برمجة الحاسب

مقدمة و حل المشكلة

تمهيد

من المعلوم اليوم أن الحاسبات انتشرت انتشارا واسعا وكبيرا لدرجة أنها أصبحت في كل موقع وفي كل مكان ولا يمكن الاستغناء عنها بأي حال من الأحوال، وذلك لما تقوم به من أعمال كبيرة وعظيمة و لما تتمتع به من قدرة عالية على إجراء العمليات الحسابية وغيرها من العمليات في وقت قصير جدا، كما أنها تتميز بالقدرات العالية على معالجة الكم الهائل من البيانات حفظا وترتيبا واسترجاعا وبحثا وغيرها الكثير من العمليات.

ونظرا لما سبق أصبح لزاما علينا - لكي نواكب هذا العصر ولكي ننهض بوطننا وشعبنا و أمتنا - أن نعرف الكثير عن هذه الحاسبات وكيف يمكن التعامل معها والاستفادة منها. ومن الوسائل التي تساعدنا على الاستفادة من هذه الحاسبات معرفة وإتقان إحدى لغات البرمجة المعروفة والمشهورة هذه الأيام، ومن هذه اللغات المشهورة والتي بدأت تستخدم على نطاق واسع لغة الجافا Java language وذلك لما تتمتع به من قدرة على العمل (التنفيذ) مع كل الحاسبات وسهولة كتابة البرامج المختلفة سواء منها البسيطة أو الكبيرة.

وهذه الحقيبة تقدم شرحا تفصيليا للمفردات الأساسية المكونة للغة الجافا وكذلك كتابة بعض البرامج البسيطة والمتوسطة باستخدام هذه اللغة. ففي الوحدة الأولى مقدمة للغات البرمجة المختلفة وشرح لكيفية تحليل وحل المشاكل البسيطة باستخدام خرائط التدفق والكود الزائف وكذلك كتابة البرامج البسيطة ومعرفة المفردات الأساسية للغة من متغيرات وأنواع البيانات والعمليات الحسابية والمنطقية وغيرها من العمليات تم شرحها وتوضيحها في الوحدة الثانية. أما الوحدة الثالثة فإنها تتناول الحلقات (looping) بأنواعها المختلفة والتفريعات (branching) وكيفية كتابتها والاستفادة منها في حل البرامج البسيطة والمتوسطة، بالإضافة إلى تنفيذ هذه البرامج على الحاسب.

الوحدة الأولى

مقدمة وحل المشكلة

في هذه الوحدة نعرض مقدمة عن ماهية برنامج الحاسب ولغة البرمجة وأنواع لغات البرمجة وأهمية مهنة البرمجة، ثم بعد ذلك نشرح القواعد التي تساعد في تحليل المشكلة ومعرفة عناصرها المكونة لها و كيف يمكن تجزئة المشكلة إلى أجزاء صغيرة يسهل التعامل معها، وفيها أيضا نوضح رموز رسم خرائط التدفق ثم رسم هذه الخرائط للمشكلة بعد كتابة الخوارزم والتي تعطي صورة لحل المشكلة.

الفصل الأول: مقدمة

الجدارة:

معرفة ماهية برنامج الحاسب ولغات البرمجة وأنواعها

الأهداف:

عندما تكمل هذه الوحدة يكون لديك القدرة على:

١ - فهم ماهية برنامج الحاسب
٢ - معرفة لغات البرمجة المختلفة التي يمكن أن يراها
٣ - الإخبار بأهمية مهنة البرمجة
٤ - معرفة ما هو علم صناعة البرمجيات

مستوى الأداء المطلوب

أن يصل المتدرب إلى إتقان هذه الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ساعة واحدة

الوسائل المساعدة:

- قلم
- دفتر

متطلبات الجدارة:

اجتياز جميع الحقائب السابقة

الفصل الأول: مقدمة

نظراً للتطور الكبير في تقنية صناعات الحاسبات الآلية وانتشارها في جميع مجالات الحياة المختلفة، واستخداماتها المتعددة في شتى المجالات، فإنه أصبح لزاماً علينا معرفة هذه الحاسبات وكيفية التعامل معها والاستفادة منها لأنها توفر الجهد والوقت وتنجز كثير من الأعمال بدقه كبيرة بالإضافة إلى قدراتها الكبيرة في الاحتفاظ بالبيانات. ومن الطرق الشائعة للاستفادة من القدرات الكبيرة للحاسبات هو: بناء البرامج التي تقوم بحل كثير من المشكلات توفيراً للجهد والوقت ووصولاً إلى الدقة المطلوبة، وفي هذه الوحدة سوف نلقي الضوء على ماهية برنامج الحاسب وكذلك أنواع لغات البرمجة المختلفة. ثم بعد ذلك نبين أهمية مهنة البرمجة وصناعة البرمجيات.

برنامج الحاسب

البرنامج هو عبارة عن مجموعة من التعليمات تعطى للحاسب للقيام بعمل ما مثل حساب مجموع قيم مختلفة، حساب المتوسط الحسابي، حساب مضروب عدد معين .....الخ
والبرنامج هو الذي يحدد للحاسب كيفية التعامل مع البيانات للحصول على النتائج المطلوبة. والبرنامج يكتب بواسطة المبرمج (Computer Programmer) الذي يفهم المشكلة ويقترح الحل وينفذه لحل هذه المشكلة ويجب أن يكون البرنامج في مجموعه صحيحاً وواضحا وليس فيه لبس أو غموض.

والبرمجيات (Software) هي التي تسهل للمستخدم استخدام المكونات المادية (Hardware) بكفاءة وراحة ويمكن تقسيم البرمجيات إلى ثلاثة أنواع رئيسية وهي: -

١ - **برامج التشغيل** Operating System

مثل النوافذ(windows) و Dos، Unix، Linux، VMS وغيرها. وهي عبارة عن برامج تقوم بدور الوسيط بين المستخدم والمكونات المادية وهي تمكن المستخدم من استخدام المكونات المادية للحاسب بكفاءة وبراحة، كما أنها تساعد المستخدم في إنشاء نظام الملفات وغيرها. ومن برامج التشغيل ما يصلح للعمل في الشبكات مثل Unix، Windows، ومنها الذي يستخدم مع الحاسب فقط مثل Dos.

٢ - **برامج التطبيقات** Application Programs

وهي برامج تساعد في إنشاء كثير من التطبيقات مثل إنشاء قاعدة بيانات والرسم باستخدام الحاسب و غيرها ومن أمثلة هذه البرامج: -

برنامج الأوتوكاد Autocad– الاكسيل Excel - الأكسس Access - الأوراكلOracle - الفوتوشوب Fotoshop وغيرها كثير.

٣ - **لغات البرمجة** Programming Languages

وهذه اللغات هي التي تستخدم في بناء البرامج المختلفة وهي تتراوح من اللغات التي تتعامل مباشرة مع المكونات المادية للحاسب والأخرى التي تتطلب تحويلها من صورتها التي تكتب بها إلى صورة أخرى يستطيع الحاسب التعامل معها.

ويوجد العديد من لغات البرمجة المستخدمة اليوم وهذه اللغات يمكن تقسيمها إلى ثلاث أنواع رئيسية هي: -

١ - لغة الآلة **Machine languages**

٢ - لغات التجميع **Assembly languages**

٣ - لغات المستوى العالي **High level languages**

لغة الآلة Machine Language

وهي اللغة الوحيدة التي يفهمها الحاسب ويستطيع التعامل معها. وهذه اللغة تعتبر لغة خاصة لكل حاسب وقد تختلف من حاسب إلى آخر وهي تعتمد على المكونات المادية للحاسب نفسه، ولغة الآلة تتكون من مجموعة أرقام من بين 0، 1 التي تعطي تعليمات للحاسب للقيام بمعظم العمليات الأساسية واحدة بعد الأخرى، وهي تختلف من حاسب إلى حاسب آخر ولذلك فإننا نجد أن نفس البرنامج الذي يعمل على حاسب معين قد لا يعمل على حاسب آخر يختلف عنه في المكونات المادية. و لغة الآلة من اللغات الصعبة في التعلم للإنسان حتى بالنسبة للمبرمجين لأنها عبارة عن مجموعة من الأرقام (٠ ، ١) فقط. وللتغلب على هذه الصعوبة تم اقتراح لغة أخرى تعتمد على استخدام اختصارات معبرة من اللغة الإنجليزية للتعبير عن العمليات الأولية التي يقوم بها الحاسب وهذه اللغة هي لغة التجميع.

لغة التجميع Assembly Languages

هي لغة تستخدم اختصارات معبرة من اللغة الإنجليزية لتعبر بها عن العمليات الأولية التي يقوم بها الحاسب مثل إضافة Add و حفظ Store وطرح Sub وغيرها.

مثال على ذلك

```
Load    A
Add     B
Store   C
```

ونظراً لأن هذه اللغة تستخدم كلمات مختصرة من اللغة الإنجليزية فإنها تحتاج محولاً لكي يحولها إلى لغة الآلة وهو ما يسمى المجمع assembler الذي يقوم بتحويل لغة التجميع إلى لغة الآلة كي يفهمها الحاسب ويستطيع تنفيذها، وبالرغم من تقليل المجهود الملقى على عاتق المبرمج للقيام بعملية البرمجة إلا أنه ما زالت توجد مشقة عند حل أبسط المسائل لأن ذلك يتطلب معرفة وكتابة العديد من التعليمات، وهذا ما دفع المبرمجين للتفكير في لغات أخرى تقلل المجهود الكبير اللازم لكتابة الكثير من التعليمات فكانت لغات البرمجة ذات المستوى العالي.

لغات البرمجة ذات المستوى العالي High Level Languages

وهذه اللغات كتبت بحيث تستخدم بعض الكلمات الإنجليزية العادية بنفس معانيها حيث يقوم كل أمر منها بتنفيذ العديد من الواجبات، وهذه اللغات كسابقتها تحتاج إلى مترجمات Compilers التي تقوم بتحويل التعليمات(الأوامر) إلى لغة الآلة، .وهذه اللغات تستخدم العلاقات والعوامل الرياضية المتعارف عليها. مثال ذلك

$$Sum = A + B + C$$

وهذه اللغات تعتبر سهلة ومرغوبة من وجهة نظر المبرمجين بالمقارنة بلغات التجميع ولغة الآلة وذلك لسهولة كتابتها وفهمها وحل المشاكل باستخدامها، ومن أمثلة هذه اللغات لغة C، C++، الباسكال Pascal، الفورتران Fortran، البيسك Basic، الآدا ADA، الجافا Java وغيرها.

ومن المعلوم أن عملية تحويل البرنامج من لغة ذات مستوى عال إلى لغة الآلة تستهلك وقتاً ولذلك تم تطوير نسخ من لغات المستوى العالي بحيث تستخدم برنامج مفسر Interpreter والذي يقوم بترجمة الكود سطرا سطرا أثناء التنفيذ.

.وبالرغم من أن البرامج المترجمة الناتجة من عملية الترجمة باستخدام المترجم compiler تكون أسرع في التنفيذ عن البرامج التي تستخدم المفسر (Interpreter) إلا أنه يفضل وجود نسخة من اللغة تعمل باستخدام المفسر وذلك لسهولة التغيير والحذف والإضافة والتصحيح. وبعد الانتهاء من كل التعديلات والوصول إلى نسخة نهائية فإنه يتم استخدام المترجم لترجمة البرنامج وإنتاج نسخة تنفيذية حتى تكون أسرع في التنفيذ بعد ذلك عند تشغيلها على الحاسب.

أهمية مهنة البرمجة

من المعلوم أن الذي يقوم بكتابة البرامج لحل المشكلات الكثيرة والمعقدة هم المبرمجون ولا يمكن الاستغناء عنهم بحال من الأحوال لأن دورهم مهم وحيوي وتكثر الحاجة لهم في شتى المجالات وذلك لعمل الآتي: -

١ - كتابة برامج وبناء الأنظمة المختلفة لحل المشاكل وتبسيط التعامل مع الحاسب.
٢ - المسئولية الكاملة عن إصلاح ما يحدث من أعطال أو حل المشاكل التي تحدث في الأنظمة المختلفة.
٣ - بناء واجهة المستخدم المختلفة في كثير من اللغات والتطبيقات.
٤ - بناء نظم التشغيل المختلفة مثل Unix، Windows وغيرها من النظم. فمثلاً تستخدم لغة C في بناء نظام التشغيل Unix.
٥ - برامج المواجهة المختلفة في الأنظمة المختلطة الرقمية و التماثلية.

صناعة البرمجيات

تعتبر صناعة البرمجيات في عصرنا الحالي من الصناعات المهمة جدا والتي تتطور باستمرار نتيجة التطور الهائل في صناعة الحاسبات الآلية، ولذلك فإن هذه الصناعة تتطلب مبرمجين مهرة ولديهم القدرة على تحليل وحل المشاكل بالإضافة إلى إلمام بكل المستجدات والعلوم والتطوير المتعلق بالحاسب وصناعة الحاسبات و ذلك حتى يستطيعوا مواكبة تطوير البرامج والنظم المختلفة للاستفادة العظمى من التقدم في الحاسبات.

تمارين

1- أكمل العبارات الآتية بكلمات مناسبة

أ - من أمثلة برامج التشغيل.....، .....، .....

ب - تُقَسَّم البرمجيات إلى ثلاثة أنواع رئيسية هي: -

١ - .....

٢ - .....

٣ - .....

ج - يوجد العديد من لغات الحاسب العالية المستوى مثل.....، .....، .....، .....

د - برنامج الحاسب هو عبارة عن ..... تُعطَى للحاسب للقيام بعمل ما مثل.....، .....

2- ضع علامة (√) أمام العبارة الصحيحة وعلامة (×) أمام العبارات الخاطئة

أ- برامج التشغيل تقوم بدور الوسيط بين المكونات المادية المكونة للحاسب والمسُتخدم ( )

ب- لغة الآلة تعتبر أسهل لغات البرمجة ( )

ت- البرامج المكتوبة بلغة المستوى العالي يتم تنفيذها مباشرةً ( )

ث- الحاسب لا يفهم إلا لغة الآلة ( )

ج- المترجمات تقوم بتحويل لغة البرمجة إلى لغة الآلة ( )

الفصل الثاني

حل المشكلة Problem Solving

الجدارة:

المساعدة في تحليل المشكلة وتخطيط الحل لهذه المشكلة باستخدام خرائط التدفق والخوارزميات

الأهداف:

عندما تكمل هذه الوحدة يكون لديك القدرة على

1- معرفة أجزاء المشكلة الرئيسية والفرعية

2- تحديد الاحتياجات المطلوبة لحل المشكلة

3- المشاركة بوضع ورسم خريطة التدفق للبرنامج

4- المشاركة في كتابة خوارزمية الحل للمشكلة

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان هذه الجدارة بنسبة 100٪

الوقت المتوقع للتدريب: 8 ساعات

الوسائل المساعدة:

- قلم
- دفتر

متطلبات الجدارة:

اجتياز جميع الحقائب السابقة

الفصل الثاني

حل المشكلة

Problem Solving

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 حتى يتم الانتهاء من المشكلة.
٤ - المراجعة لجميع الأجزاء حتى يكتمل الحل.

وبالرغم من أن هذه القواعد تم وضعها قبل ٣٠٠عام 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

المشاكل دائماً تظهر أكثر تعقيداً عن الحقيقة التي هي عليها وذلك لعدم فهم المشكلة. ومن معالجة القاعدة الأولي لديكارت والتي تنص على التأكد مما تريد يمكن الحصول على القاعدة الأولى 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 مستوى النتائج المطلوبة في المراحل الأولى كما يجب أن تكون الأهداف واضحة ومعلومة وكذلك الوسائل اللازمة لتحقيق هذه

الأهداف ، وملخص هذه القاعدة هو أن فهم المشكلة يمثل نصف الحل وكذلك الفهم الجيد والصحيح والكامل للمشكلة يعطي دائماً نتائج واضحة وصحيح.

تقسيم المشكلة

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

قاعدة٢

"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 ٢أ

حاول تقسيم المشكلة إلى مجموعة مشاكل (أجزاء) بسيطة متتابعة، وحتى 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 حتى 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 ، نتائج التجارب.....الخ، وفي مثل هذه المشاكل (الحالات) 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.

قاعدة٢ب

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.

قاعدة٢ج

في البداية حاول إيجاد حل للمشاكل في الحالات البسيطة أو الحالات المشهورة وعند الوصول إلى 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

قاعدة٣

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

المراحل الأولى 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 حتى 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 لنرى 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 تتردد في مراجعة الحلول المقترحة فإنك سوف تجد شيئاً ما يجب أن يضاف أو يعدل أو يحذف.....الخ.

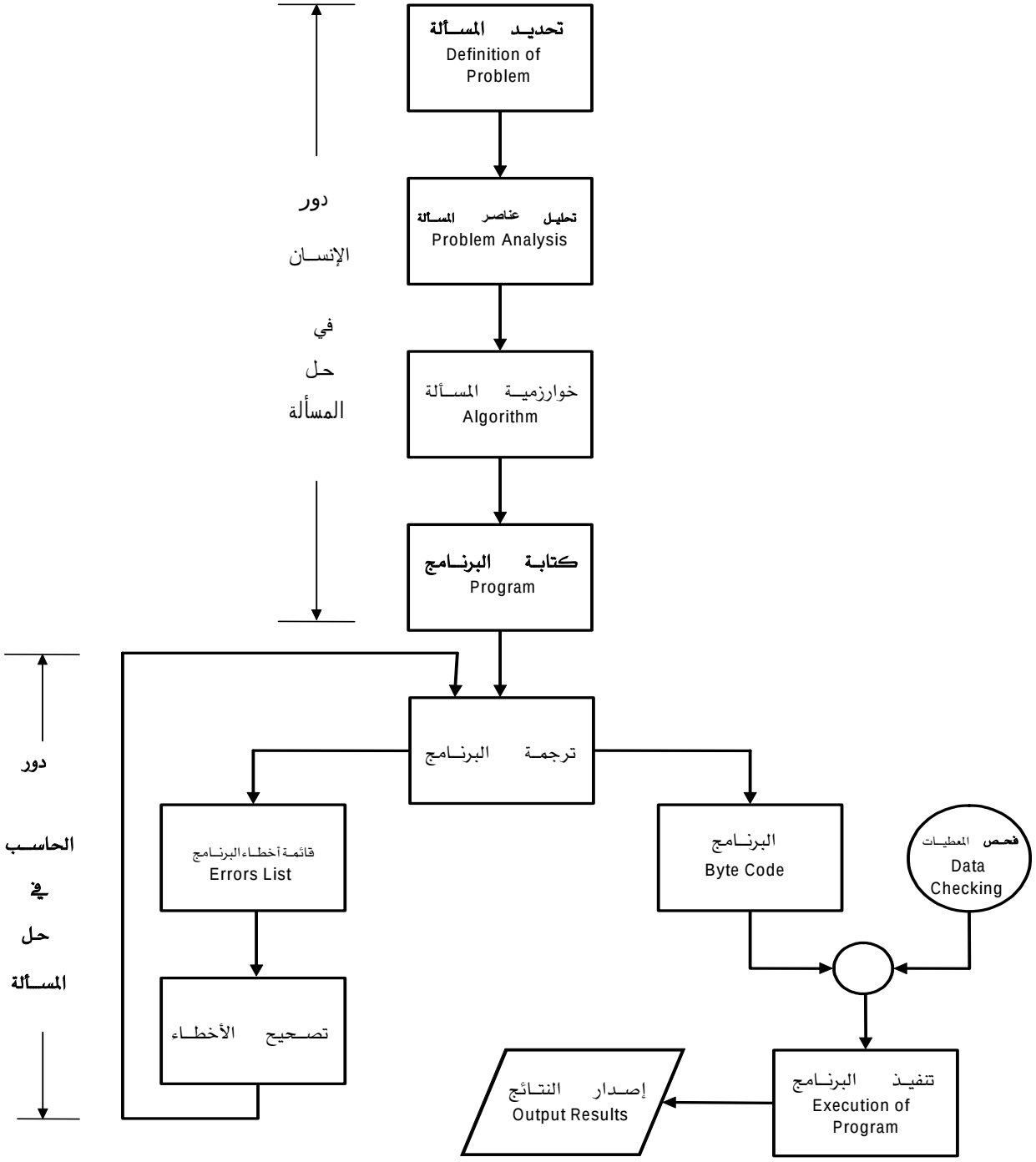
الخوارزم والكود الزائف Algorithm and Pseudo Code

بعد أن استعرضنا خطوات التفكير لحل أية مسألة برمجية وقبل أن ندخل في تفاصيل كتابة الخوارزم لحل المسألة نقول أن الحل يمر بمرحلتين كما هو مبين بشكل (١-1).

المرحلة الأولى

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 بالخوارزم 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 الخوار زم وخرائط التدفق لاحقا في هذا الفصل..
- كتابة البرنامج



شكل (1-1) مراحل حل مسألة باستخدام الحاسب

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 حتى 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 حتى 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 معنى الخوارزمية:

مثال:

إذا أردنا أن نوجد متوسط درجات الحرارة : T_3, T_2, T_1 مثلاً فإن خطوات الحل المنطقية يمكن ترتيبها في الخوارزمية التالية:

الخطوة الأولى: اقرأ قيم درجات الحرارة : T_3, T_2, T_1

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

$$AV = (T_1 + T_2 + T_3) / 3$$

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 عل النحو التالي:

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

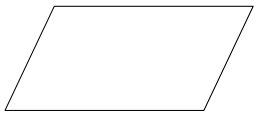
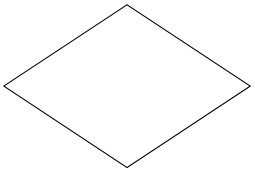
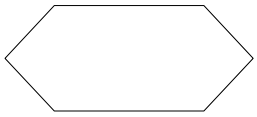
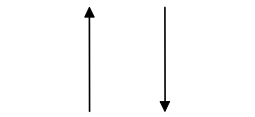
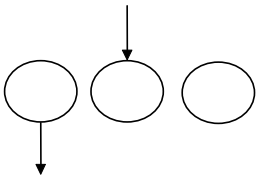
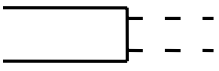
خرائط التدفق Flow charts

تستخدم خرائط التدفق في بيان خطوات حل المسألة وكيفية ارتباطها ببعض، باستخدام رموز اصطلاحية لتوضيح خطوات الحل، وهذه الرموز مبينة بشكل رقم (2-1)

أهمية استخدام خرائط التدفق:

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 لدى 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 الرئيسة بشكل واضح.

(الرمز) الشكل الإصطلاحي	معنى الرمز
	(1) بداية أو نهاية البرنامج (STRRT / STOP)
	(2) إدخال أو إخراج (INPUT / OUTPUT)
	(3) عمليات حسابية وتخزين (CALCULATION AND STORE)
	(4) تقرير (DECISON)
	(5) تكرار أو دوران (LOOPING)
	(6) استدعاء برنامج فرعي (CALL SUBROUTINE)
	(7) اتجاه سير البرنامج (FLOW LINE)
	(8) نقطة توصيل وربط (CONNECTOR)
	(9) تعليق وإيضاح (COMMENT)

شكل(2-1) الرموز الاصطلاحية لخرائط التدفق

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، والحالة هذه، بالرسوم التي يضعها المهندس المعماري عند تصميمه بيتاً أو عمارة، أو مسجداً.... الخ.

أنواع خرائط التدفق

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، ووظيفة، 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 المكوِّنة للنظام، بحيث يسهل اكتشاف أي خلل يحدث في النظام كله بمجرد النظر، مما ييسر عمليات صيانة الأجهزة، و بأقل 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 الرئيسة، 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 رئيسة 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 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 حتى 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 الشكل (3-1) ، وفيها يتم تنفيذ الحدث a ثم يليه تنفيذ الحدث b وبعده التوقف.
وكلمة الحدث a، الواردة في شكل (3-1) تعني الحدث أو العملية المطلوب تنفيذها.

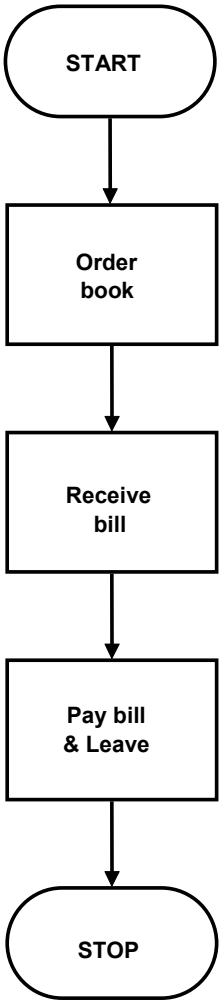
مثال(1):

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

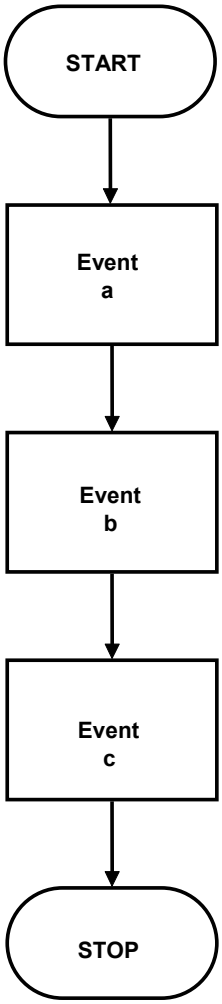
الحل:

خريطة سير البرنامج في الشكل(4-1) يمكن أن تمثلها الخطوات الآتية:

1- اطلب الكتاب
2- استلم الفاتورة
3- ادفع الفاتورة وغادر



شكل (3-1) خرائط التتابع البسيط



شكل (4-1)

مثال ٢

ارسم خريطة سير البرنامج (flow chart) لإيجاد مساحة ومحيط دائرة نصف قطرها معلوم (R)

الحل:

مساحة الدائرة = ΠR^2

محيط الدائرة = $2\Pi R$

حيث Π = النسبة التقريبية وقيمتها العددية ثابتة وتساوي ٣,١٤

بينما R متغير يمثل نصف قطر الدائرة

وحل هذه المسألة كما يأتي :

١- اقرأ قيمة R

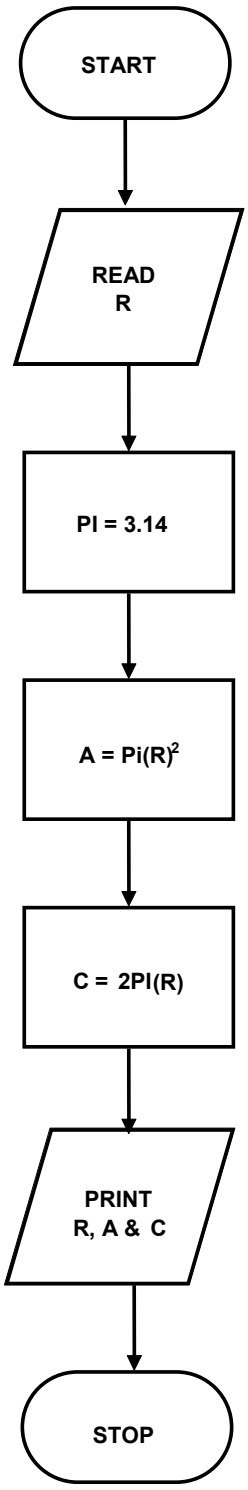
٢- ضع قيمة Π = pi= ٣,١٤

٣- احسب مساحة الدائرة A من المعادلة $A = \Pi R^2$

٤- احسب مساحة المحيط C من المعادلة $C = 2\Pi R$

٥- اطبع قيم كل من : A, R, C

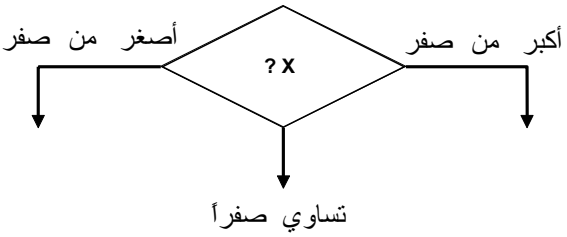
خريطة سير البرنامج التي توضح حل هذه المسألة مبينة في شكل (5-1)



شكل(5-1)

الخرائط ذات الفروع

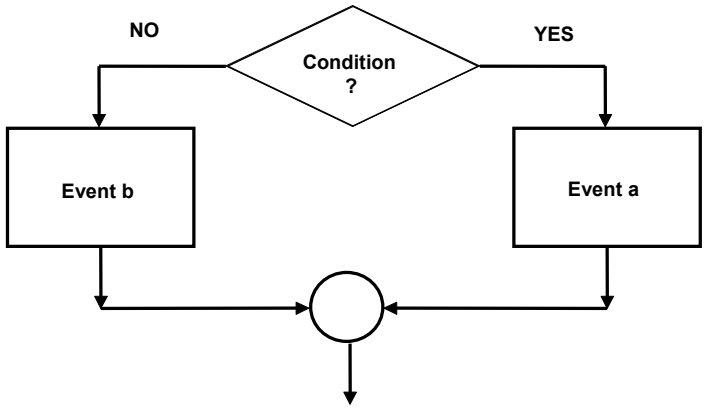
إن أي تفرع يحدث في البرنامج، إنما يكون بسبب الحاجة لاتخاذ قرار، أو مفاضلة بين اختيارين أو أكثر، فيسير كل اختيار في طريق مستقل (تفرع) عن الآخر. وهناك لونان من القرار يمكن للمبرمج استعمال أحدهما حسب الحالة التي يدرسها، والشكل (6-1) يبين هذين المسارين من القرار.



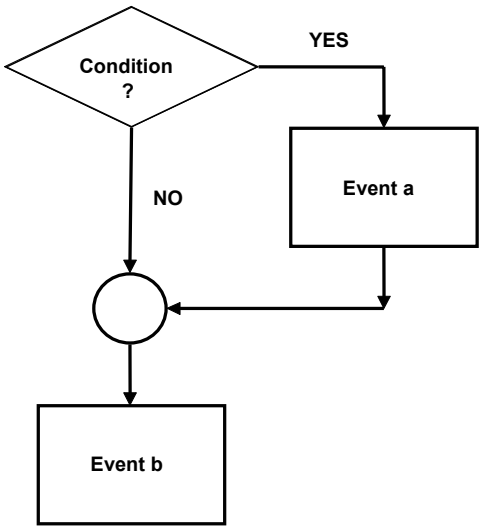
شكل (1-6-b) قرار ذو ثلاثة أفرع

شكل (1-6-a) قرار ذو فرعين

وبشكل عام فإن خرائط التفرع يمكن أن تأخذ إحدى الصورتين الآتيتين كما هو موضح بشكل (1-7). في شكل (1-7-a) يمكن ملاحظة أنه اذا 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 (1-7-b) فإننا نلاحظ أنه اذا 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 (1-7 - a)



شكل (1-7 - b)

مثال ٣

ارسم خريطة سير البرنامج (flow chart) لإيجاد قيمة الدالة F(x) المعرف كما يلي :

$$F(x) = \begin{cases} x & x \geq 0 \\ -x & x \leq 0 \end{cases}$$

الحل:

شكل (8-1) يبين خريطة سير البرنامج لحل هذه المسألة كما يلي:

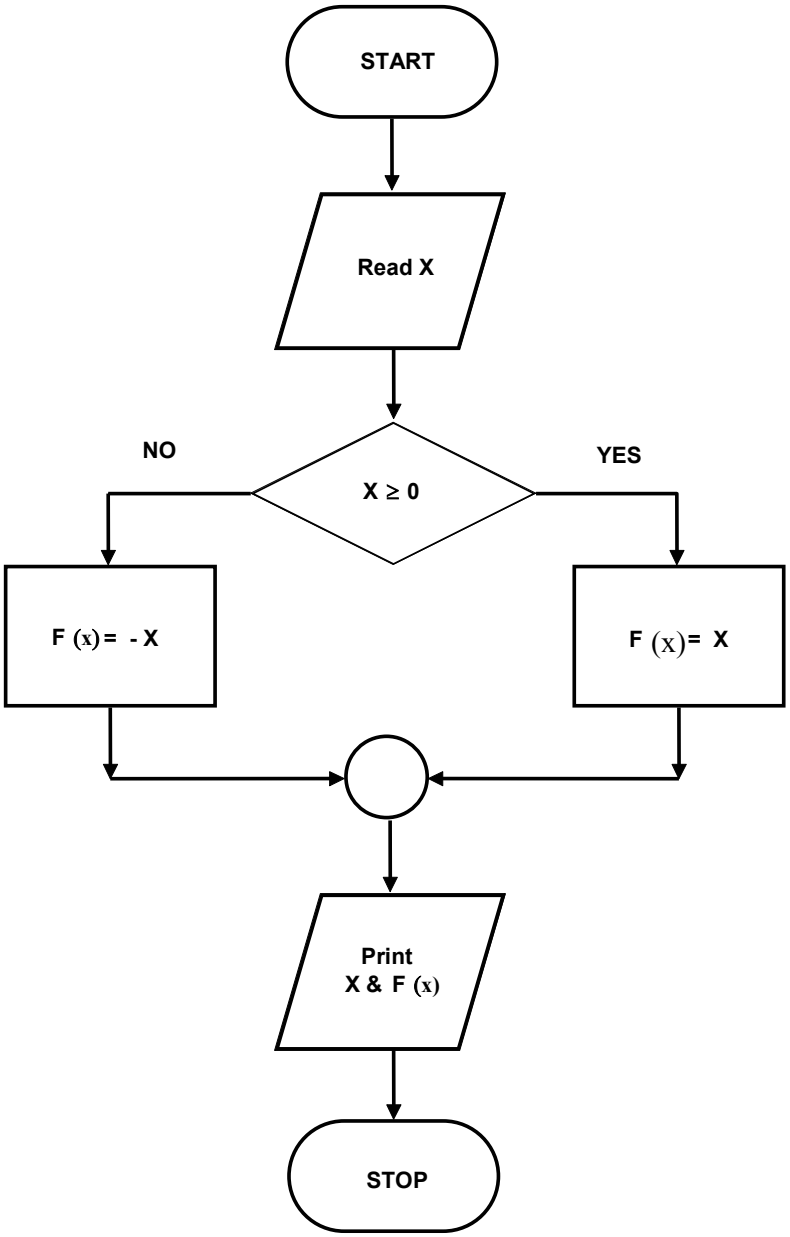
١ - اقرأ قيم المتغير X

٢ - اذا كانت X أكبر من أو تساوي صفرا اذهب إلى خطوة ٣، وإلا فاذهب إلى الخطوة ٤

٣ - احسب قيمة الدالة F(x) من F(x) =X ثم اذهب إلى الخطوة ٥

٤ - احسب قيمة الدالة F(x) من F(x)

٥ - اطبع قيم كل من X , F(x)



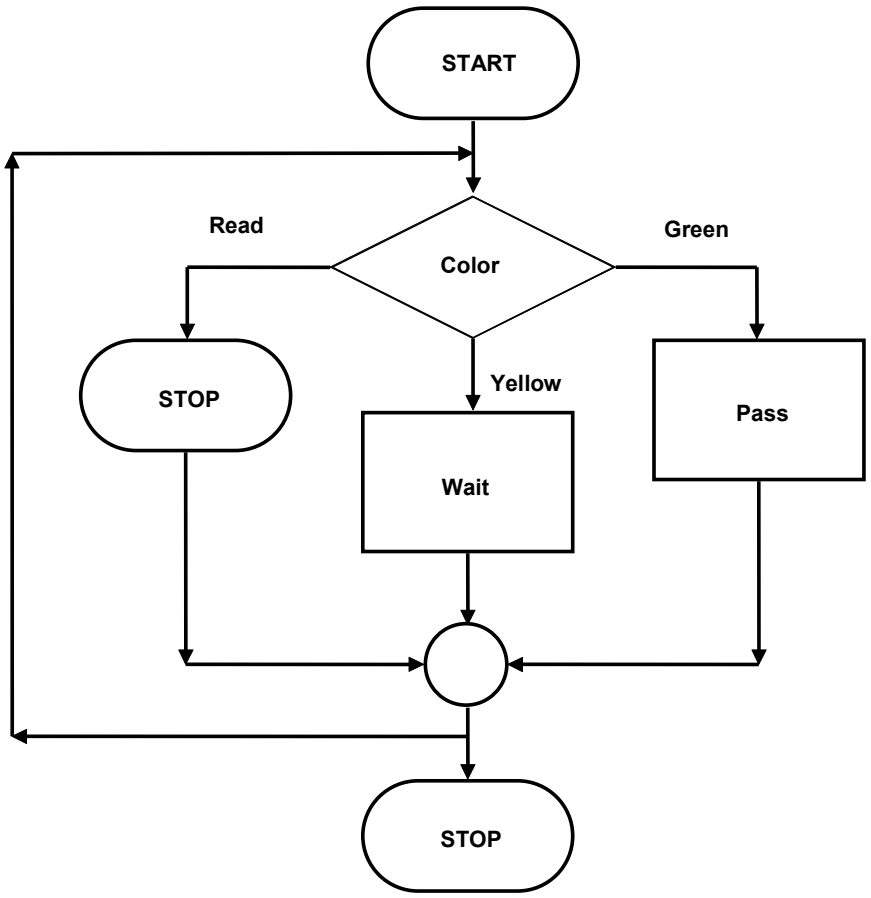
شكل (1-8)

مثال ٤

ارسم خريطة سير البرنامج لإشارات السير الضوئية (اشارات المرور)

الحل:

حل هذه المسألة مبين بشكل (9-1)



شكل (9-1)

مثال ٥

ارسم خريطة سير البرنامج لحساب قيمة W ،
من المعادلات الآتية علماً بـأن قيمة المتغير X
معلومة

$$W = \begin{cases} X+1 & \text{if } x > 0 \\ \sin(x)+5 & \text{if } x = 0 \\ 2X-1 & \text{if } < 0 \end{cases}$$

الحل :

خطوات الحل مبينة في شكل
(1-10) وهي:

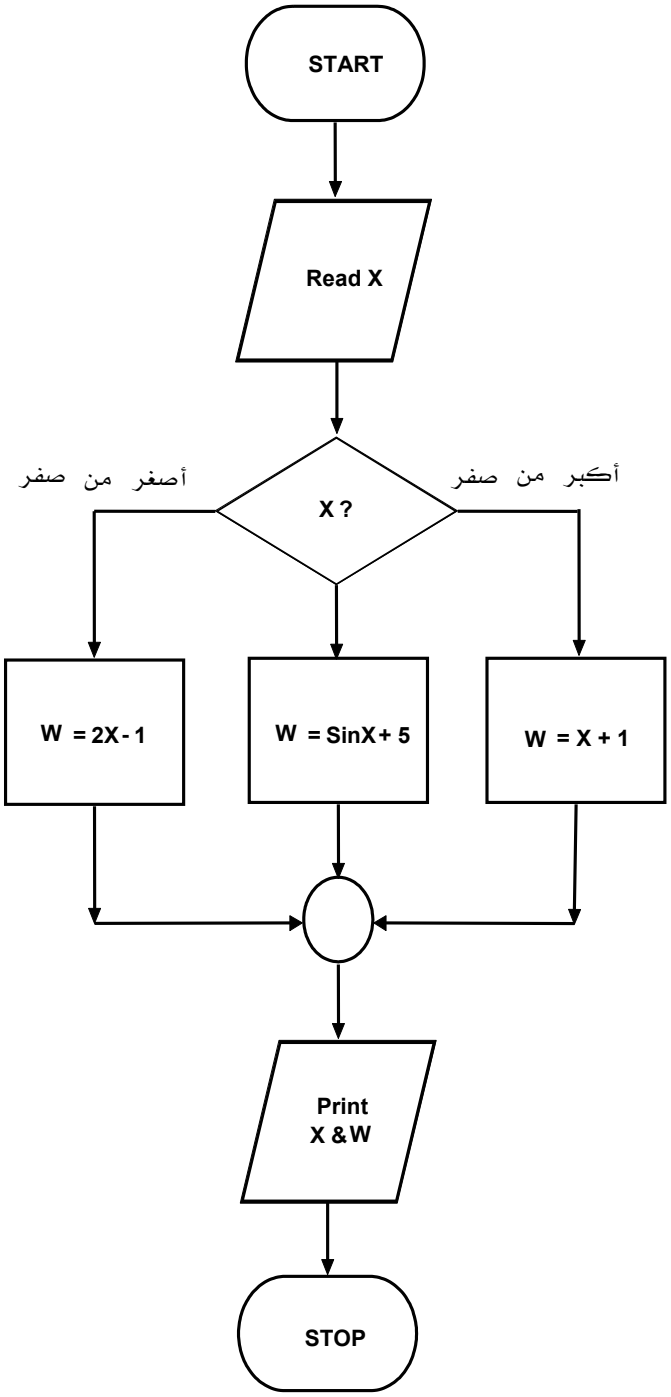
١- إذا كانت X أكبر من الصفر اذهب إلى
الخطوة ٢
إذا كانت X تساوي صفرا اذهب إلى الخطوة ٣
أما إذا كانت X أصغرمن الصفر اذهب إلى
الخطوة ٤

٢ - احسب W من المعادلة W= X+1 ثم
اذهب إلى الخطوة ٥

٣ - احسب W من المعادلة W= sin (X) +5
ثم اذهب إلى الخطوة ٥

٤ - احسب W من المعادلة W=2X -1

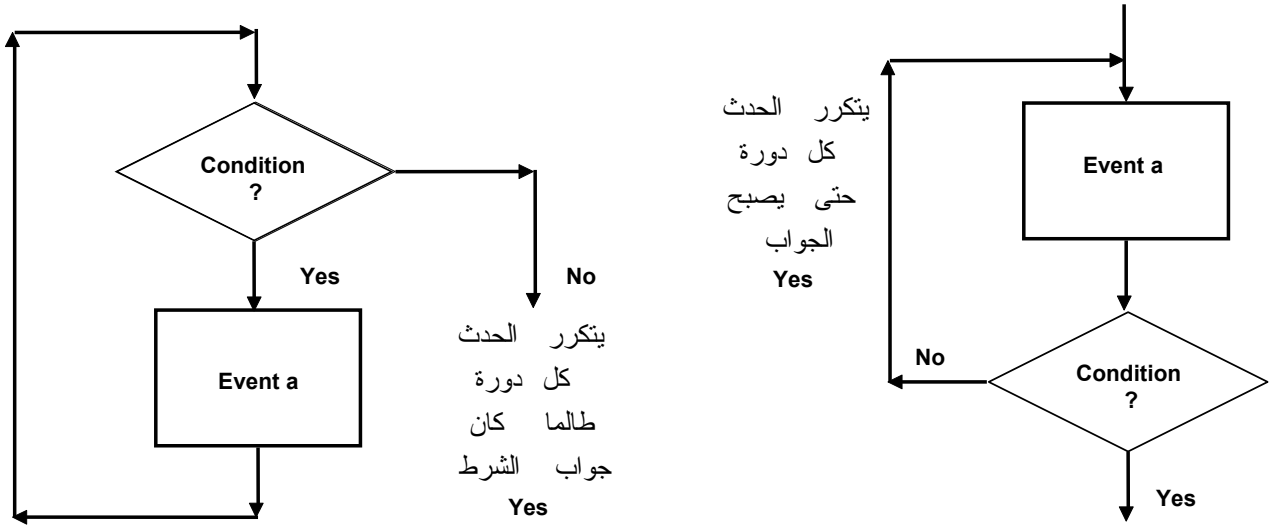
٥ - اطبع قيم كل من X, W



شكل (1-10)

خرائط الدوران الواحد:

وهذه الخرائط نحتاج اليها لإعادة عملية أومجموعة من العمليات في البرنامج عددا محدودا أو غير محدود من المرات، والشكل العام لمثل هذه الخرائط مبين بشكل (11-1). وقد سميت هذه الخرائط بخرائط الدوران الواحد لأنها تستعمل حلقة واحدة، وتسمى أحيانا خرائط الدوران البسيط،



شكل(11-1)

مثال ٦

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

الحل : خطوات الحل مبينة في شكل (12-1) 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

مثال ٧

من واجبات بيت مال المسلمين أن يساعد الشباب على الزواج وذلك بتقديم الدعم المادي المناسب لهم (٣٠٠٠٠ ريال للزيجة الواحدة) على فرض أن السن المناسب للزواج هو عشرون عاما، اقترح خريطة لسير البرنامج لهذا المشروع.

الحل: خطوات الحل مبينة في شكل (13-1) وهي:

١- خذ شابا على الدور

٢- هل مضى من عمره عشرون عاما؟

إن كان نعم اذهب الى الخطوة ٣

أما إن كان لا، اذهب إلى الخطوة ٤

٣- زوج الشاب المذكور

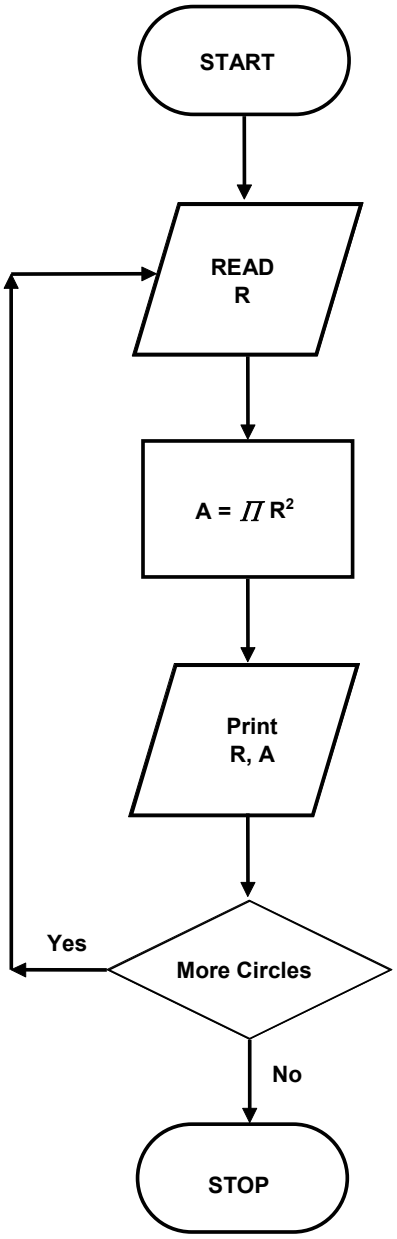
٤- هل هناك مزيد من الشباب؟

إن كان نعم اذهب الى الخطوة ١

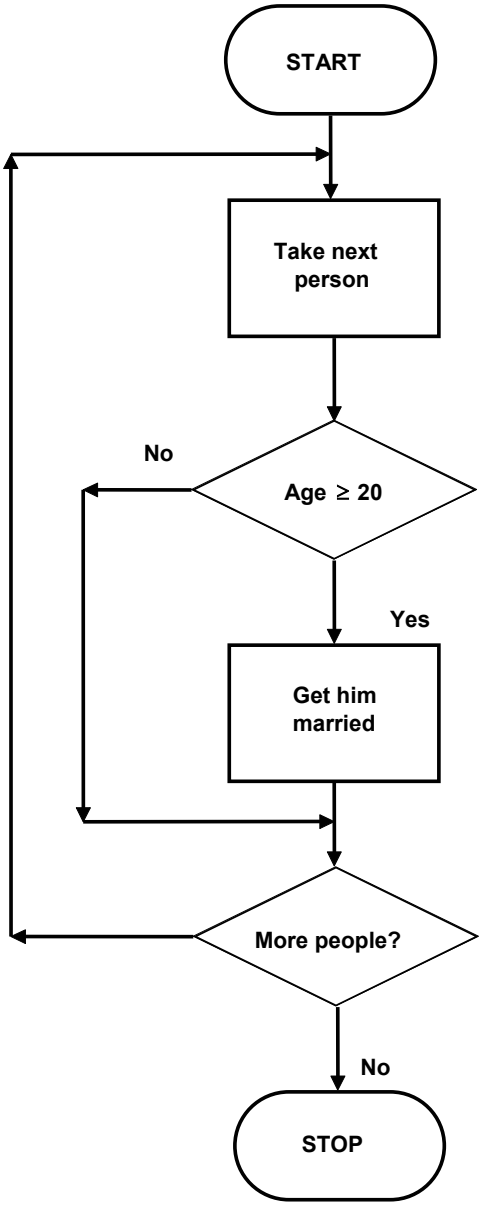
أما إن كان لا، فتوقف

ملحوظة:

ينبغي التنبيه هنا إلى أن عملية الانتقال من خطوة ٢ إلى الخطوة ٤ – عندما تكون الاجابة "لا" – لا تمثل دورانا أو تكرارا لأن عملية الدوران إنما تتم بالانتقال من خطوة متأخرة إلى خطوة متقدمة عدة مرات لإعادتها، ولذا فإن هناك دورانا بسيطا واحدا في هذا المثال، ويمثله العودة من خطوة ٤ إلى خطوة ١



شكل (1-12)



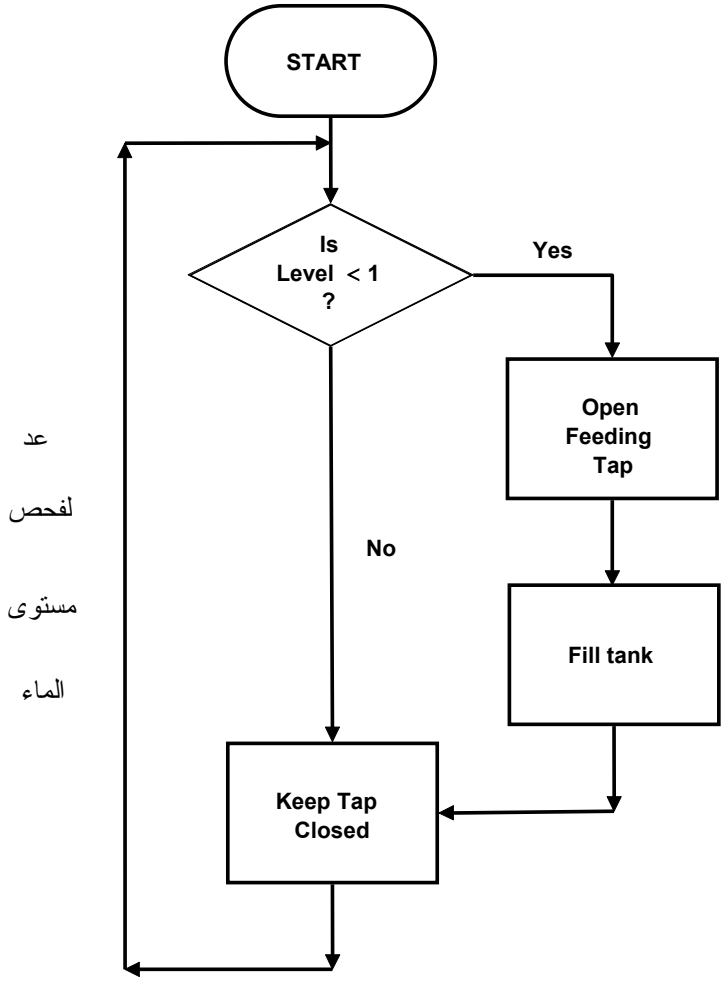
شكل (1-13)

مثال ٨

ارسم خريطة سير البرنامج لخزان يُملأ بالماء ذاتيًّاً (أتوماتيكيًّاً)، عندما يصبح ارتفاع مستوى 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 مستوى الماء في الخزان إلى الارتفاع المطلوب وبالتالي فإن خطوات الحل المبينة في الشكل (14-1) تكون كما يأتي:

1- هل مستوى الماء أقل من متر ؟ إذا كان الجواب نعم فاذهب إلى الخطوة (2)
وإذا كان الجواب لا ، فاذهب إلى الخطوة (4)
2- يفتح صنبور التغذية.
3- يملأ الخزان إلى المستوى المطلوب.
4- أغلق الصنبور (أو حافظ عليه مغلقاً).
5- عد إلى الخطوة (1) لفحص مستوى الماء مرة بعد مرة للحفاظ على الوضع المطلوب وبشكل دائم.



شكل (14-1)

مثال ٩

الشكل (15-1) يمثل خريطة سير البرنامج لمجموعة من العمليات الحسابية. ادرس العمليات بعد تتبع الخريطة.

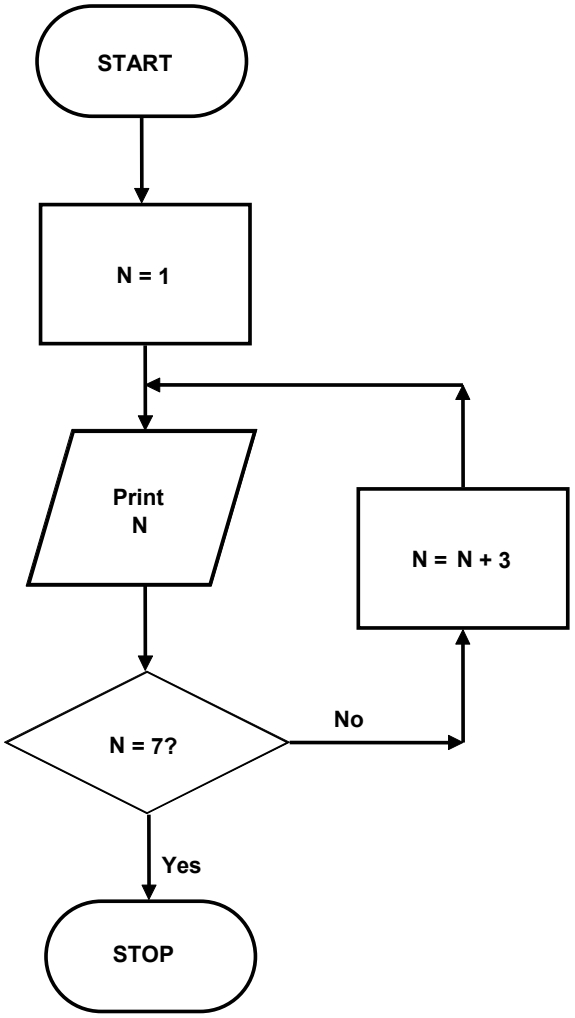
من الشكل نلاحظ أن الحاسب يبدأ بوضع قيمة مبدئية أولى مقدارها 1 في مخزن الذاكرة N كما في الشكل (16a-1)، ثم يقوم بطبع هذه القيمة، من خلال جهاز الإخراج، وعلى الوسط الخارجي ثم يسأل (هل القيمة المخزونة في مخزن N تساوي 7؟) الجواب بالطبع : لا، لأن 7 ≠ 1، لهذا فهو ينفذ الأمر التالي: N = N + 3 وهذا الأمر يعني أن الحاسب سيضع في مخزن N ما كان فيه سابقاً مضافاً إليه 3 لتصبح القيمة المخزونة في المخزن N تساوي 4 بدلاً من 1 (انظر الشكل (16b-1)) ثم يعود مرة أخرى بعد أن يطبع N الجديدة على الوسط الخارجي ليسأل هل 7 = 4 ؟ ويكرر العملية السابقة حتى تصبح القيمة المخزونة في المخزن N تساوي 7 وعندها يتوقف البرنامج بالأمر: توقف، ويكون قد طبع لنا على الوسط الخارجي القيم التي خزنت في مخزن N على التوالي وهي:

1

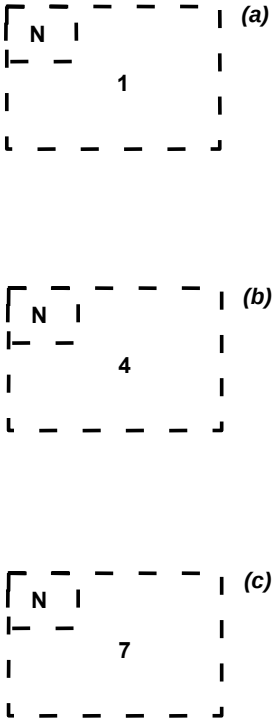
4

7

- ملحوظة: هناك مخزن واحد فقط تحت اسم N في وحدة الذاكرة تخزن فيه قيمة واحدة في الوقت الواحد، ولذا فإن آخر قيمة تبقى في المخزن N في المثال هي 7



شكل (1- 15)



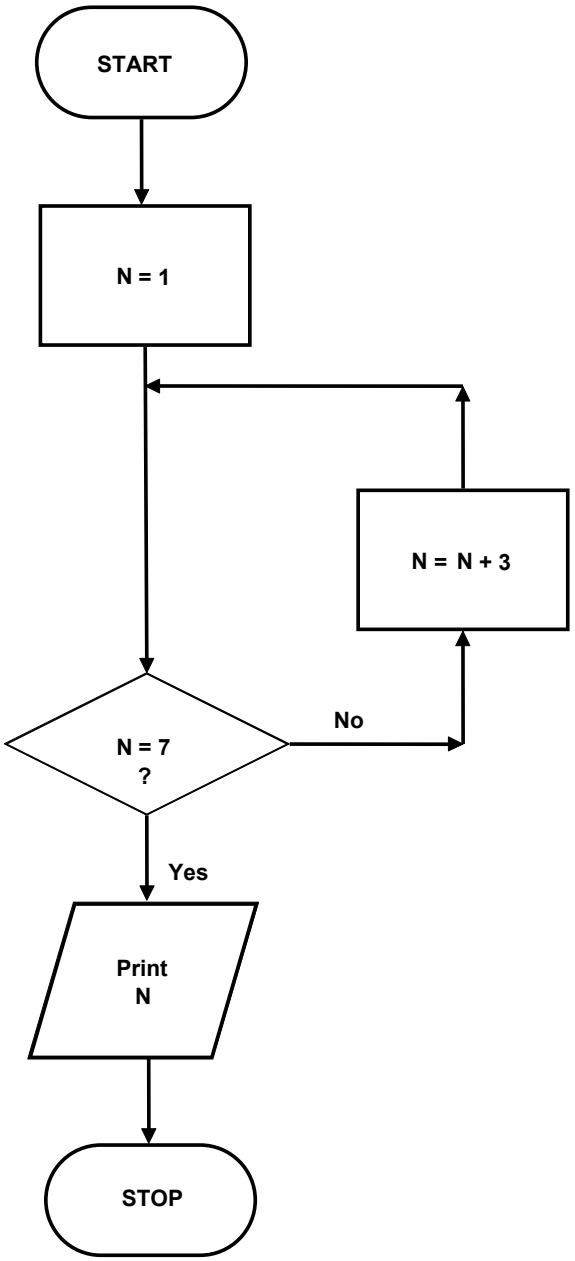
شكل (1- 16)

مثال ١٠

لو أحدثنا تغييراً بسيطاً في الشكل (15-1) ليصبح كما هو مبين في شكل (17-1) فما أثر ذلك التغيير؟

نلاحظ من الشكل (15-1) أن التغيير الذي حدث يتلخص في أن خطوة كتابة قيمة المتغير N قد تأخرت عن خطوة التقرير (هل N = 7)؟ وهذا يعني أن كتابة قيمة المتغير N تأتي بعد الانتهاء من الدوران أي بعد أن تصبح قيمة N تساوي 7، ولذا فإن نتائج الإخراج تكون قيمة واحدة فقط وهي: 7

في حين أن نتائج الإخراج في المثال السابق كانت تطبع في كل دوران، مما جعل النتائج في المثالين مختلفة بسبب التغير المذكور.



شكل (17 -1)

مثال ١١

ارسم خريطة سير البرنامج لإيجاد مجموع m من الأعداد الحقيقية

$(X_1, X_2, ..., X_m)$

الحل: النتيجة المطلوبة هي مجموعة الأعداد T حيث إن $T = \sum_{i=1}^{m} Xi$

خطوات الحل يمكن أن تسير على النحو التالي:

$$T_0 = 0$$
$$T_1 = T_0 + X_1 = 0 + X_1 = X_1$$
$$T_2 = T_1 + X_2 = X_1 + X_2$$
$$T_m = T_{m-1} + X_m = X_1 + X_2 + ... + X_{m-1} + X_m$$

ونموذج الحل هذا يمكن أن يختصر بنموذج مكافىء هو:

$$T_i = T_{i-1} + X_i \qquad (1)$$

حيث $T_0 = 0$ وتتغير i من 1 إلى m

ويمكننا اختصار عدد المتغيرات T_1 .... T_m بمتغير واحد هو T، الذي تكون قيمته في كل دوران مساوية لقيمته السابقة مضافاً إليها قيمة X المراد إضافتها إليه. وذلك بإعادة كتابة المعادلة (1) على النحو:

$$T_i = T + X_i \qquad i = 1, m \qquad (2)$$

على أن تكون القيمة الأولى للمجموع T تساوي صفراً.

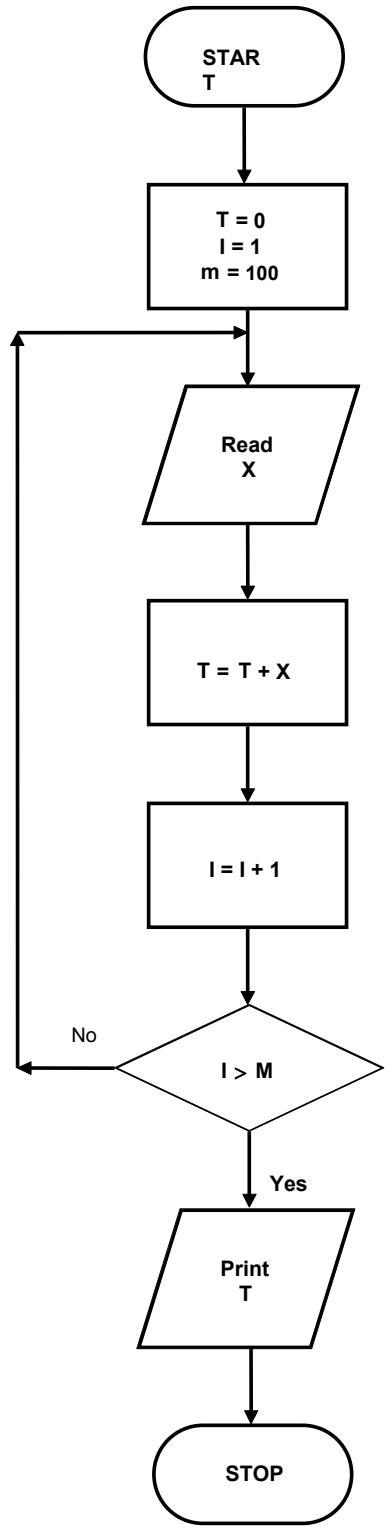
وترتيب النموذج في المعادلة (2) من شأنه أن يوفر أكبر عدد ممكن من المخازن الشاغرة في الذاكرة، لاستخدامها في أغراض أخرى، وكذلك فإن صيغة المعادلة (2) أسهل من صيغة المعادلة (1) وبالتالي فإنها تساعد على تسهيل عملية البرمجة.

أما خريطة سير البرنامج فمبيَّنة في شكل (18-1) (افرض أن قيمة m تساوي 100 هنا)

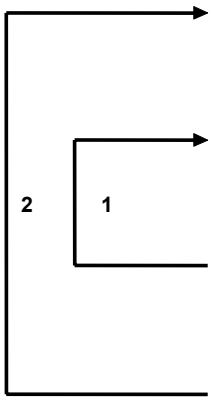
خرائط الدورانات المتعددة

في هذه الحالة تكون الدورانات داخل بعضها البعض بحيث لا تتقاطع فإذا كان لدينا دوران من هذا النوع (انظر شكل (1-19)) فيسمى 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 تسمى أحياناً بخرائط الدورانات المتداخلة أو المتراكبة أو الضمنية nested، وكل هذه التسمية تؤدي إلى معنى واحد.



شكل (1-18)



شكل(1-19)

مثال ١٢

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

خطوات الحل المبينة في شكل (20-1) هي:

1- خذ قطعة

2- اقطع منها قطعة طولها 5 متر

3- هل المتبقي يزيد عن 5 متر؟

إذا كان الجواب نعم، فاذهب إلى الخطوة (2)

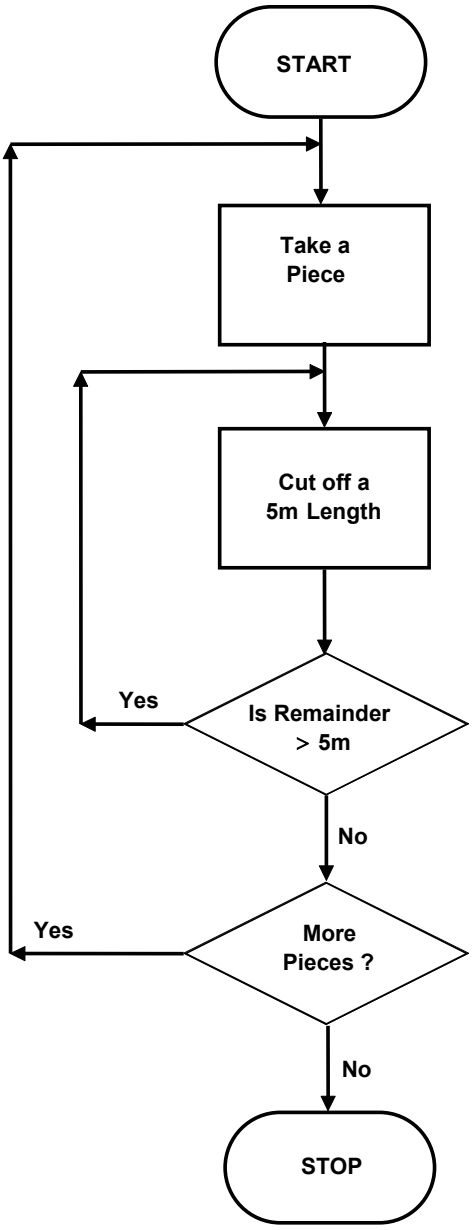
إذا كان الجواب لا، فاذهب إلى الخطوة (4)

4- هل هناك مزيد من القطع المراد تقطيعها؟

إن كان الجواب نعم، فاذهب للخطوة (1)

وإن كان لا، فتوقف

- **ملحوظة:** يلاحظ من الشكل (20-1) أن الدوران الداخلي يتضمن تقطيع القطعة الواحدة إلى قطع متعددة، طول كل منها 5 متر، بينما يمثل الدوران الداخلي تناول قطعة واحدة جديدة لتنفذ عليها إجراءات الدوران الداخلي.

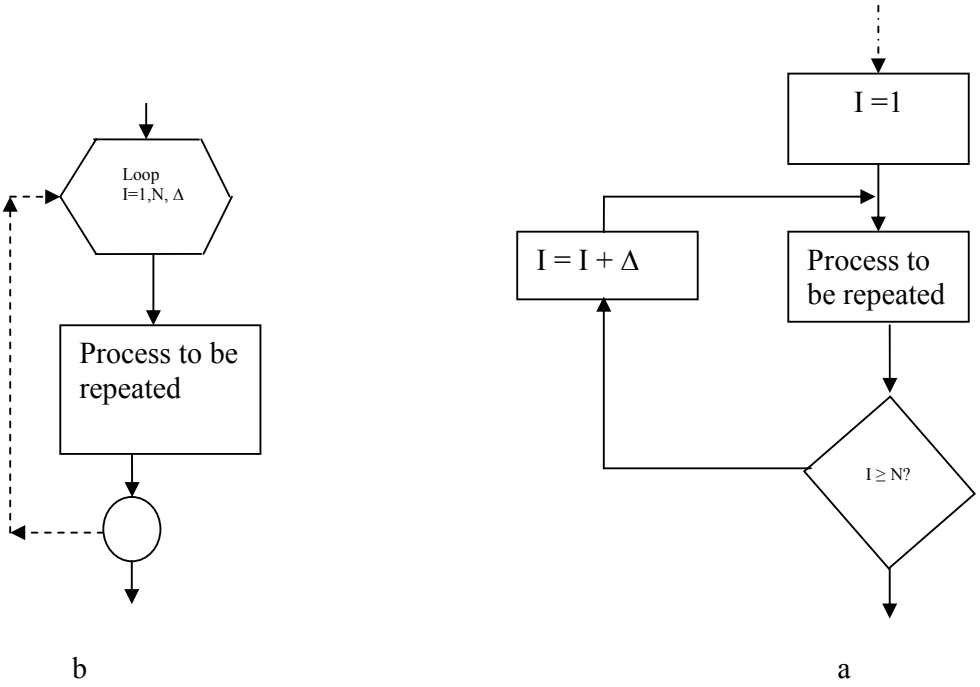


شكل (20)

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 (21-1) أننا نحتاج إلى العناصر الآتية:



a b

شكل (21-1)

العدّاد(I)

القيمة الأولية للعدّاد I (هنا i = 1)

القيمة النهائية للعدّاد I (هنا N)

قيمة الزيادة في العدّاد عند نهاية كل دورة (Δ)

نلاحظ من الشكل (a-21-1) أن إجراءات الدوران كانت تتم طبقاً للخطوات الآتية والمفصلة من قبل المبرمج:

1- أعط I قيمة أولية.

2- أتم الإجراءات المطلوب إعادتها.

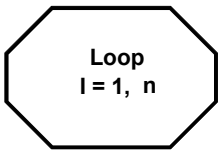
3- اتخاذ قرار: إذا كانت قيمة العداد I وصلت إلى القيمة النهائية N، فاخرج إلى الخطوة التالية في البرنامج وإلا فاذهب إلى الخطوة (4)

4- زد العداد بمقدار الزيادة Δ

5- عد إلى (2)

يمكننا استبدال الخطوات المفصلة (5,4,3,1) في الشكل (a-21-1) بخطوة مجملة واحدة مبينة في الشكل الاصطلاحي للدوران شكل (b-21-1)، حيث تنفذ هذه الخطوات بصورة أوتوماتيكية من قبل الحاسب. وهذا من شأنه تسهيل عملية البرمجة، واختصار عدد العمليات في البرنامج وتجنب بعض الأخطاء.

- ملحوظة: تعتبر قيمة Δ تساوي 1 دائماً إذا لم تُعطَ قيمة أخرى بخلاف ذلك، وفي حالة عدم ذكر قيمة Δ يصبح الشكل الاصطلاحي الوارد في شكل (b-21-1) كما هو موضح بشكل (22-1) وتكون قيمة الزيادة Δ تساوي 1، بصورة أتوماتيكية.



شكل (22 -1)

مثال ١٣

أعد حل مثال (6) لإيجاد مساحة من الدوائر باستخدام الشكل الاصطلاحي للدوران.

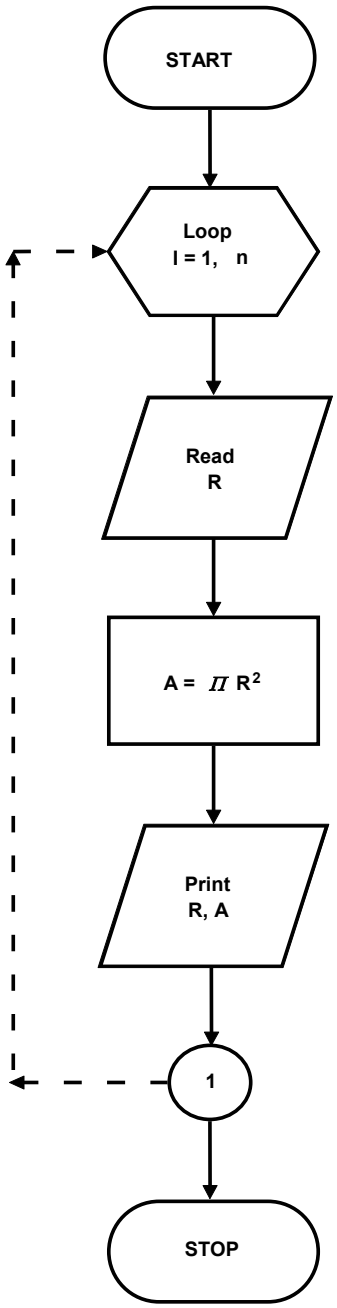
خطوات الحل كما مبينة في الشكل (23-1)

مثال ١٤

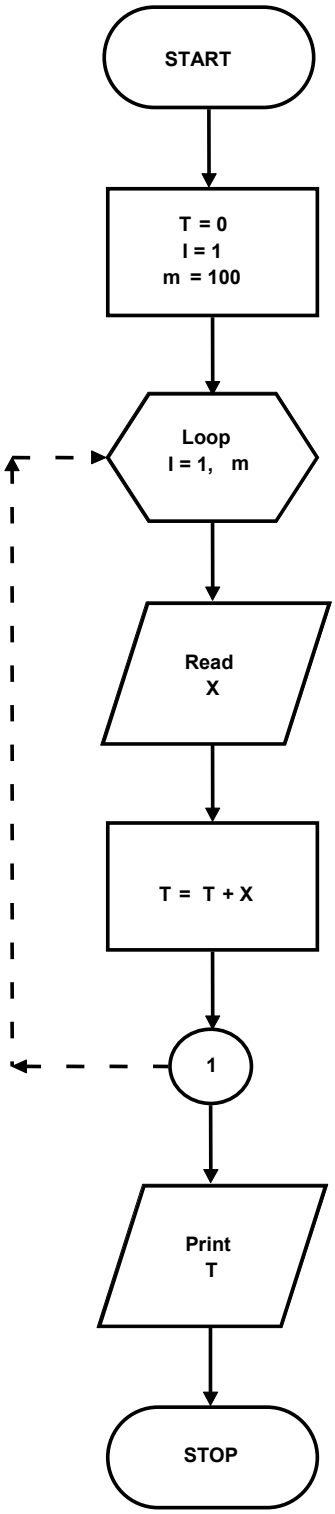
أعد حل مثال (11) باستخدام الشكل الاصطلاحي للدوران. بحيث

m = 100

خطوات الحل كما هي مبينة في الشكل (24-1)



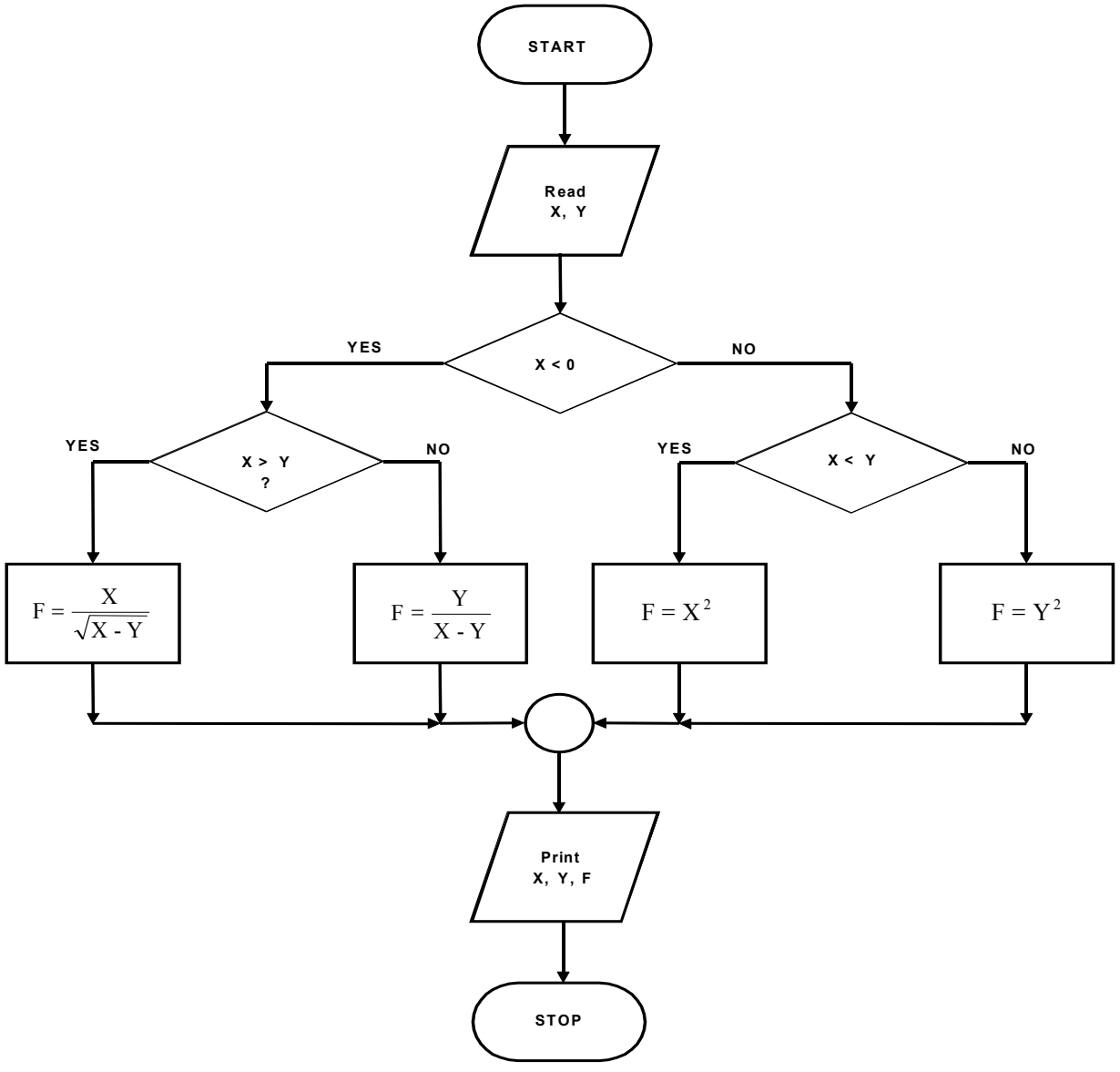
شكل (1-23)



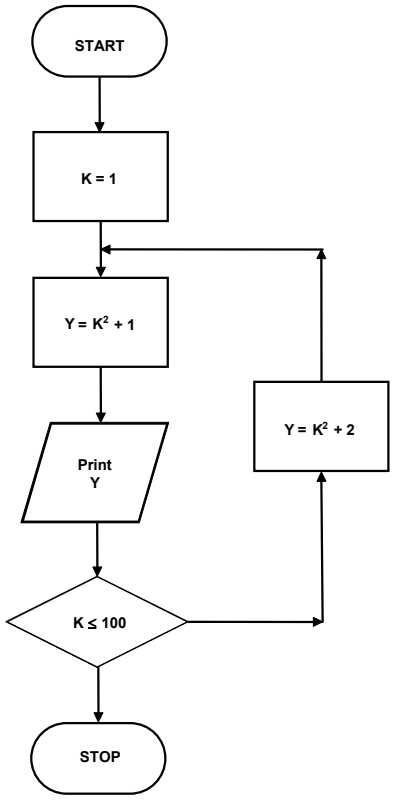
شكل (1-24)

تدريبات

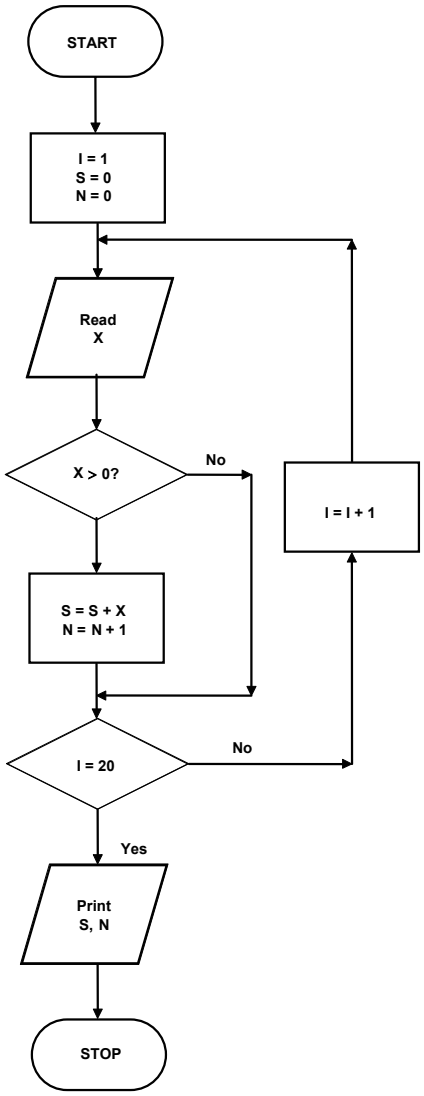
(١) ادرس المخططات (أ ،ب،ج) مبيناً أهداف كل مخطط والنواتج النهائية التي سيطبعها الحاسب عند تنفيذ التعليمات المبينة إزاء كل مخطط ، Y=5 , x=3 .



(أ)


START
K = 1
Y = K² + 1
Print
Y
K ≤ 100
Y = K² + 2
STOP

(ب)


START
I = 1
S = 0
N = 0
Read
X
X > 0?
No
I = I + 1
S = S + X
N = N + 1
I = 20
No
Yes
Print
S, N
STOP

(ج)

٢- ارسم خريطة سير البرنامج التي تمثل كلا من الخوارزميات التالية:

(أ)

1- ضع قيمة SUM صفراً، وقيمة N تساوي 1

2- اجمع N إلى SUM

3- إذا كانت N<6 فأضف 1 إلى قيمة N الحالية، ثم اذهب إلى الخطوة 1

4- اطبع قيمة SUM

(ب)

1- اقرأ قيمة X

2- إذا كانت $X \geq 0$ فاذهب إلى الخطوة (5)

3- احسب قيمة W من المعادلة $W = \sqrt{x^2 + 5x - 4}$ ، ثم اذهب إلى الخطوة (5)

4- احسب قيمة W من المعادلة: $W = -X + 13$

5- اطبع قيمتي X و W

3- ارسم خريطة سير البرنامج لحساب كلٍ من الاقترانات الآتية:

(أ) $f(X) = | X-3 |$

(ب) $SUM = \sum_{i=1}^{n} i$

(ج) $F = n! = n(n-1) \ldots (2)(1)$

(د) إيجاد قيمة أكبر عدد في المجموعة S حيث:

$S = [A, B, C]$

(هـ) إيجاد قيمة أصغر عدد في المجموعة S نفسها تنازلياً ثم تصاعدياً.

(ز) إيجاد قيمة أكبر عدد في السلسلة الحسابية:

$a_1, a_2, a_3, \ldots, a_{n-1}, a_n$

(ح) ارسم خريطة سير البرنامج لإيجاد قيمة أصغر عدد في المتسلسلة الحسابية في السؤال السابق.

(ط) ارسم خريطة سير البرنامج بحيث ترتب حدود المجموعة التالية ترتيباً تنازلياً:

$a_1b_1, a_2b_2, a_3b_3, \ldots, a_{n-1}b_{n-1}, a_nb_n$

(ى) ارسم خريطة سير عمليات لترتيب حدود المجموعة ترتيباً تصاعدياً.

(ل) اكتب أول 200 حد في المتوالية الهندسية التي تبدأ بالعدد 5، بحيث يكون معدل التغير 3.

(م) اكتب أول ثلاثين حداً في المتسلسلة التالية:

1, 3, 5, 7, ...

(ن) أوجد قيمة الاقتران عديد الحدود :poly حيث:

$$POLY = 1+Z+Z^2+ \dots +Z^{10}$$

إذا كانت قيمة المتغير z معروفة لديك.