الفصل الثاني

حل المشكلة Problem Solving

القدرة على حل المشاكل بواسطة البرمجة هي مهارة وطريقة مرتبة ولا تعتمد على العشوائية، وهذه القدرة يمكن اكتسابها وتعلمها باتباع بعض القواعد التي تساعد على ذلك.

بالانتهاء من هذا الفصل ستكتسب المعارف وتتدرب على المهارات التي تجعلك قادرا على:

- خطوات حل المشكلة
- خوارزمية الحل
- خرائط المسار (التدفق)

مقدمة

القدرة على حل المشاكل بواسطة البرمجة هي مهارة وطريقة مرتبة ولا تعتمد على العشوائية، وهذه القدرة يمكن اكتسابها وتعلمها باتباع بعض القواعد التي تساعد على ذلك وبعض هذه القواعد ذكرها رين ديكارت الرياضي والفيلسوف المعروف وهي:

١. لا يمكن قبول أي شيء حقيقة مسلمة إلا إذا ثبت ذلك بالتجربة والمشاهدة.
٢. كل مشكلة أو معضلة يتم تبسيطها وتقسيمها إلى أجزاء عدة كلما أمكن ذلك.
٣. فكر بطريقة منظمة ومنطقية وذلك بالبدء بالأجزاء البسيطة والسهلة الفهم ثم التدرج إلى الأجزاء الأصعب وهكذا حتي يتم الانتهاء من المشكلة.
٤. المراجعة لجميع الأجزاء حتي يكتمل الحل.

وبالرغم من أن هذه القواعد تم وضعها قبل ٣٠٠ عام من صناعة أول حاسب إلكتروني إلا أنها ما زالت مطبقة وصالحة للاستخدام، والتفكير الجيد والمنظم لتعريف وتحديد المشكلة ضروري ومهم جداً وأساسي للحصول على نتائج صحيحة وبخاصة عند التعامل مع الحاسب، ولذلك فإن أول خطوة لحل المشكلة هو فهمها.

فهم المشكلة / صياغة المشكلة

المشاكل دائماً تظهر أكثر تعقيداً عن الحقيقة التي هي علىها وذلك لعدم فهم المشكلة. ومن معالجة القاعدة الأولي لديكارت والتي تنص على التأكد مما تريد يمكن الحصول على القاعدة الأولي لحل المشكلة وهي:

قاعدة ١

حلل المشكلة بعناية فائقة محاولاً فهم كل جزئياتها وتحديد كل المتطلبات للحصول على الحل المقبول وفهم كل ما يؤدي للحصول على الحل المقبول للمشكلة.

فإذا وجد حل، بين كيف يمكن العمل لتحقيق هذا الحل. ولذا يجب تحديد مستوي النتائج المطلوبة في المراحل الأولي كما يجب أن تكون الأهداف واضحة ومعلومة وكذلك الوسائل اللازمة لتحقيق هذه الأهداف، وملخص هذه القاعدة هو فهم المشكلة يمثل نصف الحل وكذلك الفهم الجيد والصحيح والكامل للمشكلة يعطي دائماً نتائج واضحة وصحيحة.

تقسيم المشكلة

بزيادة فهم المشكلة يزداد تبعاً له وضوح تفصيلات وأبعاد المشكلة، وبالتالي تصبح المشكلة أكثر تفصيلاً وثباتاً ووضوحاً، مما يجعل من الصعب التعامل مع كل هذه التفاصيل في نفس الوقت، وهذا يوضح القاعدة الثانية لديكارت والتي تنص على :

القاعدة ٢

"حاول أن تقسم المشكلة إلى أجزاء بسيطة وغير معتمدة على بعضها البعض ثم ركز على كل جزء على حدة". وفي هذا الإطار يمكن استخدام العديد من الطرق المختلفة لتقسيم المشكلة، وبذلك يمكن الحصول على القواعد الفرعية التالية من القاعدة الثانية.

قاعدة ٢ أ

حاول تقسيم المشكلة إلى مجموعة مشاكل (أجزاء) بسيطة متتابعة، وحتي نحصل على الحل الكامل للمشكلة الأصلية بحل المشاكل الفرعية البسيطة الواحدة تلو الأخرى.

والغرض من تقسيم المشكلة هو العمل مع جزء واحد فقط وعزل تأثير الأجزاء الأخرى حتي يسهل التعامل معه، ولكن يجب عدم إهمال ما تقوم به الأجزاء الأخرى من المشكلة لأنه لا يمكن أن تكون معزولة نهائياً عن باقي الأجزاء، ومن المؤكد أن بعض أجزاء المشكلة يجب أن ينظر له ويتم التعامل معه أولاً لأن الأجزاء الأخرى تتأثر به أو تعتمد على النتائج التي تنتج منه. وعند حل كثير من المشاكل فإن ذلك يتضمن تكرار التعامل مع بعض الحالات والأوضاع مثل المستهلكين، ونتائج التجارب.....إلخ، وفي مثل هذه المشاكل (الحلات) يجب التأكيد على كيفية التعامل مع الحالات الفردية. وإذا كان حل أحد هذه المشاكل (المسائل) كافياً وصحيحاً يمكن للمبرمج أن يعيد استخدام هذا الحل لكل المشاكل المشابهة في جميع الحالات.

قاعدة ٢ ب

إذا كانت المشكلة تتضمن بعض العمليات التي يعاد تكرارها حاول عزل العمليات التي لا تتطلب الإعادة من تلك التي تتطلب الإعادة.

إذا كنت لا تستطيع أن تقرر من أين تبدأ فإن هذا يحدث لوجود بعض الحالات الخاصة التي تسبب إزعاجاً عند فصلها. وفي هذه الحالة يكون من المفيد أن يتم إهمال هذه الحالات الخاصة وكذلك الحالات غير المفيدة وغير النافعة في البداية ثم في نهاية الحل يمكن التعامل مع جميع الحالات بما فيها الحالات الخاصة وذلك بعد إجراء بعض التعديلات البسيطة على الحل المقترح.

قاعدة ٢ ج

في البداية حاول إيجاد حل للمشاكل في الحالات البسيطة أو الحالات المشهورة وعند الوصول إلي حل مرضٍ وصحيح يمكن تطوير هذا الحل ليشمل الحالات الخاصة والمعقدة.

ومن هذه القاعدة نستنتج أن التعامل مع الحالات البسيطة والمشهورة وعند الحصول منها على نتائج مرضية فإن ذلك يشجع على إمكانية الوصول إلى حل للحالات الخاصة. وأما إذا لم نستطيع الحصول على نتائج في الحالات البسيطة فلن نستطيع الحصول على نتائج

صحيحة في الحالات الخاصة والمعقدة. ونلخص ذلك بأن تبدأ بالتعامل مع الأجزاء البسيطة ثم تتدرج إلى الأصعب فالأصعب وهكذا.

عملية حل المشاكل

القواعد المؤدية للحل يمكن أن تطبق بطرق مختلفة، كما أنها يجب أن تطبق ببطء وعناية وهذا ما توضحه القاعدة الثالثة.

قاعدة ٣

"عند تقسيم المشكلة إلى أقسام صغيرة يجب أن يكون التقسيم على خطوات متعددة بحيث تستخدم القواعد العامة في المراحل الأولي ثم يتم الانتقال إلى المراحل الخاصة بعد ذلك"

المراحل الأولي في الحل تتطلب اعتبارات عامة وواسعة بينما المراحل المتأخرة تتطلب التركيز على التفاصيل والانتقال من العام إلى الخاص وهذا ما يعرف بطريقة من الأعلى إلى الأسفل top-down design. ويقترح ألا يتجاوز عدد الأجزاء المقسمة في كل خطوة ٥ أجزاء. والقاعدة الأساسية في عملية التقسيم هي أن يستمر التقسيم حتي يمكن عزل الأجزاء عن بعضها البعض، وأن يكون حل هذه الأجزاء سهلاً. والقدرة على التقسيم تتطلب مهارة عالية وخبرة إلا أن هذه الخبرة يمكن اكتسابها وتطويرها وتنميتها.

قاعدة ٤

"في كل مرحلة من المراحل يجب مراجعة الحل المقترح ليتم التأكد من أنه كامل وصحيح".

يعني ذلك أن مراجعة واحدة للحل لن تكون كافية ويجب تطبيق القاعدة الرابعة عند كل مرحلة. بعد حل واحد من البرامج الفرعية أو الأجزاء يجب إعادة النظر في الحل المقترح لنري إذا كان يحقق المطلوب بدقة من هذا البرنامج الفرعي، وعند تجميع حلول البرامج الفرعية يجب التأكد من التوافق بين كل هذه الحلول للبرامج الفرعية والتأكد من أنها تحقق المطلوب وأنها تأخذ في الحسبان كل الحالات الخاصة. وأخيراً لا تردد في مراجعة الحلول المقترحة فإنك سوف تجد شيئاً ما يجب أن يضاف أو يعدل أو يحذف.....إلخ.

خوارزمية الحل *Algorithm*

بعد أن استعرضنا خطوات التفكير لحل أية مسألة برمجية وقبل أن ندخل في تفاصيل كتابة الخوارزم لحل المسألة نقول أن الحل يمر بمرحلتين.

المرحلة الأولي

هذه المرحلة تمثل دور الإنسان في حل المسألة وتتكون من عدة خطوات تعرضنا لها فيما سبق ونجملها فيما يأتي:

- **تحديد معالم المسألة.**
- **تحليل عناصرها، وذلك بمعرفة معطياتها، والهدف الأساسي لها، وأهم النتائج المطلوبة منها، وما هي الصورة المراد عرض النتائج فيها، وكذلك صورة تقديم المعطيات.**
- **البحث والتفكير في طريقة حل المسألة.**
- **تدوين الحل في خطوات متسلسلة متعاقبة، يعبر عنها باللغة العادية محكومة بالمنطق الرياضي. هذه الخطوات في مجموعها تسمي الخوارزم Algorithm، كما يمكن تمثيل هذه الخطوات والارتباط فيما بينها بما يعرف بخريطة التدفق Flowchart، وذلك لكي تساعد في تسلسل المنطق العام لحل المسألة– وسوف نتعرض بالتفصيل لشرح كل من الخوارزم وخرائط التدفق لاحقا في هذا الفصل.**
- **كتابة البرنامج.**

المرحلة الثانية

وهذه المرحلة تمثل دور الحاسب نفسه في حل المسألة، والتي تبدأ بترجمة البرنامج المكتوب بلغة المستوى العالي إلى لغة الآلة بواسطة المترجم Compiler، ومن ثم يقوم بحفظ البرنامج في الصورة الجديدة حتي يتم تنفيذه بعد ذلك لإخراج النتائج إلى الوسط الخارجي، ليقوم المستخدم بالاستفادة منها بالشكل الذي يريده وذلك عند عدم وجود أخطاء في البرنامج. أما في حالة وجود أخطاء في البرنامج فإنه يجب تصحيح هذه

الأخطاء أولاً ثم تعاد الترجمة مرة ثانية وهكذا حتي نحصل على برنامج بدون أخطاء ثم بعد ذلك يتم تنفيذ البرنامج.

الخوارزميات (Algorithms)

لقد استخدمت كلمة الخوارزمية، في القرن الماضي، وبشكل واسع، في أوروبا وأمريكا، وكانت تعني، الوصف الدقيق لتنفيذ مهمة من المهمات، أو حل مسألة من المسائل. وقد اشتق الغربيون هذه الكلمة من اسم عالم الرياضيات المسلم المعروف، محمد بن موسى الخوارزمي.

وتستخدم كلمة الخوارزمية، على نطاق واسع، في علوم الرياضيات والحاسب، الآن حيث تعرف بأنها:

مجموعة الخطوات (التعليمات) المرتبة، لتنفيذ عملية حسابية، أو منطقية، أو غيرها بشكل تتابُعي متسلسل ومنظم.

إن أي خوارزمية تتكون من خطوات مرتبة، بعضها إثر بعض، وكل خطوة تعتبر بنفسها وحدة من وحدات البناء الكامل للخوارزمية، ويختلف حجم هذه الخطوات باختلاف الخوارزميات، واختلاف الأشخاص، الذين يقومون بتنفيذ تلك الخطوات. والمثال التالي يوضح معني الخوارزمية:

مثال: إذا أردنا أن نوجد متوسط درجات الحرارة: (T3,T2,T1) مثلاً فإن خطوات الحل المنطقية يمكن ترتيبها في الخوارزمية التالية:

الخطوة الأولي: اقرأ قيم درجات الحرارة: (T3,T2,T1)

الخطوة الثانية: احسب متوسط درجات الحرارة، (AV)، من المعادلة:

(AV=(T1 + T2 + T3) / 3)

الخطوة الثالثة: اطبع النتيجة

مثال آخر:

أراد شخص أن يحسب الزكاة (z) عن أمواله النقدية، (CM) والتي بلغت النصاب الشرعي، بعد مرور حول قمري عليها، وهي في حوزته، فكيف يفعل؟

الحل: من المعروف أن قيمة الزكاة تحسب بنسبة 2.5%، من قيمة المال البالغ النصاب، ولذا فإن خطوات الحل يمكن ترتيبها على النحو التالي:

الخطوة الأولى: اقرأ قيمة ما بحوزته من مال نقدي بالغ النصاب، CM
الخطوة الثانية: احسب قيمة الزكاة المستحقة Z، من المعادلة Z=.025 CM
الخطوة الثالثة: اطبع النتيجة Z

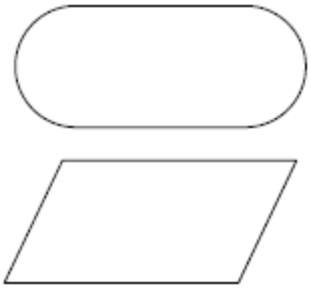
خرائط التدفق (المسار) Flow charts

تستخدم خرائط التدفق في بيان خطوات حل المسألة وكيفية ارتباطها ببعض، باستخدام رموز اصطلاحية لتوضيح خطوات الحل، وهذه الرموز مبينة بشكل رقم (1-1).

أهمية استخدام خرائط التدفق (المسار)

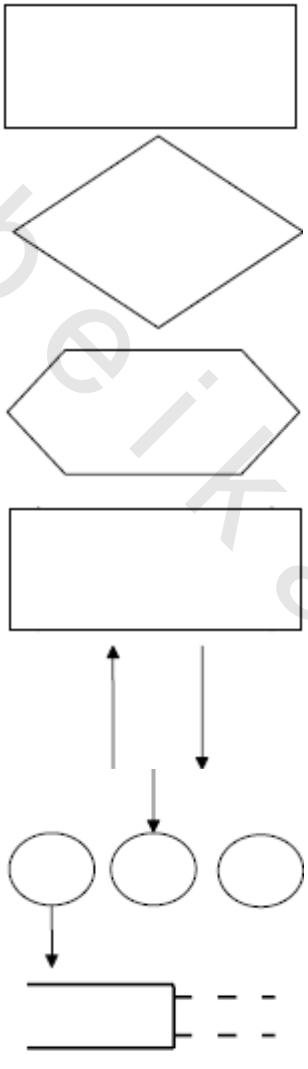
من أهم فوائد استخدام خرائط التدفق قبل كتابة أي برنامج، الأمور الآتية:

١. **تعطي صورة متكاملة للخطوات المطلوبة لحل المسائل في ذهن المبرمج، بحيث تمكنه من الإحاطة الكاملة بكل أجزاء المسألة من بدايتها وحتى نهايتها.**
٢. **تساعد المبرمج على تشخيص الأخطاء التي تقع عادة في البرامج، وبخاصة الأخطاء المنطقية منها، والتي يعتمد اكتشافها على وضع التسلسل المنطقي، لخطوات حل المسألة لدي المبرمج.**
٣. **تيسر للمبرمج أمر إدخال أي تعديلات، في أي جزء من أجزاء المسألة، بسرعة، ودون الحاجة لإعادة دراسة المسألة، برمتها من جديد.**
٤. **في المسائل التي تكثر فيها الاحتمالات والتفرعات، يصبح أمر متابعة دقائق التسلسل، أمراً شاقاً على المبرمج، إذا لم يستعن بمخطط تظهر فيه خطوات الحل الرئيسية بشكل واضح.**



١) بداية أو نهاية البرنامج (STRRT/STOP)

٢) إدخال أو إخراج (INPUT/OUTPUT)



٣) عمليات حسابية وتخزين
(CALCULATION AND STORAGE)

٤) تقرير
(DECESION)

٥) تكرار أو دوران
(LOOPINO)

٦) استدعاء برنامج فرعي
(CALL SUBROUTINE)

٧) اتجاه سير البرنامج
(FLOW LINE)

٨) نقطة توصيل وربط
(CONNECTOR)

٩) تعلق وإيضاح
(COMMENT)

شكل 1-1 الرموز الاصطلاحية لخرائط التدفق (المسار)

٥. تعتبر رسوم خرائط التدفق المستعملة في تصميم حلول بعض المسائل، مرجعاً، في حل مسائل أخرى مشابهة، ومفتاحاً لحل مسائل جديدة لها علاقة مع المسائل القديمة المحلولة، فتُشبَّه رسوم خرائط التدفق (المسار)، والحالة هذه، بالرسوم التي يضعها المهندس المعماري عند تصميمه بيتاً أو عمارة، أو مسجداً.....الخ.

أنواع خرائط التدفق (المسار)

بشكل عام، يمكن القول بأن هناك نوعين، رئيسين من خرائط العمليات وهما:

أ) خرائط سير النظم System Flowcharts

يستخدم هذا النوع من الخرائط عند تصميم الأجهزة الهندسية، في المصانع وغيرها، والتي تستعمل أنظمة تحكم ذاتية، مثل العوامة في خزانات المياه، وإشارات المرور الضوئية، وأجهزة ضبط الضغط ودرجات الحرارة في أبراج تقطير البترول، فتعتبر خرائط التدفق هنا، بمثابة المخطط الكامل الذي يبين ترتيب، وعلاقة، و وظيفة، كل مرحلة بما قبلها، وبما بعدها، داخل إطار النظام المتكامل، ويمكن تلخيص الدور الذي تقدمه هذه الخرائط بما يأتي:

١. **تبين موقع كل خطوة من الخطوات الأخرى المكونة للنظام، بحيث يسهل اكتشاف أي خلل يحدث في النظام كله بمجرد النظر، مما ييسر عمليات صيانة الأجهزة، وبأقل التكاليف.**
٢. **تسهل إجراء التعديلات التي قد تطرأ مستقبلاً على برنامج النظام في أي موقع منه.**
٣. **بيان التفصيلات عن المعطيات المطلوب إدخالها إلى النظام.**
٤. **بيان التفصيلات عن أنواع النتائج المتوقعة أو المطلوبة من البرنامج المعد للنظام.**
٥. **بيان طرق ربط النظام، ببقية الأنظمة الموجودة في المؤسسة المعنية.**

ب) خرائط سير البرامج Programs Flowchart

ويستعمل هذا النوع من الخرائط، لبيان الخطوات الرئيسية، التي توضع لحل مسألة ما، وذلك بشكل رسوم اصطلاحية، تبين العلاقات المنطقية، بين سائر خطوات الحل، وموقع ووظيفة كل منها في إطار الحل الشامل للمسألة.

هذا، ويمكن تصنيف خرائط سير البرامج هذه إلى أربعة أنواع رئيسية هي:

١. **خرائط التتابع البسيط** Simple Sequential Flowcharts
٢. **الخرائط ذات الفروع** Branched Flowcharts
٣. **خرائط الدوران الواحد** Simple-Loop Flowcharts

٤. خرائط الدورانات المتعددة Multi-Loop Flowcharts

ويمكن للبرنامج الواحد أن يشمل علي أكثر من نوع واحد من هذه الأنواع، ونتناول فيما يأتي شرح هذه الأنواع بالتفصيل.

خرائط التتابع البسيط

ويتم ترتيب خطوات الحل لهذا النوع من الخرائط، بشكل سلسلة مستقيمة، من بداية البرنامج حتي نهايته، بحيث تنعدم فيها أية تفرعات على الطريق، كما تخلو من أي دورانات مما هو موجود في الأنواع الأخرى من الخرائط. ويكون الشكل العام لهذا النوع كما هو مبين في شكل (2-1)، وفيها يتم تنفيذ الحدث a ثم يليه تنفيذ الحدث b وبعده التوقف.

وكلمة الحدث a، الواردة في شكل (2-1) تعني الحدث أو العملية المطلوب تنفيذها.

مثال (١):

ارسم خريطة سير البرنامج التي تمثل عملية شراء كتاب من مركز بيع الكتب.

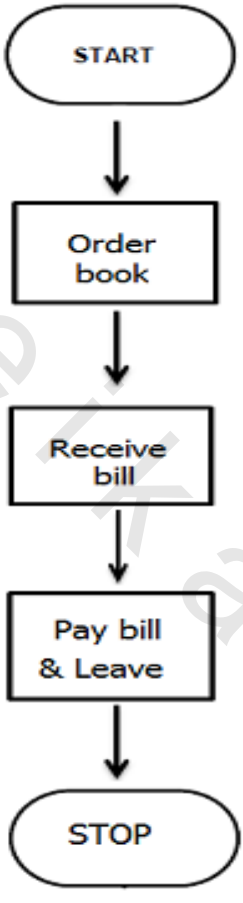
الحل:

خريطة سير البرنامج في الشكل (3-1) يمكن أن تمثلها الخطوات الآتية:

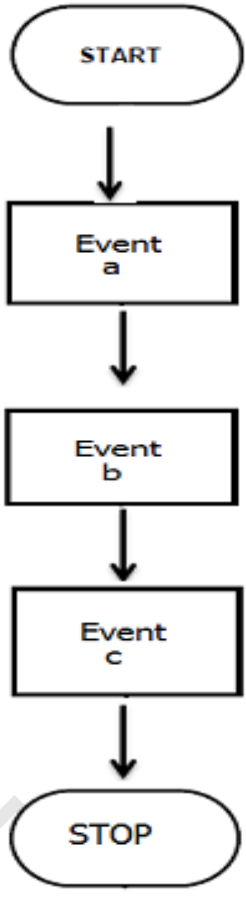
١. اطلب الكتاب.

٢. استلم الفاتورة.

٣. ادفع الفاتورة وغادر.



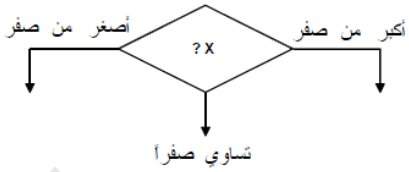
شكل 3-1



شكل 2-1 خرائط التتابع البسيط

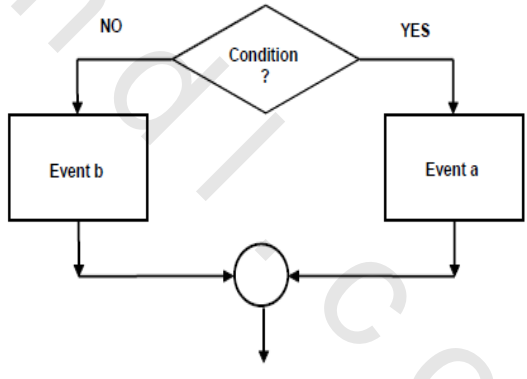
خرائط التفرع

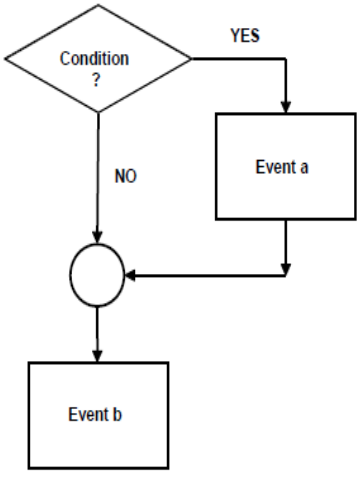
إن أي تفرع يحدث في البرنامج، إنما يكون بسبب الحاجة لاتخاذ قرار، أو مفاضلة بين اختيارين أو أكثر، فيسير كل اختيار في طريق مستقل (تفرع) عن الآخر. وهناك لونان من القرار يمكن للمبرمج استعمال أحدهما حسب الحالة التي يدرسها، والشكل (4–1) يبين هذين المسارين من القرار.



شكل (4–1 a) قرار ذو فرعين شكل (4–1 b) قرار ذو ثلاثة أفرع

وبشكل عام فان خرائط التفرع يمكن أن تأخذ إحدى الصورتين الآتيتين كما هو موضح بشكل (5-1). في شكل (1-5-a) يمكن ملاحظة أنه إذا كان جواب الشرط: نعم فإن الحدث التالي في التنفيذ يكون الحدث (a). أما اذا كان الجواب: لا، فإن الحدث التالي يكون الحدث (b). أما في الشكل (1-5-b)فإننا نلاحظ أنه إذا كان جواب الشرط: نعم، فإن الحدث التالي في التنفيذ يكون الحدث (a) ثم يتبعه الحدث (b). أما اذا كان جواب الشرط: لا، فإن الحدث التالي يكون الحدث (b) مباشرة.





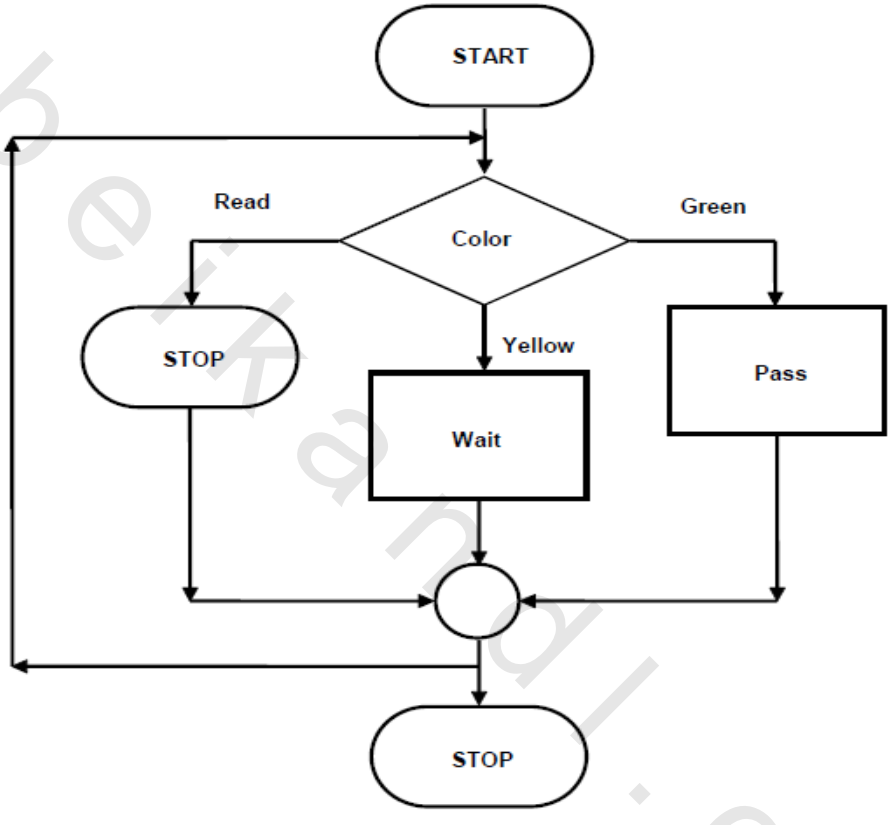
شكل (1-5-a) شكل (1-5-b)

مثال

ارسم خريطة سير البرنامج لإشارات السير الضوئية (اشارات المرور)

الحل:

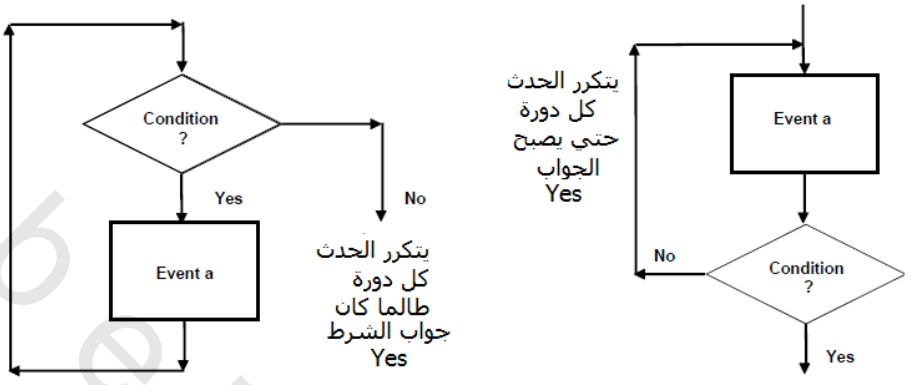
حل هذه المسألة مبين بشكل (6-1)



شكل 6-1

خرائط الدوران الواحد:

وهذه الخرائط نحتاج إلها لإعادة عملية أو مجموعة من العمليات في البرنامج عددا محدودا أو غير محدود من المرات، والشكل العام لمثل هذه الخرائط مبين بشكل (7-1). وقد سميت هذه الخرائط بخرائط الدوران الواحد لأنها تستعمل حلقة واحدة، وتسمي أحيانا خرائط الدوران البسيط.



شكل 7-1

مثال

ارسم خريطة سير البرنامج لإيجاد مساحة مجموعة من الدوائر أنصاف أقطارها معلومة.

الحل: خطوات مبينة في شكل (8-1) وهي:

١. **اقرأ نصف قطر الدائرة R**

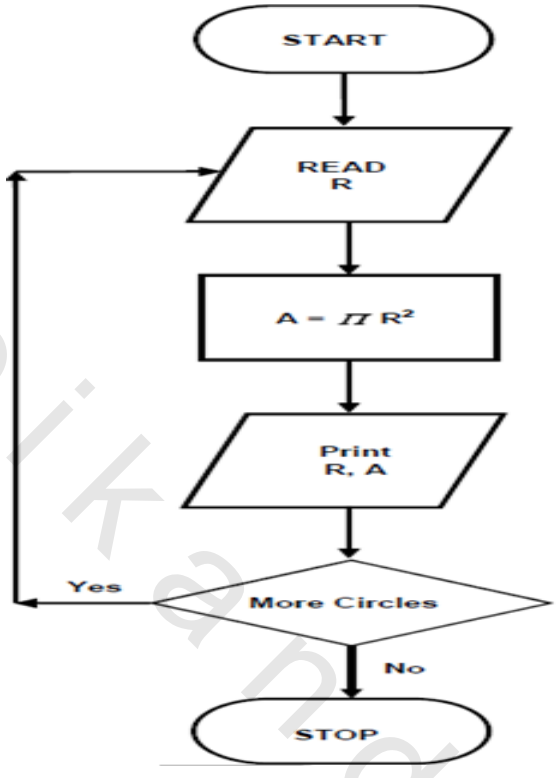
٢. **أوجد مساحة الدائرة A**

٣. **اطبع قيم كل من A,R**

٤. **هل هناك المزيد من الدوائر؟**

إذا كان نعم عد للخطوة ١

أما إذا كان لا فتوقف



شكل (8-1)

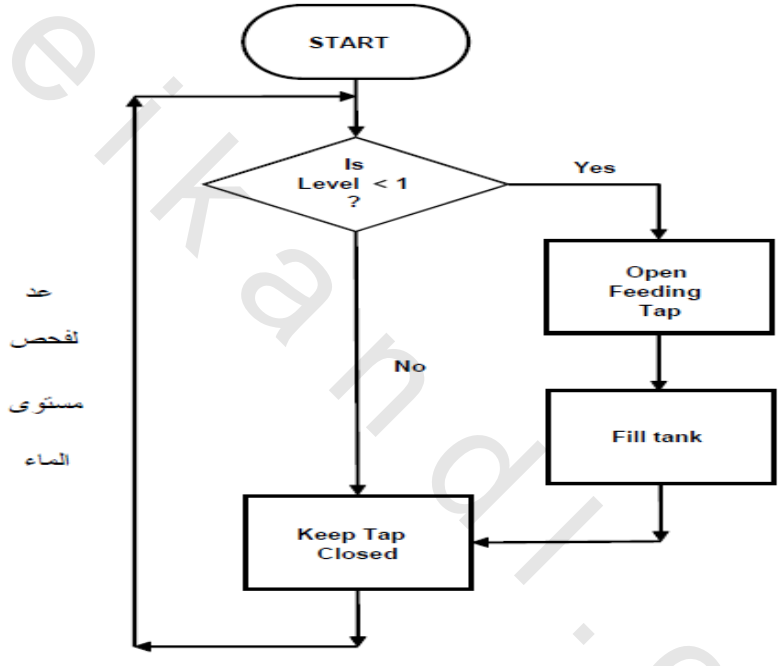
مثال آخر

ارسم خريطة سير البرنامج لخزان يُملأ بالماء ذاتياً (أتوماتيكياً)، عندما يصبح ارتفاع مستوي الماء فيه أقل من متر.

الحل: من المعلوم أن عملية ملء الخزان تقوم على فكرة وجود العوامة التي تفتح صنبور التغذية ذاتياً عندما يصل ارتفاع الماء حداً معيناً (متراً واحداً في هذا المثال) وتغلق صنبور التغذية عند وصول مستوي الماء في الخزان إلى الارتفاع المطلوب وبالتالي فإن خطوات الحل المبينة في الشكل (9-1) تكون كما يأتي:

١. هل مستوي الماء أقل من متر ؟ إذا كان الجواب نعم فاذهب إلى الخطوة (2) وإذا كان الجواب لا، فاذهب إلى الخطوة (4).

٢. يفتح صنبور التغذية.
٣. يملأ الخزان إلى المستوي المطلوب.
٤. أغلق الصنبور (أو حافظ علية مغلقاً).
٥. عُد إلى الخطوة (1) لفحص مستوي الماء مرة بعد مرة للحفاظ على الوضع المطلوب وبشكل دائم.

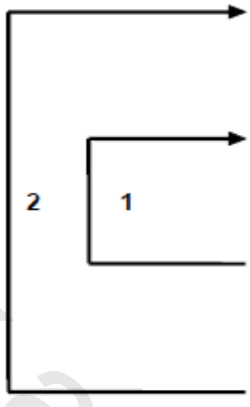


شكل (9-1)

خرائط الدورانات المتعددة

في هذه الحالة تكون الدورانات داخل بعضها البعض بحيث لا تتقاطع فإذا كان لدينا دورانان من هذا النوع انظر شكل (10-1) فيسمي الدوران رقم (1) دوراناً داخلياً Inner Loop بينما الدوران رقم (2) دوراناً خارجياً Outer Loop، ويتم التنسيق بين عمل مثل هذين الدورانين، بحيث تكون أولوية التنفيذ للدوران الداخلي.

وقد سميت هذه الخرائط بخرائط الدورانات المتعددة لأنها تستعمل أكثر من حلقة دوران واحدة، وقد تسمي أحياناً بخرائط الدورانات المتداخلة أو المتراكبة أو الضمنية nested، وكل هذه التسمية تؤدي إلى معني واحد.



شكل (10-1)

مثال

يرغب تاجر في تقطيع مجموعة من قطع القماش طول كلٍ منها يزيد عن 5 أمتار، إلى قطع صغيرة، طول الواحدة منها يساوي 5أمتار، ارسم خريطة سير البرنامج لهذا المشروع.

خطوات الحل المبينة في شكل (11-1) هي:

١. خذ قطعة

٢. اقطع منها قطعة طولها 5 متر

٣. هل المتبقي يزيد عن 5 متر؟

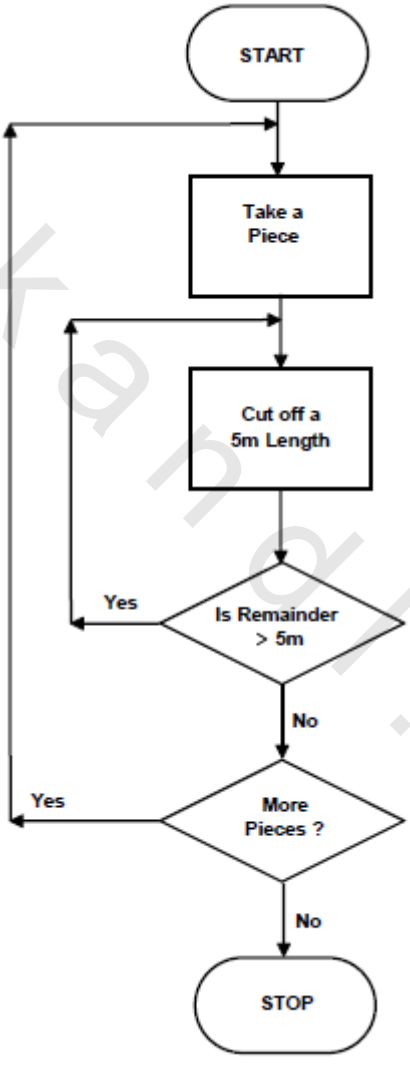
إذا كان الجواب نعم، فاذهب إلى الخطوة (2)

إذا كان الجواب لا، فاذهب إلى الخطوة (4)

٤. هل هناك مزيد من القطع المراد تقطيعها؟

إن كان الجواب نعم، فاذهب للخطوة (1) وإن كان لا، فتوقف.

يلاحظ من الشكل (10-1) أن الدوران الداخلي يتضمن تقطيع القطعة الواحدة إلى قطع متعددة، طول كل منها 5متر، بينما يمثل الدوران الخارجي تناول قطعة واحدة جديدة لتنفذ عليها إجراءات الدوران الداخلي.

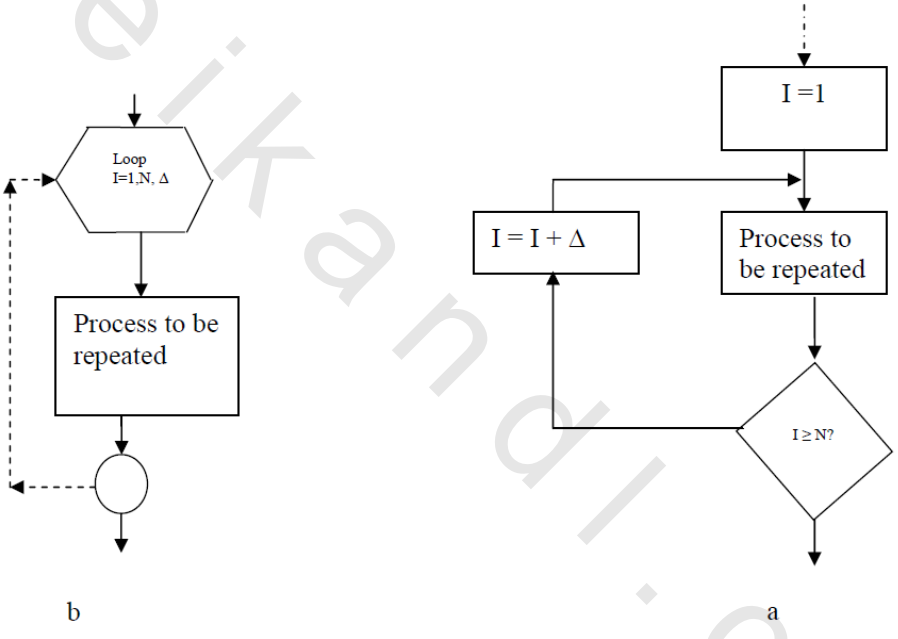


شكل (11-1)

صيغة الدوران باستعمال الشكل الاصطلاحي

لقد عرفنا في الفقرتين السابقتين مفهوم الدوران البسيط والدورانات الضمنية، ويمكننا الآن استخدام الشكل الاصطلاحي للدوران– الوارد ضمن الرموز الاصطلاحية لخرائط سير البرنامج– على النحو التالي:

نلاحظ في الشكل (12-1) أننا نحتاج إلى العناصر الآتية:



شكل (12-1)

العدّاد (I)

القيمة الأولية للعدّاد (I= 1)

القيمة النهائية للعدّاد I (هنا N)

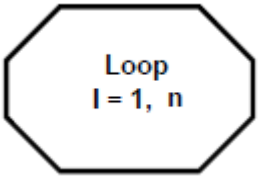
قيمة الزيادة في العدّاد عند نهاية كل دورة (Δ)

نلاحظ من الشكل (a-12-1) أن إجراءات الدوران كانت تتم طبقاً للخطوات الآتية والمفصلة من قبل المبرمج:

١. أعط I قيمة أولية.
٢. أتم الإجراءات المطلوب إعادتها.
٣. اتخاذ قرار: إذا كانت قيمة العداد I وصلت إلى القيمة النهائية N، فاخرج إلى الخطوة التالية في البرنامج وإلا فاذهب إلى الخطوة (4)
٤. زد العدّاد بمقدار الزيادة
٥. عد إلى (2)

يمكننا استبدال الخطوات المفصلة (5,4,3,1) في الشكل (a-12-1) بخطوة مجملة واحدة مبينة في الشكل الاصطلاحي للدوران شكل (b-12-1)، حيث تنفذ هذه الخطوات بصورة أوتوماتيكية من قبل الحاسب. وهذا من شأنه تسهيل عملية البرمجة، واختصار عدد العمليات في البرنامج وتجنب بعض الأخطاء.

تعتبر قيمة تساوي 1 دائماً إذا لم تُعطَ قيمة أخري بخلاف ذلك، وفي حالة عدم ذكر قيمة يصبح الشكل الاصطلاحي الوارد في شكل (b-12-1) كما هو موضح بشكل (13-1) وتكون قيمة الزيادة تساوي 1، بصورة أتوماتيكية.



شكل (13-1)