآلات هي البرامجيات (Software) .

البرامجيات : Software

البرامجيات عبارة عن مجموعة من البرامج تعمل لتنفيذ مهمة معينة. والبرنامج عبارة عن مجموعة من الأوامر مكتوبة بلغة معينة .و تقسم البرمجيات إلى قسمين رئيسيين هما:

1. برامجيات النظام : System Software

وهي البرامج التي يستخدمها الحاسوب للتحكم والتوجيه ُتبرمج من قبل الجهة المصنعة لأجهزة الحاسوب .ُتخزن هذه البرامج بصورة دائمية داخل ذاكرة الحاسوب إذ لا يمكن حذفها أو تغير محتوياتها .وتقسم برمجيات النظام إلى ثلاث مجاميع:

- **أنظمة التشغيل**(Operating Systems) : هي البرامج المسؤولة عن أدارة المكونات المادية للحاسوب .
- **المترجمات** (Compilers) : هي البرامج المسؤولة عن ترجمة أوامر البرنامج المصدر إلى لغة الآلة (Machine Language) التي يستطيع الحاسوب فهمها.
- **البرامج الخدمية** :هي البرامج التي تؤدي وظيفة أو وظائف محددة ، برامج كشف ومعالجة الفيروس.

2. برامجيات التطبيقات Applications Software

هي برامج أما أن يقوم المستخدم بكتابتها مستخدمًا إحدى لغات البرمجة ، أو يقوم المستخدم بشرائها كبرامج جاهزة مثل معالج النصوص والعاب الحاسوب وغيرها.

لغات البرمجة Programming Languages

يعرف الحاسوب على انه مجموعة من الأجهزة الالكترونية التي يتم التحكم في أدائها بواسطة البرامج .وهذه البرامج تكون مكتوبة بلغات خاصة تسمى لغات البرمجةProgramming Languages. لذا لغة البرمجة هي مجموعة من الأوامر الخاضعة لقواعد معينة يفهما الحاسوب ويتم إدخالها من خلال وحدات الإدخال وتصنف إلى مستويين:

أو لا" : لغات المستوى الواطئ : Low-level Languages

و يقسم هذا المستوى إلى نوعين:

1. **لغة الآلة Machine Language** :

ظهرت هذه اللغة مع أول ظهور للحاسوب ، إذ أنها تستخدم الأرقام الثنائية (0 ,1) ، التي يفهمها الحاسوب مباشرة بدون وسيط مما جعلها لغة صعبة على المستخدم البسيط ، لذلك استخدمت فقط من قبل الشركات التي كانت تصنع

أجهزة الحاسوب حيث كان لكل حاسوب لغة آلة خاصة به .ومع التطور الهائل في كتابة البرامج باستخدام لغات برمجة حديثة استمرت هذه اللغة كونها اللغة الوحيدة التي يفهما الحاسوب حيث يتم ترجمة جميع اللغات المختلفة إلى هذه اللغة باستخدام برامج خاصة يطلق عليها المترجمات أو المفسرات.

2. **لغة التجميع Assembly Language** :

وهي لغة أعلى مستوى من لغة الآلة وأسهل منها ، حيث تستخدم فيها الرموز أو مجموعة من الحروف للدلالة على أمر معين والتي يطلق عليها (Mnemonic codes)يتم تحويل أوامر هذه اللغة إلى لغة الآلة عن طريق برنامج خاص يسمى المجّمع (Assembler). تمتاز هذه اللغة بسرعة التنفيذ والتفاعل المباشر مع الكيان المادي للحاسوب.

ثانيا" : لغات المستوى العالي High-level Languages

نتيجة لتطور تقنيات الحاسوب المادية (Hardware) ولحاجة المستخدمين للغات برمجة ذات صيغة مفهومة ، ظهرت لغات على أنها لغات ذات مستوى عالي .وأوامر هذه اللغة تستخدم كلمات اللغة الانكليزية المفهومة من قبل المستخدم ويمكن استخدامها على أجهزة حاسوب مختلفة بغض النظر عن الشركات المصنعة لها .ومن اهم هذه اللغات لغة LISP , Prolog , BASIC , FORTRAN , Pascal , C++ و تعد هذه اللغة كأساس للغات البرمجة الكائنية الموجهة (Object-oriented language) وهذه اللغات تتعامل مع الأوامر والبيانات على شكل كتل أو مجاميع. و اللغات المرئية (Visual Programming) والمصممة من قبل شركة Microsoft والتي تعمل تحت بيئة Windows . مكونات البرنامج تعتمد على أسلوب القوائم (Menu choices) والإيقونات (Icons) والأزرار(Buttons). ومن أمثلة اللغات المرئية Visual Basic , Visual C++ , C # , وغيرها من اللغات الحديثة التي تخص قواعد البيانات مثل لغة Visual FoxPro.

لغات الانترنت المرئية هي لغات مطورة للغات المرئية تتعامل بشكل مباشر مع بيئة الانترنت وتسمى (.Net) وتستخدم بصورة أساسية في كتابة تطبيقات الانترنت.

أنظمة التشغيل Operating Systems

هي البرامج المسؤولة عن أدارة المكونات المادية للحاسوب .إن نظام التشغيل يمثل حلقة الاتصال بين المستخدم والحاسوب ، فالمستخدم يتعامل مع مكونات الحاسوب المختلفة من خلال نظام التشغيل الذي ينظم العمليات التي يقوم بها الحاسوب .ولهذا يمكن تشبيه نظام التشغيل بالمدير(Manager) الذي يدير مشروعًا معينًا ، ولكي يكتب لهذا المشروع النجاح وبكفاءة عالية فان على المدير أن يقوم بتنظيم أنشطة المشروع والموارد المستخدمة في تنفيذ هذه الأنشطة بهدف تنفيذ المشروع بأقل وقت ممكن وبأكفأ استغلال للموارد المتاحة. يعد نظام التشغيل (Microsoft Disk Operating System) MS-DOSالأساس لأنظمة التشغيل للحاسوب الشخصي (Personal Computer- PC). ويتميز هذا النظام بأنه سهل الاستخدام ويعمل على أجهزة الحواسيب ذوات المعالجات القديمة مثل معالجات عائلة (80x) .والتي تعتبر ارخص الأجهزة وتلبي متطلبات الاستخدامات الشخصية , حيث أن نظام التشغيل MS-DOS يسمح بتشغيل برنامج تطبيقي واحد.

ونتيجة للتطور التكنولوجي لمكونات الحاسوب المادية والبرمجية أدى إلى تطور أنظمة تشغيل جديدة تتلائم مع طبيعة هذا التطور ، ومن هذه الأنظمة نظام التشغيل ويندوز windows و نظام التشغيل ماكنتوش(Macintosh) و نظام التشغيل يونيكس (UNIX).

البيانات : Data

غالبا ما تشير كلمة بيانات (البيانات المدخلة) إلى مجموعة من الحقائق الضرورية التي تعبر عن مواقف و أفعال معينة سواء أكان ذلك التعبير بأرقام أو رموز أو كلمات أو إشارات أو ما شابه ذلك ولا تفيد هذه البيانات في شئ و هي على صورتها الحالية , و من الأمثلة على ذلك :-

- درجة الطالب في مقرر ما .
- عدد ساعات العمل لموظف في الأسبوع.
- أجرة الساعة الواحدة لعمل الموظف.
- عدد الرحلات الجوية بين مدينتين.

و البيانات على أنواع منها:-

❖ الثوابت Constants : هو مقدار غير متغير و ذا قيمة ثابتة خلال فترة تنفيذ الخوارزمية او البرنامج. و يمكن التمييز بين نوعين من الثوابت :

- الثوابت العددية Numerical Constants و هي سلسلة من الأرقام (0 إلى 9) تستخدم في تمثيل المعطيات العددية. و يمكن أن يكون الثابت العددي على عدة أنواع منها الصحيح و الحقيقي و الأسي .
- ثوابت السلسلة الرمزية String Constants و هي عبارة عن سلسلة من الرموز تكتب بين علامتي اقتباس . يستخدم هذا النوع في تمثيل المعطيات غير العددية .

❖ المتغيرات Variables : هو اسم رمزي يمثل موقعا تخزينيا" في وحدة التخزين و تتغير قيمة المتغير عدة مرات أثناء تنفيذ البرنامج و هناك نوعين من المتغيرات:

✓ المتغيرات العددية Numerical Variables و هي المتغيرات التي تستخدم لتخزين القيم العددية فقط .

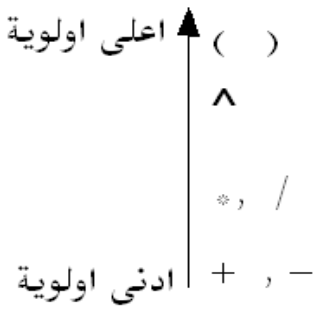
✓ المتغيرات الحرفية (الرمزية) String Variables و تستخدم لتخزين القيم الحرفية .

التعابير:- Expressions

تقسم إلى نوعين :-

التعابير الحسابية : Arithmetic Expressions

هو مجموعة من الحدود تتكون من ثوابت عددية و متغيرات عددية بالإضافة إلى رموز العمليات الحسابية (+ , - , * , / , ^) . وإذا اجتمع في التعبير الحسابي أكثر من عملية واحدة فان تسلسل تنفيذ العمليات يتم من اليسار إلى اليمين وفقا لسلم الأولويات :



- التعابير المنطقية : Logic Expressions

تستعمل التعابير المنطقية مع عبارات الشرط (IF) للمقارنة وتستخدم للمقارنة العوامل المنطقية مثل
(< , >, <= , >= , #, =, …)

المعلومات : Information

هي المعرفة التي تكونت نتيجة لتحليل البيانات المدخلة بإجراء العمليات المطلوبة عليها و المعلومات الناتجة تكون أكثر معنى من البيانات و تساعد متخذي القرارات في تحقيق أغراض معينة, و من أمثلة ذلك :-

- معدل الطالب.
- دخل الموظف الأسبوعي.

معالجة البيانات : Data Processing

و تعني إخضاع البيانات المدخلة للتحليل و إجراء مجموعة من العمليات عليها باستخدام وسائل معينة بغرض الحصول على معلومات مفيدة في اتخاذ القرارات, و الشكل التالي يوضح العلاقة التي تربط بين البيانات , و المعلومات , ومعالجة البيانات:-

مثال

أفترض أن المطلوب هو حساب معدل الطالب في مجموعة من الدروس فان :-

- البيانات المدخلة تتمثل في درجات الطالب الواحد في كل الدروس.
- عمليات المعالجة تتمثل في مجموعة من الأوامر المتتابعة:-
  - ☒ إيجاد مجموع درجات الطالب.
  - ☒ إيجاد معدل الطالب.
- المعلومات الناتجة تتمثل في إظهار (طباعة) معدل الطالب.

الخوارزميات : Algorithms

تتمثل الخطوات المهمة في تحليل المسائل بما يلي:-

1. فهم المسألة :-

يتطلب ذلك قراءتها و دراستها جيدا" حيث أن محاولة حل مسالة دون استيعابها يؤدي بالتأكيد إلى نتائج خاطئة أو ربما نتائج ناقصة أو عدم الحصول على أي نتائج مطلقا".

2. تحديد معطيات المسألة :-

يجب توفير البيانات التي يتطلبها حل المسالة حيث أن هذه البيانات على درجة كبيرة من الأهمية بالنسبة لمعالجة الخوارزمية و تسمى المعطيات (Inputs).

3. تحديد النتائج المطلوبة :-

تتمثل هذه الخطوة بتحديد الهدف من حل الخوارزمية , أي تحديد النتائج التي يفترض أن نحصل عليها و تسمى هذه النتائج بالمخرجات (Outputs).

4. طريقة الحل :-

يجب وضع خطوات مناسبة للحل بالاعتماد على المعطيات وصولا إلى النتائج.

تعريف الخوارزمية : Algorithm Definition

هي وصف مفهوم لطريقة حل المسائل بخطوات محددة تتوفر فيها قواعد أساسية ثابتة تستطيع الآلة أو الإنسان فهمها ثم إطاعتها (تنفيذها).

و سميت بهذا الاسم نسبة إلى العالم المعروف محمد بن موسى الخوارزمي و نلاحظ من تعريف الخوارزمية إنها ليست بالضرورة مخصصة لحل المسائل الحسابية , وإنما يمكن أن تكون هناك خوارزمية لأي عمل معين.

يجب أن يكون لكل خوارزمية الخصائص (الصفات)التالية:-

1. الخوارزمية تكون دقيقة :-

كل خطوة في الخوارزمية يجب أن تكون واضحة تماما دون غموض أو شك عن العملية المقصودة بتلك الخطوة. وان تقسم إلى خطوات معينة و متتالية تؤدي إلى النتيجة المطلوبة , وتصلح لحل جميع المسائل من نفس النوع أو الطراز.

2. الخوارزمية يجب أن تكون منتهية:-

عندما ينفذ شخص ما أو الآلة خوارزمية معينة يجب الوصول إلى في النهاية إلى نقطة تكون عندها المهمة قد اكتملت أي لا يستمر تنفيذ الخوارزمية إلى ما لانهاية. وان يكون الزمن اللازم لتنفيذها معقولا لأنه من المنطق أن نرفض الخوارزمية التي تأخذ وقتا طويلا في التنفيذ مهما كانت درجة صحتها.

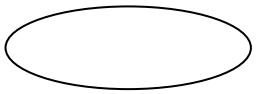
3. الخوارزمية يجب أن تكون كاملة و فعالة:-

الخوارزمية تأخذ بنظر الاعتبار جميع الظروف والاحتمالات التي يمكن أن تجابه طريق التنفيذ , كما لا يجب أن تحتوي على تعليمات يصعب على الشخص أو الآلة تتبعها لتنفيذ الخوارزمية.

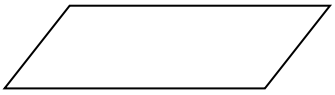
المخططات الانسيابية : Flow Charts

هي طريقة أخرى لعرض الخوارزمية و يتكون من مجموعة من الأشكال الهندسية التي يتصل بعضها بالبعض الآخر بواسطة الخطوط و تحتوي الأشكال بداخلها على التعليمات . يتم تتبع الخطوط من شكل إلى آخر لتنفيذ التعليمات داخل كل شكل كلما وجد , ويمكن تلخيص مجموعة الأشكال كما يلي:-

1. رمز البداية والنهاية: عبارة عن شكل بيضوي تكتب فيه كلمة بداية (Start) أو كلمة نهاية (End) و يشير إلى نقاط البداية و النهاية في المخطط الانسيابي ويكون للمخطط نقطة بداية واحدة فقط و لكن ربما يكون له أكثر من نقطة نهاية ممكنة والشكل هو :

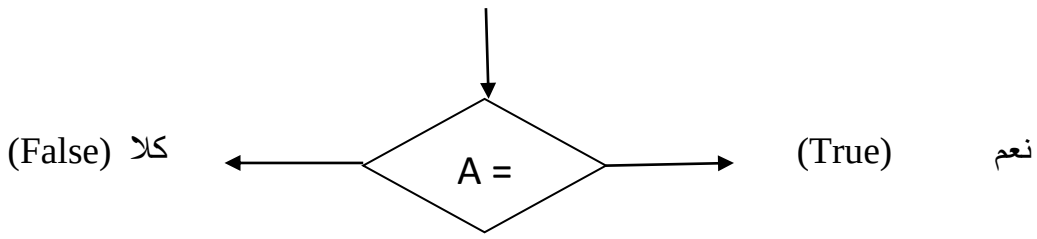


2. رمز الإدخال / الإخراج: (بدون تحديد وسيلة الإدخال و الإخراج) يستخدم شكل متوازي الأضلاع في تمثيل عملية الإدخال / الإخراج للبيانات أو المعلومات كما في الشكل التالي:-

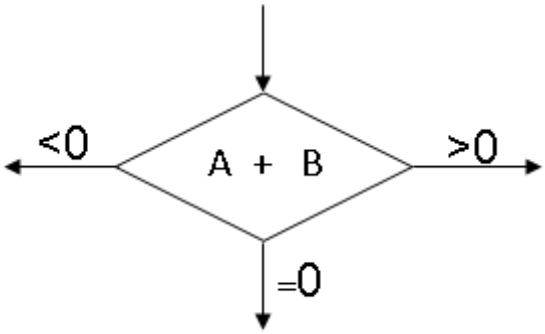


3. رمز العمليات الحسابية: يستخدم شكل المستطيل لبيان وجود عملية حسابية يراد انجازها و الشكل هو:-

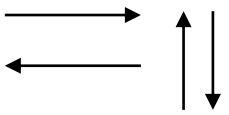
4. رمز القرار المنطقي: يستخدم الشكل المعيني لبيان اتجاه التسلسل المنطقي لتنفيذ العمليات حسب جواب الشرط المتضمن في عملية منطقية داخل الشكل المعيني وقد يكون للشكل اتجاهين الأول إذا كان جواب الشرط نعم (True) و الثاني إذا كان جواب الشرط كلا (False) كما في الشكل التالي:-



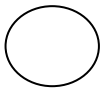
أو يكون له ثلاث اتجاهات تعتمد على قيمة التعبير المستخدم في الشرط مثلا:-



أسهم الاتجاهات: تستخدم الأسهم بالاتجاهات الأربعة لبيان انسياب السيطرة في الخوارزمية و كما في الشكل التالي:



5. رمز الاتصال: تستخدم الدائرة في وصل نقاط تتحد عندها مسارات مختلفة معا و الشكل المستخدم هو الدائرة:



فوائد المخططات الانسيابية:-

يمكن تلخيص الفوائد التي يمكن الحصول عليها من جراء استعمالنا للمخططات الانسيابية كما يلي:

1- تعطي المخططات صورة واضحة و بسيطة عن طبيعة الخطوات المتبعة لحل المسالة.

2- تعتبر المخططات ضرورية في حل المسائل التي تحتوي على قرارات وبدائل كثيرة لأنه من الصعب على المبرمج أن يقوم بكتابة البرنامج دون توفر صورة واضحة و بسيطة عن طبيعة الخطوات المتبعة لحل المسالة.

3- تسهيل دراسة البرامج أو النظام من قبل المستفيدين أو الدارسين.

4- تعتبر المخططات احد الخطوات المهمة في توثيق البرنامج.

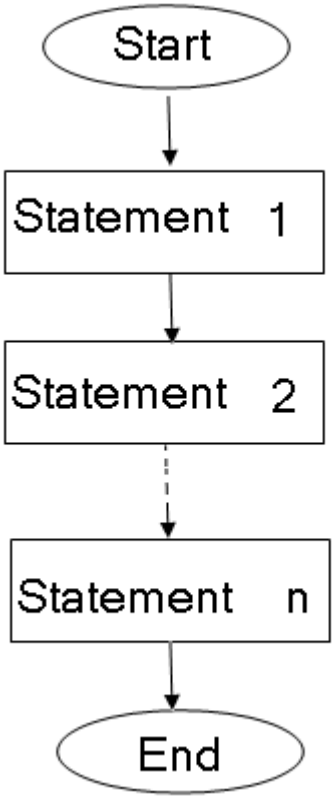
5- تسهيل عملية مراجعة البرنامج أو النظام من اجل التعديل أو لاكتشاف الأخطاء الموجودة فيه لتصحيحها.

أنواع الخوارزميات :-

تقسم الخوارزميات الى عدة أنواع هي :-

1. الخوارزميات المتسلسلة : Sequential Algorithms

في هذا النوع من الخوارزميات لا نجد أي تفرعات أو تكرارات و إنما تتم كتابة و تنفيذ التعليمات الواحدة تلو الأخرى . و يمكن تمثيلها بمخطط انسيابي بشكل عام بالشكل التالي:-



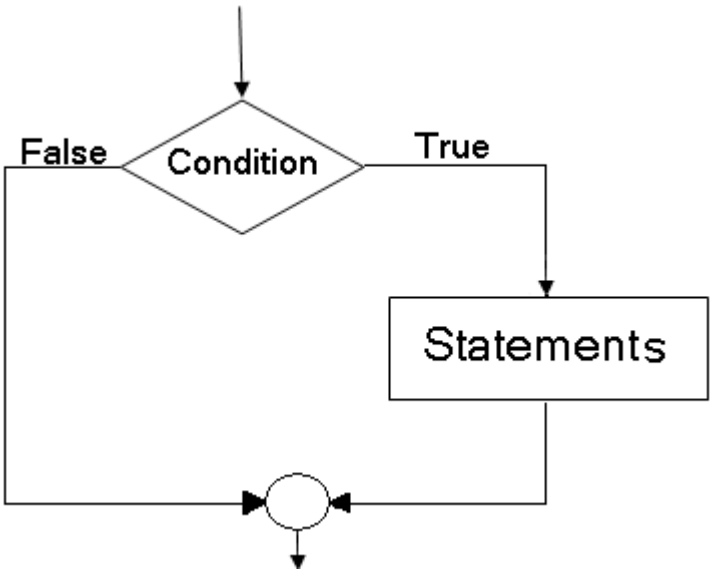
2. الخوارزميات الشرطية : Conditional Algorithms

هي الخوارزميات التي تحتوي واحدة أو أكثر من خطواتها على أوامر شرط (اتخاذ قرار) و تكون على نوعين:-

أ- خوارزميات شرط ذات مسار واحد One way selection وتكتب بالشكل التالي:-

إذا شرط إذن خطوة (أو مجموعة خطوات)

و يمكن تمثيلها بمخطط انسيابي (كجزء من خوارزمية متكاملة) بالشكل التالي:-

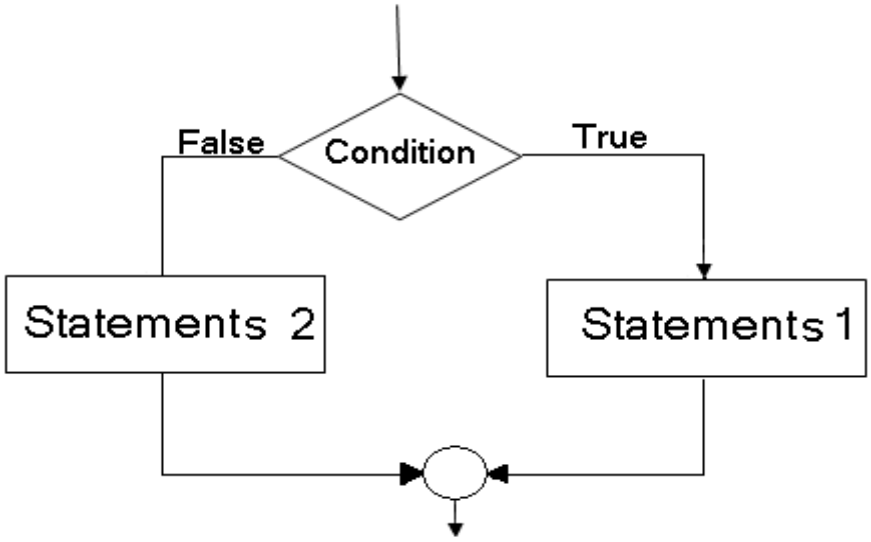


ب- خوارزميات شرط ذات مسارين Two way selection

إذا شرط إذن خطوة 1 (أو مجموعة خطوات)

و إلا خطوة 2 (أو مجموعة خطوات)

و يمكن تمثيلها بمخطط انسيابي (كجزء من خوارزمية متكاملة) بالشكل التالي:-



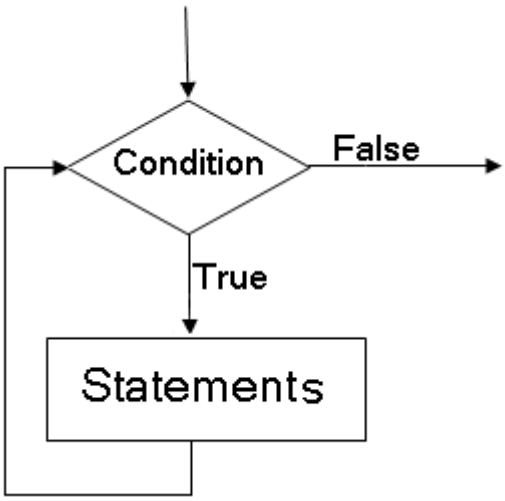
3. خوارزميات التكرار: Iteration Algorithms

يتم في هذا النوع من الخوارزميات تكرار تنفيذ مجموعة من التعليمات (عدد محدود او غير محدود من المرات) حسب النوعين التاليين منها:

أ- خوارزمية Do-While وتكتب بالشكل التالي:-

طالما شرط نفذ خطوات محددة

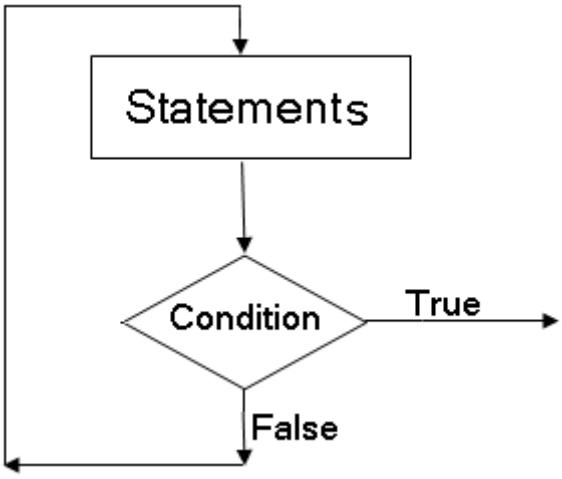
كما يمكن تمثيلها بشكل عام بمخطط انسيابي بالشكل التالي:-



ب- خوارزمية Repeat-Until و تكتب بالشكل التالي:-

كرر خطوات حتى شرط

و تمثل بمخطط انسيابي كما يلي:-



أمثلة :-

اكتب الخوارزمية وارسم المخطط الانسيابي لكل من المسائل التالية:-

1- حساب مساحة المستطيل.

2- إيجاد الوسط الحسابي لعددين.

3- حساب جذور المعادلة التربيعية باستخدام صيغة الدستور.

4- قراءة درجات طالب واحد في الصف السادس العلمي ثم إيجاد و طبع المعدل.

5- تحويل درجة الحرارة من المقياس الفهرنهايتي إلى المئوي علما أن التحويل يتم بطرح 32 من الدرجة ثم ضربها في 9/5 .

6- في كل مهرجان للكتاب يعلن دائما عن تخفيضات للسعر فإذا علمت أن الكتب تخفض بنسبة 25% و تريد شراء كتاب سعره (8000) دينار فكم يجب أن يكون معك كي تشتريه.

7- إيجاد نسبة المتعلمين في المجتمع إذا علمت إن (L) يمثل عدد غير المتعلمين وان (T) يمثل العدد الكلي لإفراد المجتمع.

8- إذا جمعت ثلاثة أعداد هل يكون المجموع أكثر من أو يساوي (500) أم اقل.

9- في المثال رقم (5) قم بتحويل عدد من درجات الحرارة الفهرنهايتية إلى المئوية.

10- إذا كانت لديك مجموعة أعداد تتألف من N عددا مختلفا , اوجد اكبر هذه الأعداد.

11- أربعة عوائل اشترت كل منها على التوالي 6,4,2,5 كيلو من اللحم إذا علمت أن سعر الكيلو الواحد من اللحم هو (12000) دينار احسب المبلغ الذي يجب أن تدفعه كل عائلة إلى صاحب المحل.

12- أطبع الأعداد الزوجية من 6-40.

13- لديك عددان X , Y قارن بينهما أيهما اكبر.

14- لديك 15 عددا كم عدد فيها سالب و كم عدد منها موجب و كم عدد قيمته صفر.

15- اجمع أعداد معلومة ابدأ بالعد من (-2) إلى (2) و خذ الزيادة (0,2).

16- خذ العدد (6) و اضربه على التوالي بالأعداد من 1-10 وأطبع النتائج.

مراحل تطور البرنامج :-

يمكن انجاز مراحل حل المسالة (تطور البرنامج) باستخدام الحاسب بما يلي:-

1. مرحلة التعريف وتحليل المسالة.
2. تصميم الخوارزمية.
3. وضع المخطط الانسيابي للخوارزمية.
4. اختيار لغة البرمجة.
5. كتابة البرنامج.
6. التنفيذ والاختبار.
7. مرحلة التوثيق.

ويمكن شرحها مفصلا" فيما يلي:-

1. مرحلة التعريف وتحليل المسالة:-

التعريف هي عملية كتابة المسالة بصورة واضحة ومبسطة .

أما تحليل المسالة فيعني اختيار طريقة الحل و تستخدم فيها العلاقات و القوانين الرياضية حيث أن العناصر الأساسية في تحليل المسالة هي :-

- ❖ تحديد أهداف المسالة (المخرجات).
- ❖ تحديد عناصر المدخلات (المعطيات) و تشمل المتغيرات و المعلمات كما يحدد نوع عناصر المدخلات و توصيفها بدقة.
- ❖ تحديد العلاقات الرياضية و حصر طرق الحل.

2. تصميم الخوارزمية : Algorithm Design

قبل البدء بكتابة البرنامج يجب التفكير في تصميمه العام , فكما أن بناء ناطحة سحاب جديدة يتطلب خططا للعمل , كذلك تكون كتابة البرنامج.

وان إحدى الخصائص العامة للأشياء المعقدة هي انه عند اكتمالها لأول مرة فإنها نادرا ما تعمل كما خطط لها , حيث توجد دائما عيوب قليلة يجب التخلص منها فإذا كان الشئ قد عرف جيدا من البداية فان تصحيح العيوب غالبا يكون في حد أدنى و يتكون أساسا من تصحيح القليل من الأشياء , لكن إذا كان التصميم الأصلي رديئا فانه يكون مطلوبا كميات جوهرية من الوقت والجهد والأموال في تصحيح عيوب التصميم.

أي انه كلما صرفنا وقتا (مفيدا) في التصميم كلما قل الوقت المحتمل صرفه للتخلص من العيوب.

يجب مراعاة ما يلي عند التصميم :-

أ- قابلية التركيب (Modularity)

التركيبة (Module) هي اي جزء من نظام اكبر واكثر تعقيدا تنجز وظيفة معرفة جيدا. عندما نفكر في وظيفة نظام معقد فاننا نفكر بدلالة التركيبات المكونه له وعليه تسمى هذه التركيبات بالانظمة (Systems) , و لهذا فاننا نفكر في السيارة ليس فقط كخليط كبير من الاجزاء و لكن كتركيبات تتكون من نظام كهربائي و نظام تبريد ونظام تسيير و نظام توقف والخ... و كل من هذه الانظمة هو مثال لتركيبة.

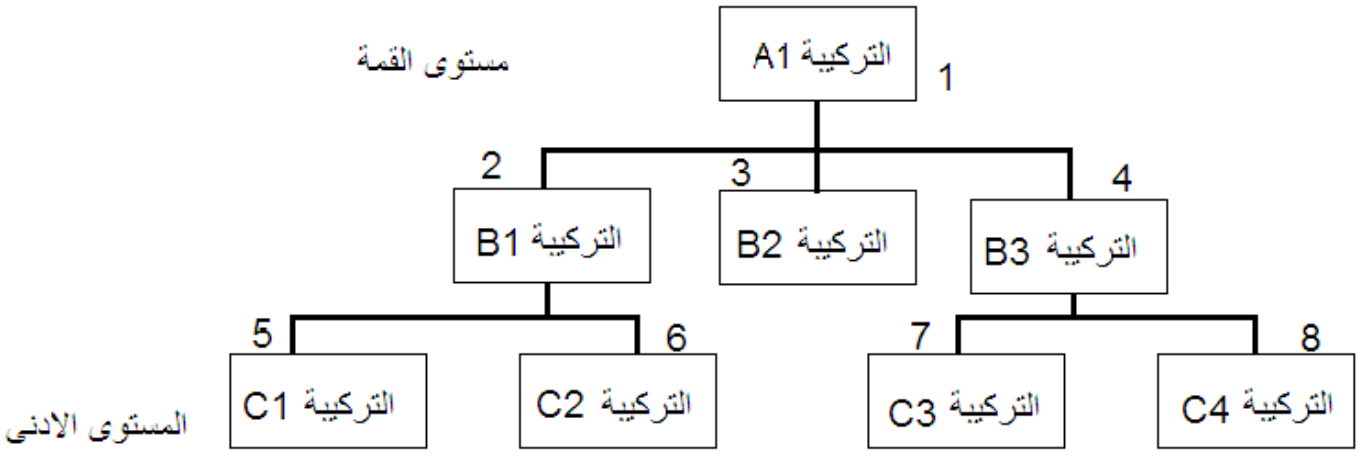
ولهذا عند اصلاح سيارة فاننا نركز انتباهنا على احدى هذه التركيبات مستثنين الاخريات.

ب- التصميم من الاعلى الى الاسفل (Top-Down Design)

يدعى هذا الاسلوب باسماء اخرى مرادفة لهذه التسمية مثل البرمجة النظامية (Systematic Programming) او التجزئة الهرمية او التنقية التدريجية (Step Wise refinement)

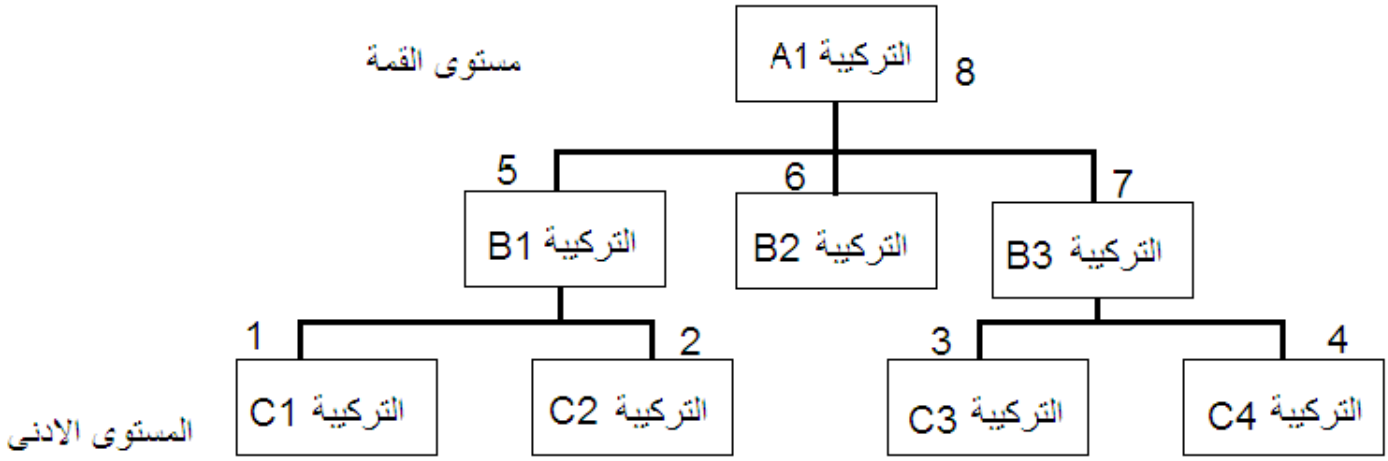
ان هذا الاسلوب من التصميم يتطلب اتخاذ قرارت متكاملة قبل الدخول في اعداد تفاصيل التصميم لمكونات البرنامج , اي تحديد الاعمال الرئيسية المراد انجازها. و من ثم التقدم لتحديد الاعمال الاقل مستوى التي يمكن استنتاجها من الاعمال الرئيسية , وبتكرار هذه العملية (اي عملية تجزئة الاعمال الى اعمال ثانوية اقل مستوى) لعدة مرات الى ان نصل في النهاية الى مستوى ثانوي اخر و هذا المستوى يضم اعمالا بسيطة جدا بحيث يمكن التعبير عنها بواسطة ايعاز او مجموعة ايعازات باستخدام احدى لغات البرمجة . و يستخدم المخطط الهرمي كواسطة مساعدة لتمثيل عملية التجزئة.

ان احد محاسن التصميم من الاعلى الى الاسفل تكمن في ان هذا الاسلوب يركز ومنذ البداية على ماهية الاعمال الواجب القيام بها و من ثم يهتم بطريقة نظامية بالكيفية التي تنجز بها هذه الاعمال.



ت- التصميم من الادنى الى الاعلى (Bottom-Up Design)

كما هو واضح من التسمية فان هذا الاسلوب يستهل بناء هيكل البرنامج مبتدئا باوطا مستوى فيه , ان ذلك يتم بتحديد القرارات الواجب اتخاذها في هذا المستوى و من ثم الارتقاء الى المستويات العليا لبناء الهيكل التام للبرنامج , ان القرارات المتخذة تبدأ بقرارات تخص كتل البناء الاساسية و الاعمال الداخلية للبرنامج و من ثم تتقدم لتشمل القرارات التي تخص المستويات العليا ولذلك تكون القرارات المتخذة في المستويات المتقدمة متاثرة بصورة كبيرة بقرارات المستويات الاوطأ.



ث- الدوال و انهجة المعالجة كتركيبات:

احدى طرق تطبيق التصميم من الاعلى الى الاسفل على الخوارزميات هي جعل الخوارزمية الرئيسية تركيبة مستوى القمة و استعمال الدوال وانهجة المعالجة لمختلف التركيبات التابعة. لهذا فاننا نستهل بكتابة الخوارزمية الرئيسية و لكن اينما ظهرت الحاجة الى انجاز عملية معقدة , فاننا نكتب استدعاء دالة او نهج معالجة لانجاز تلك العملية و بالطبع ينبغي ان ندون في مذكرة ما هو مطلوب عمله بالضبط من هذه الدوال و انهجة المعالجة حيث انه سيكون مطلوبا منا كتابتها فيما بعد.

وانهجة المعالجة و الدوال التي تستدعيها الخوارزمية الرئيسية تكون تركيبات المستوى الثاني و عند كتابة هذه فأننا مرة أخرى نستدعي انهجة معالجة و دوالا لانجاز عمليات معقدة , هذه الانهجة والدوال تكون تركيبات المستوى الثالث و في النهاية نصل الى المستوى النهائي (Bottom).

3. **وضع المخطط الانسيابي للخوارزمية** :- Flow Chart

اي رسم الخوارزمية باستخدام الاشكال المناسبة في مخطط انسيابي للتمكن من تتبع سير العمليات في الخوارزمية و سهولة اكتشاف الاخطاء.

4. **اختيار لغة البرمجة** : Programming Language

و هي خطوة مهمة جدا , حيث ان اختيار لغة البرمجة المناسبة يساعد في الحصول على النتائج المرغوبة بدقة, اذ تختلف امكانيات و اغراض لغات البرمجة , وان قرار اختيار لغة البرمجة يتخذ من قبل محلل الانظمة.

5. **كتابة البرنامج** : Program Writing

تلي عملية اختيار لغة البرمجة كتابة البرنامج على وسط ادخال مناسب و غالبا ما تكون مهمة كتابة البرنامج من مهام المبرمج . ويعتمد المبرمج في كتابة البرنامج على المخطط الانسيابي الذي يعده محلل الانظمة او المبرمج نفسه, و يسمى البرنامج بشكله الاخير بالبرنامج المصدري (Source Program) وبعد ان يتم ادخال البرنامج المصدري الى الحاسبة ياتي دور الحاسبة في معالجة البرنامج المصدري كما يلي:-

- ✓ ادخال البرنامج المصدري.
- ✓ ترجمة البرنامج المصدري الى شفرة لغة الالة و هو ما يعرف بالبرنامج الهدف(Object Program) .
- ✓ اكتشاف الاخطاء.
- ✓ اعادة ترجمة البرنامج المصدري بعد تصحيح الاخطاء و تحويله الى البرنامج الهدف (لغة الالة).
- ✓ فحص البرنامج و نقله الى ذاكرة الحاسب بواسطة البرنامج المحمل (Loader).

أنواع الأخطاءالبرمجية:

هناك ثلاثة أنواع من الأخطاء البرمجية ألا وهي:

1- أخطاء هجائية : Syntax Errors

وهو خطأ في سوء كتابة جملة معينة في شفرة البرنامج ، على سبيل المثال نجد هناك بعض كلمات المحجوزة التي يجب كتابتها كما هي دون تغيير , يمكن أن يخطأ المبرمج في كتابتها، فيكتشفها البرنامج على الفور ، وهذه الأخطاء تظهر أثناء كتابة شفرة البرنامج ، أو بمعني أدق كتابة الكلمة أو الجملة في صورة غير صحيحة.

2- أخطاء أثناء التنفيذ : Runtime Errors

هذا الخطأ لا يظهر إلا بعد ترجمة البرنامج وتنفيذه ، لأن لغة البرمجة لا تلاحظها إلا بعد عملية التنفيذ ، علي سبيل المثال إذا قمت بقسمة أي عدد على صفر :

Num = 50 / 0

هذه العملية تقوم بقسمة 50 علي 0 ، ونحن نعلم جميعاً أن قسمة أي رقم علي صفر تعطي رقم غير نهائي ، وهذا لا يجوز .

3- أخطاء منطقية : Semantic Errors

هذا النوع يعتبر أخطر نوع من الأخطاء ، حيث أن لغة البرمجة لا ولن تكتشفها بعد ، لأن هذا الخطأ خطأ في المعالجة الحسابية

مثل كتابة معادلة خاصة بمساحة المربع علي أنها مساحة المثلث

$$N = 0.5 * a * b$$

وصحتها كالتالي : $N = a \wedge 2$

6. التنفيذ والاختبار : Running & Testing

ان اي برنامج بالضبط مثل **اي** نظام معقد اخر , يجب اختبار كل جزء فيه اختبارا شاملا بعد اكتماله و قبل ان يسلم الى مستعمليه. و هذه الخطوة حيوية جدا وان احسن طريقة لاختبار اي نظام سواء كان برنامجا ام جهازا ميكانيكيا معقدا هي اختباره جزء بعد جزء اي اننا نختبر كل تركيبة بمفردها و كذلك مختلف التركيبات متحدة.

وان الطريقة الاكثر وضوحا لعمل هذا تسمى بالاختبار من الاسفل الى الاعلى اي ان تختبر التركيبات الاساسية في المستويات الدنيا اي تلك التي لا تستدعي اية تركيبات اخرى , وعندما نتاكد من صلاحيتها للعمل ننتقل الى المستوى الاعلى و نختبر تلك التركيبات التي بنيت من التركيبات التي سبق اختبارها. و نستمر بهذه الطريقة حتى نصل في النهاية الى مستوى القمة و يمكننا اختبار البرنامج كاملا.

و يمكن الاختبار ايضا من الاعلى الى الاسفل و هنا نبدا بتركيبة مستوى القمة و نختبرها اولا ثم نختبر تركيبات المستوى الثاني وهكذا حتى يتم اختبار التركيبات الاساسية في اخر مستوى.

و لاختبار اي تركيبة او برنامج كامل يجب تزويده ببيانات اختبار تكون مخرجاتها الصحيحة معروفة وان نقارن المخرجات التي ينتجها البرنامج مع المخرجات الصحيحة و اذا ما ثبتت صحة الحل يمكن تنفيذ البرنامج على البيانات الحقيقية.

7. مرحلة التوثيق و الصيانة : Documentation & Maintenance

ان التوثيق هو احد اهم شروط بناء البرنامج فالتوثيق يكون ضمن البرنامج المكتوب و ذلك بوضع شرح مبسط لعمل كل برنامج فرعي فيه.

والهدف الرئيسي من التوثيق هو تسهيل استيعاب فهم و استخدام البرنامج من قبل كافة المتعاملين معه يتم في هذه المرحلة اعداد دليل استخدام البرنامج تشرح فيه خطوات عمل البرنامج و بيان اساليب ادخال المعلومات و نماذج البيانات و يحقق التوثيق الفعال اغراضا كثيرة من بينها:-

1) توضيح نطاق العمل و توصيف الاساليب المستخدمة و بيان التغيرات التي ادخلت على العمليات السابقة.

2) تزويد القائمين على اختبار البرنامج و تشغيله بالمعلومات الاساسية التي يتم الاعتماد عليها في اعداد البرنامج.

3) تسهيل تعديل وصيانة وتطوير البرنامج.

مزايا البرنامج الجيد:-

ان اهم الخصائص النوعية الجيدة التي يجب توافرها في البرنامج هي كالاتي:-

1. **البرنامج يؤدي العمل**: يجب ان يؤدي البرنامج العمل المطلوب منه وفق المواصفات المعرفة لوظيفته و مدخلاته و مخرجاته و العمليات التي يؤديها.
2. **البرنامج قابل للقراءة والفهم** : يجب ان يكون البرنامج المصمم واضح القراءة و بصياغة تسهل فهم خطواته و وظائفه من قبل الاخرين. ويجب اعتماد خصائص البرمجة المهيكلة التي تؤدي الى استخدام افضل لوقت المبرمج و كتابة برنامج يحقق الاستخدام الامثل لموارد منظومة الحاسب خصوصا ما يتعلق بالمساحة الخزنية و وقت التنفيذ.
3. **البرنامج قابل للتطوير**: يجب تصميم البرنامج بطريقة يمكن تطويره لاحقا او تعديله بسهولة اذا تطلب الامر ذلك بجهود مقبولة و وقت مناسب, اذ ان الضرورة قد تقتضي تعديل البرنامج في مرحلة التصميم و الاختبار.
4. **انجاز البرنامج**: يجب ان يعد البرنامج بصورته المتكاملة و بمراحله المختلفة ضمن الفترة الزمنية و التخصيص المالي المحددين له واي تجاوز على ذلك سيفقد جدواه الاقتصادية.

العمليات

Processes

مقدمة

في بدايات الحاسب كان النظام يسمح لبرنامج واحد أن ينفذ في وقت معين .وكان هذا البرنامج يسيطر سيطرة تامة على النظام.

ولكن في الحاسبات الحالية يسمح النظام لأكثر من برنامج أن يحمّل إلى الذاكرة وأن ينفذوا في نفس الوقت .وهذا التطور يتطلب تحكم أكبر وتقسيم البرامج المختلفة إلى أجزاء مستقلة، وهذه الاحتياجات أنتجت لنا ما يدعى بالعملية process وهي البرنامج في مرحلة التنفيذ. والعملية هي وحدة العمل في ، أنظمة مشاركة الوقت time-sharing الحديثة.

وكلما كان نظام التشغيل معقدًا، كلما توقعنا منه عمل أمورًا أكثر بالنيابة عن مستخدميه .لذا يتكون النظام من مجموعة من العمليات :عمليات نظام التشغيل تنفذ شفرة (code) النظام،وعمليات المستخدم تنفذ شفرة المستخدم. ومن الممكن أن تعمل كل هذه العمليات في نفس الوقت، بجعل وحدة المعالجة المركزية (CPU) تعمل عليهم بتعدد multiplexed.

بتبديل وحدة المعالجة بين العمليات، يمكن أن يجعل نظام التشغيل الحاسب أكثر إنتاجيه.

أولاَ :مفهوم العملية Process Concept

العملية هي تنفيذ برنامج متسلسل، وعموما يحتاج تنفيذ العملية إلى عدة موارد منها :وقت وحدة المعالجة المركزية (CPU)، الذاكرة، الملفات وأجهزة الإدخال والإخراج .وهذه الموارد تعطى للمهمة إما وقت, إنشاءها أو وقت تنفيذها.

والعملية عادة هي وحدة عمل متكاملة في أغلب أنظمة التشغيل operating systems)) وهي إما عمليات خاصة بأنظمة التشغيل وتنفذ برامج التشغيل، أو عمليات خاصة بالمستخدمين وتنفذ برامج المستخدمين . وجميع هذه العمليات تنفذ في نفس الوقت.

و المسؤول عن إدارة هذه العمليات وكل ما يتعلق بها من إنشاء أو إلغاء وجدولة وآلية تزامن واتصالات العمليات هو نظام التشغيل.

العملية Process

تحتوي العمليات (على الأقل) على:

- مساحه العنونة (address space) هي مساحه محجوزة بالذاكرة (تحتوي على معلومات العملية)
- الكود المستخدم في البرنامج المراد تنفيذه (program code).
- البيانات المخزنة للبرنامج المراد تنفيذه (program data).
- ومؤشر للتكديس (stack pointer).
- عداد للبرنامج (program counter) (حيث أن البرنامج يتألف من عدة سطور وهذا العداد يعد هذه الأسطر).
- السجل (register) وقيمه.
- الكومة (heap) عمليه تطويق أو تحديد البيانات التي استخدمناها في هذه العملية .

ثانيًا :حالات العمليات Processes State :

كل عمليه من العمليات لابد أن تمر بأكثر من حاله وقت تنفيذها ,هذه الحالات تدل على نشاطها في هذه اللحظة.

الحالات التي تمر بها أي عمليه هي:

التجديد (new) (أو حالة لم تستخدم من قبل أو بالأصح لم يفعل استخدامها) :

وهي وقت تعريف العملية ووقت السماح لها بالدخول إلى قائمه العمليات الموجودة في الذاكرة الرئيسية RAM ويتم ذلك بالضغط على البرنامج ضغطة مزدوجة وبالتالي تنتقل هذه الحالة من

الحالة الخاملة إلى حاله أخرى، مثل":حاله التنشيط."

الاستعداد (ready):

هي العملية الجاهزة للتنفيذ والدخول إلى وحدة المعالجة المركزية CPU، ولن يسمح لها بالتنفيذ بسبب وجود عمليه أخرى تنفذ في نفس الوقت.

التشغيل أو التنفيذ (Running):

هي حالة العمليات والأوامر وقت التنفيذ في وحدة المعالجة المركزية CPU

الانتظار (waiting):

هي حالة العملية عند انتظار حدوث أمر معين، مثلا :ينظر إدخال بيانات من المستخدم أو عمليه طباعة.

الانتهاء (terminated):

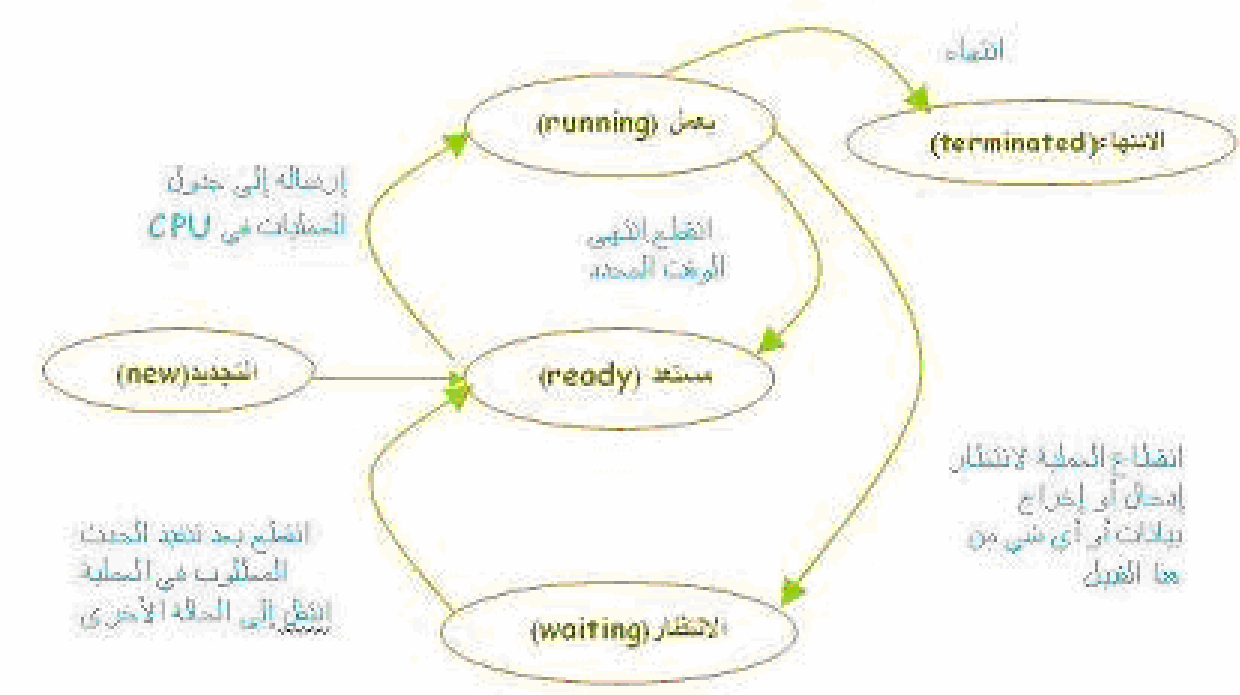
هي حالة العملية عند الانتهاء ,وهي إما أن تكون العملية قد انتهت بشكل سليم أو أنه قد حصل لها خطأ معين أدى إلى إنهاءها.

ومن مفهومنا للحالات يمكن أن نستنتج كم عملية يمكن أن تتم في كل حالة.

- في حالة (الاستعداد): يمكن أن توجد أكثر من عملية في حالة استعداد في نفس الوقت، وكل هذه العمليات مستعدة للتنفيذ في أي وقت.

-في حالة (التشغيل) :فإنه في وقت معين يتم تشغيل عملية واحدة على وحدة المعالجة المركزية (CPU)، ولا يمكن أن يكون هناك أكثر من عملية تعمل على وحدة المعالجة المركزية (CPU) في نفس الوقت (أي في حالة التشغيل)، فقط عملية واحدة تنفذ في وقت واحد ولا يمكن أن ينفذ أكثر من ذلك.

-في حالة (الانتظار) :تشبه حالة (الاستعداد) من الممكن أن تكون هناك أكثر من عملية في حالة انتظار لحدث معين أيًا كان هذا الحدث في نفس الوقت.



حالات العمليات

شرح مبسط للرسم الموضح أعلاه:

تبدأ العملية من حاله التجديد وذلك بالضغط عليها من جهاز الحاسب لديك ثم تنتقل بعد ذلك إلى حالة (الاستعداد) وهذه الحالة يتم إضافتها إلى الجدولة في (CPU) ليتم تنفيذها عندما يحين الوقت المخصص لها

تبدأ العملية بالتنفيذ وتنتقل من حالة إلى حالة في حالات معينة:

- تنتقل إلى حالة الانتهاء (terminated)عندما تنتهي العملية بسلام بشكل كامل أو عند حدوث خطأ معين أدى إلى أن يقرر النظام إنهاء العملية

- تنتقل إلى حالة الاستعداد (ready) عندما ينتهي الوقت المحدد لهذه العملية ولا تحتاج إلى تنفيذ حدث معين سواء إدخال بيانات أو غيره

- تنتقل إلى حالة الانتظار (waiting)عندما تكون العملية تمت بشكل جزئي ولكن تحتاج إلى حدث معين يطلب من المستخدم سواء إدخال أو طباعة أو أوامر أخرى

- عندما تكون العملية في حالة الانتظار وانتهى الحدث المطلوب تنتقل من حالتها إلى حاله الاستعداد، إذا انتهى الحدث بشكل كامل فهي الآن مستعدة للتنفيذ.

الروتينات الفرعية (البرامج الفرعية) : Subroutines

كثيرا ما نحتاج الى تكرار تنفيذ جملة معينة من البرنامج او مجموعة من الجمل المتعاقبة مرة واحدة او اكثر حسب الحاجة. وان تكرار مثل هذه الجمل من شانه ان يجعل البرنامج طويلاً و بطيئاً و قد وجد ان الحل الامثل في مثل هذه الحالة هو استعمال ما يسمى بالبرامج الفرعية.

البرنامج الرئيسي هو البرنامج الذي يحتوي على برامج فرعية (Main Program) اما البرامج الفرعية (Subprograms-Subroutines) فهي برامج مستقلة تحتوي على مجموعة من الجمل المتتالية و يستخدم لانجاز عمليات معينة يمكن ان تتكرر اكثر من مرة ضمن البرنامج الرئيسي .ويكتب البرنامج الفرعي مرة واحدة فقط و لكن يمكن استدعاؤه عدة مرات من خلال البرنامج الرئيسي.

ان الفائدة الرئيسية التي يتم الحصول عليها من جراء اتباع هذا الاسلوب في البرمجة تكمن في النقاط الاتية:-

1. ان هذا الاسلوب يوفر امكانية استدعاء البرنامج الفرعي الواحد اكثر من مرة واحدة في البرنامج الرئيسي او في برنامج فرعي اخر .
2. كتابة البرامج مرة واحدة فقط يوفر الجهد و الوقت للمبرمج .
3. امكانية تصميم البرنامج من قبل اكثر من مبرمج واحد .
4. يمكن كتابة البرنامج الرئيسي قبل كتابة البرامج الفرعية التابعة له او بالعكس.
5. سهولة تصحيح الاخطاء.
6. تقليل حجم البرنامج وبذلك توفير السعة المطلوبة ضمن وحدة الذاكرة الرئيسية.
7. سهولة صيانة البرنامج واجراء التغييرات و التعديلات عليه.
8. سهولة قراءة البرنامج وتتبعه.

تقسم البرامج الفرعية الى نوعين:

1. الدوال Functions
2. أنهجة المعالجة (الاجراءات) Procedures

و الفرق بين الدالة ونهج المعالجة يكمن في ان الدالة تعيد قيمة واحدة فقط الى البرنامج المستدعي بينما يمكن لنهج المعالجة اعادة اكثر من قيمة الى البرنامج المستدعي.

أولا" - الدوال : Functions

تقسم الدوال الى نوعين:

أ- دوال مكتبية جاهزة (مبنية داخليا" Built- in Functions) يمكن للمبرمج استخدامها بذكر اسمها والمعلمات الخاصة بها و من أمثلتها:-

Sin, Cos, Tan, Sqrt, Abs, Log, Exp. …

ب- دوال يعرفها المبرمج و أيضا تكون على نوعين:-

1. دوال السطر الواحد Single Line Functions :-

يستعمل هذا النوع من الدوال عندما يجد المبرمج أثناء كتابته للبرنامج انه بحاجة إلى تكرار تعبير حسابي في أكثر من جملة حسابية في البرنامج فيمكنه اختصار البرنامج بتعريف دالة بجملة واحدة تسمى جملة الدالة بحيث يتم استعمالها في جمل حسابية و لأكثر من مرة في مواضع مختلفة من البرنامج بمجرد ذكر اسم الدالة.

DEF FNname(a_1, a_2, …, a_n) = تعبير حسابي

مثال :-

DEF FN x(y, c) = y^2 + 3 + c ← تعريف الدالة

Let G = FN x(a, b) ← استدعائها في البرنامج الرئيسي

2. دوال السطور المتعددة Multi- Line Functions:

في بعض لغات البرمجة نحتاج إلى تعريف دالة ونجد بان تعريفها يتطلب أكثر من عبارة واحدة و قد يستعمل مع التعبير الحسابي عبارات شرطية أو تكرارية ... الخ.

مثلا يمكن تعريف الدالة كما يلي:-

Function Function name(par_1, par_2,…,par_n)

مجموعة تعليمات تمثل جسم الدالة

End (Function name)

الشرط الأساسي في كتابة الدالة هو أن يحتوي جسم الدالة على تعليمة إعطاء قيمة واحدة على الأقل لاسم الدالة و بالشكل التالي:-

Function name = تعبير حسابي

يمكن أن نتصور بان اسم الدالة هو أيضا اسم لموقع خاص يستعمل لحفظ قيمة الدالة و عندما تنفذ التعليمات الموجودة في جسم الدالة , فان القيمة المخزونة في موقع الذاكرة الخاص بها سيعاد كقيمة للدالة.

لنأخذ المثال التالي:- عرف دالة لتحسب قطر شكل مستطيل اذا علمت طوله و عرضه ؟
و نجد بان التعبير المستخدم في حساب القطر سيكون باستخدام نظرية فيثاغورس و كما يلي:-

SQRT(length^2 + width ^2)

الحل :-

تعريف الدالة كما يلي:-

```
Function     Diagonal(length, width )
  Diagonal =  SQRT(length^2 + width ^2)
End  Diagonal
```

حيث نفترض أن تنشيط الدالة (استدعائها في البرنامج الرئيسي) يكون بالشكل التالي:-

```
X = 4
Y = 3
Z = Diagonal(X, Y)
```

وعند تنشيط الدالة Diagonal فان جسم الدالة سينفذ مع استبدال المعلمتين الشكليتين (length, width) بقيمتي المعلمتين الحقيقيتين (X, Y) .
كما نلاحظ انه يمكن لتعريف الدالة أن ينشط دوالا" أخرى مثل الدالة SQRT .

مثال :- عرف دالة لتميز العدد الأكبر بين عددين معلومين؟

```
Function     Max(K, H )
      If   K > H   then
                  Max = K
            Else
             Max = H
            Endif
   End  Max
```

حيث أن هذه الدالة يمكن تنشيطها بتعاقب التعليمات التالية:-

```
P = 8
Q = 9
R = Max(P, Q)
```

و يمكن أن يكون احد أو كلا المتغيرين P, Q تعبيرا حسابيا.

ثانيا" – أنهجه المعالجة (الإجراءات) : Procedures

يمكن للدالة أن تعيد قيمة واحدة فقط لذلك عندما تنتج الحسابات التي نريد عملها أكثر من قيمة (نتيجة) فانه يمكن كتابتها كنهج معالجة , يمكنه إعادة أي عدد من النتائج . و الأكثر من هذا فان نهج المعالجة يمكنه معالجة بعض متغيرا ت الخوارزمية المنشطة بطرق اختيارية و لهذا يمكن مثلا للمتغيرات نفسها التي توفر قيمها بيانات الحسابات أن تستعمل أيضا لخزن النتائج.

والشكل العام لتعريف نهج المعالجة يكون بالشكل التالي:-

```
Procedure     Procedure name(par1, par2,…,parn)
        -------
        -------                مجموعة تعليمات تمثل جسم النهج
        --------
   End (Procedure name)
```

نلاحظ أن نهج المعالجة لا يحتوي على تعليمة إعطاء قيمة لاسم نهج المعالجة حيث أن المعلمات الشكلية , بدلا من اسم النهج تستعمل لإعادة القيم إلى البرنامج المستدعي.

الاختلاف البارز بين الدوال و انهجة المعالجة هو في الطريقة التي تستعمل بها معلماتها الشكلية , فبالنسبة إلى الدوال , فان قيم المعلمات الحقيقية تستبدل بالمعلمات الشكلية و لهذا السبب فإننا نشير إلى المعلمات الشكلية للدوال كمعلمات قيمة (Value Parameter) . أما بالنسبة إلى انهجة المعالجة يجب أن تكون المعلمات الحقيقية متغيرات , وهذه المتغيرات (وليس قيمها) تستبدل المعلمات الشكلية و لهذا السبب فإننا نشير إلى المعلمات الشكلية لانهجة المعالجة كمعلمات متغير (Variable Parameter) .

كما أن نهج المعالجة ينشط بتعليمة استدعاء لها الهيئة التالية:-

Procedure name(var_1, var_2, … , Var_n)

وان المعلمات (var_1, var_2, … , Var_n) هي معلمات حقيقية , عندما ينشط نهج المعالجة (يستدعى من قبل البرنامج الرئيسي) فان التعليمات التي في جسم نهج المعالجة يتم تنفيذها مع استبدال المعلمات الشكلية بالمتغيرات (var_1, var_2, … , Var_n).

<u>مثال :-</u>

اكتب نهج المعالجة لحساب مساحة وقطر مستطيل بمعرفة طوله و عرضه؟

```
Procedure        Rectangle(length,width,Area,Diagonal)
    Area = Width * length
      SQRT(length^2 + width ^2) Diagonal =
End  Rectangle
```

المعلمتان الشكليتان Width , Length توفران البيانات للحسابات بينما تستعمل المعلمتان الشكليتان Area , Diagonal لإعادة النتائج إلى البرنامج المستدعي.

نفترض أننا نشطنا نهج المعالجة بالتعليمات التالية:-

```
W = 12
X= 5
Rectangle(W, X, Y, Z)
```

عند تنشيط نهج المعالجة يتم تنفيذ التعليمات في جسمه بعد استبدال المعلمات الشكلية بالمعلمات الحقيقية و لهذا فان تنشيط نهج المعالجة Rectangle له نفس تأثير تنفيذ التعليمات التالية:-

$$Y = W * X$$

$$Z = SQRT(W^2 + X^2)$$

لذلك فبعد تنفيذ نهج المعالجة والعودة منه تبقى قيم W, X كما هي بينما تتغير قيم Y, Z.
كما يمكن لانهجة المعالجة أن تأخذ متغيرات منظومات(مصفوفات) كمعلمات حقيقية أو شكلية.

المتغيرات المحلية:- Local Variables

هي المتغيرات المستعملة في تعريف دالة أو نهج معالجة وان ظهورها في خوارزميات أو دوال أو انهجة معالجة مختلفة يكون غير مرتبطا احدها بالآخر حتى لو حدث و كان لها نفس الاسم . و يتم البدء بها غير معرفة لكل تنشيط.

المتغرات العالمية:- Global Variables

هي المتغيرات المشتركة بين خوارزمية وبعض الدوال و انهجة المعالجة التي تنشطها و ستحتفظ هذه المتغيرات بقيمها بين التنشيطات للدوال و انهجة المعالجة.

الدوال و انهجة المعالجة المتداخلة :- Recursive Function & Procedures

إن الدوال و انهجة المعالجة يمكن تعريفها بدلالة دوال و انهجة معالجة أخرى , كما يمكن تعريف دالة أو نهج معالجة جزئيا بدلالة نفسها . مثل هذه الدوال و انهجة المعالجة يقال بأنها متداخلة (Recursive) أو ذات تكرار ذاتي (تستدعي نفسها).

مثال ذلك الدالة Factorial (المضروب) حيث أن (N!) هو حاصل ضرب كل الأعداد الصحيحة من 1 إلى N مثلا":-

$$Factorial(1) = 1$$

$$Factorial(2) = 2 * 1$$

$$Factorial(3) = 3 * 2 * 1$$

وهكذا ...

وحيث أن Factorial(1) = 1 نلاحظ بأن :

$$Factorial(2) = 2 * Factorial(1)$$

و بنفس الطريقة نجد أن Factorial(3) = 3 * Factorial(2)

و هكذا ...

أي انه يمكننا تعريف الدالة Factorial بالتعليمتين التاليتين :-

Factorial(1) = 1

Factorial(N) = 2 * Factorial(N-1)

حيث N اكبر من 1 و يمكن كتابة الدالة كما يلي:-

```
Function    Factorial(N )
        If   N = 1   then
                    Factorial = 1
              Else
                Factorial = N * Factorial(N-1)
        Endif
End  Factorial
```

افرض أن هذه الدالة قد نشطت لحساب Factorial(5) أثناء حساب الدالة Factorial(5) فان الدالة تنشط مرة أخرى لحساب Factorial(4) وأثناء حساب Factorial(4) فان الدالة تنشط مرة أخرى لحساب Factorial(3) و هكذا ... و في النهاية تنشط الدالة لحساب Factorial(1) و يمكن حساب Factorial(1) بدون تنشيط الدالة ... لماذا ؟

و بتوفر قيمة !1 فانه يمكن حساب !2 و بتوفر قيمة !2 يمكن إكمال حساب !3 و هكذا إلى أن نحصل أخيرا" على قيمة !5 .

من ناحية الذاكرة والمعلمات والمتغيرات فانه لكل تنشيط فانه لكل تنشيط للدالة تخصص منطقة ذاكرة منفصلة لخزن قيم المعلمات و المتغيرات المحلية.

مثال :-

$$power(a, n) = \begin{cases} a & \text{if } n = 1 \\ a * power(a, n-1) & \text{if } n>1 \end{cases}$$

أسلوب البرمجة المهيكلة

إن اللغات العليا هي لغات موجهة لحل مشاكل معينة بينما اللغة الدنيا هي لغة موجهة نحو الآلة و بعبارة أخرى إن اللغة العليا هي وسائل مناسبة و بسيطة لوصف هياكل البيانات و تعاقب الأفعال المطلوبة لانجاز مهمة معينة. ولكل برنامج مكتوب بلغة عليا هيكل خاص .

هيكل البرنامج :

يعرف هيكل البرنامج في مستويات عديدة البناء الكامل للبرنامج , هيكل التعليمات الفردية , و هيئة أجزاء التعليمات , مثلا التعبيرات الحسابية. يحدد هيكل البرنامج بمجموعة من القواعد تسمى Rule of Syntax يعبر عن هذه القواعد بطرق مختلفة فالبعض يستعمل مخططات بيانية أو تدوين (Bacus – Naur Form BNF) . بذلت جهود كبيرة في تطوير و استعمال اللغات العليا إلى الطريقة التي يقسم فيها البرنامج إلى كتل Blocks كل كتلة تنجز مهمة خاصة. وتسمى هذه الكتل Subroutines أو Functions أو Procedure . سنستخدم لغة خوارزمية غير رسمية لوصف الخوارزميات المكتوبة بلغات عليا مهيكلة (Informal Algorithmic Language) . كل لغات البرمجة – العليا و الدنيا - لها طرق لنقل الضبط من جزء برنامج إلى جزء آخر . يوجد في اللغات العليا عموما ثلاث أنواع من هياكل الضبط و هي :-

1. تراكيب السلسلة أو التعاقب Sequence
2. تركيب الاختيار Selection
3. تركيب التكرار Repetition

تراكيب السلسلة(التعاقب) Sequence

هو انسياب الضبط من تعليمة إلى أخرى . في معظم لغات البرمجة يتم هذا بكتابة التعليمات واحدة بعد الأخرى في سطور متعاقبة أو مفصولة بفاصلة منقوطة (;) .

والشكل العام لهذا التركيب كما يلي :

```
Algorithm algorithm-name
Statement 1
Statement 2
.
.
.
Statement n
End    algorithm-name
```

مثال :-

خوارزمية تحسب أجور شخص عندما تعطى الساعات التي اشتغلها و المبلغ المدفوع عن كل ساعة عمل.

```
Algorithm Wages
        Input    Name , Hours ,    Rate
        Gross_Wages  =  Hours * Rate
        Output 'Name: ' , Name
        Output 'Payment: ' , Gross_Wages
End Wages
```

مثال :-

خوارزمية لحساب مساحة الدائرة إذا عرف قطرها؟

```
Algorithm Circle _ Area
        Input  Radiance
        Pi = 3.1415
        Area = (Radiance / 2)^2 * Pi
        Output ' Circle Area = '  ,  Area
End  Circle _ Area
```

تركيب الاختيار Selection

يهتم بنقل الضبط إلى جزء من البرنامج عندما يتحقق شرط و إلى جزء آخر عندما لا يتحقق الشرط.

وهو عموما بالشكل التالي:

If Condition (شرط) then

Statements 1

Else

Statements 2

End If

إذا تحقق الشرط فان مجموعة التعليمات الأولى Statements 1 ستنفذ وخلافا لذلك تنفذ مجموعة التعليمات الثانية Statements 2 , في كلا الحالتين يستمر التنفيذ أخيرا من التعليمة التي تلي End If.

مثال :-

خوارزمية تقبل اسمين و تطبعهما حسب الترتيب الأبجدي؟

```
Algorithm Alphabetized
    Input  First_name ,  Second_name
    If   First_name < = Second_name  Then
        Output    First_name , Second_name
    Else
        Output    Second_name  , First_name
    End If
End  Alphabetized
```

مثال :-

خوارزمية لإيجاد قيمة اقتران القيمة المطلقة لأي عدد مدخل؟

```
Algorithm   Absolute_value
        Input  Number
        If   Number < 0   Then
                Abs_Num =  - Number
        Else
                Abs_Num =   Number
        End If
        Output  ' Absolute Value = '  Abs_Num
End  Absolute_value
```

تركيب التكرار Repetition

توجد الحلقات (الدوارات Loop) في كل لغات البرمجة العليا لتكرار تنفيذ مجموعة من التعليمات في حال تحقق شرط معين . و من أهم هذه الحلقات تكرار طالما يتحقق شرط While – do.

و تأخذ حلقة While – do الشكل التالي:

```
While   Condition   Do
        Statements
End   While
```

يبدأ الحاسب باختبار الشرط فإذا كانت له القيمة True تنفذ مجموعة التعليمات المحصورة داخل الحلقة وخلافا لذلك يستمر التنفيذ من التعليمة التي تلي عبارة End While بعد تنفيذ التعليمات يعود الحاسب و يختبر الشرط مرة أخرى وهذا التعاقب من الأحداث يكرر نفسه حتى يصبح للشرط القيمة False في النهاية. وعندما يحدث ذلك ينتهي التكرار و يستمر التنفيذ من العبارة التي تلي عبارة End While.

نلاحظ أن الشرط يتم اختباره قبل تنفيذ أي من التعليمات و لهذا السبب يمكن أن لا تنفذ تعليماتها نهائيا" , إذا كان الشرط غير متحقق في البداية عندما يصل الحاسب إلى تعليمة While – Do وهذه خاصية مميزة لهيكل الضبط While – Do.

مثال :-

خوارزمية لحساب معدل عدد غير معروف من الأعداد الموجبة ؟

```
Algorithm Average
        Sum = 0
        Count = 0
        Input  Number
        While   Number > = 0    Do
                Sum  =  Sum + Number
                Count = Count  +  1
                 Input  Number
        End While
        If   Count  >  0   Then
                Output  Sum / Count
        Else
                Output  ' No Numbers Were Entered'
        End If
End Average
```

مثال :-

خوارزمية لطبع جدول ضرب العدد 6 ؟

```
Algorithm   Multiplication
        X = 1
        While   X < = 10   Do
                Output   X  ' * 6 = '  X * 6
                X = X + 1
        End While
End Multiplication
```