المملكة العربية السعودية

جـامعــة نجران

حاسب آلي (٢)

(١٠٢ حال - ٣)

كلية العلوم الادارية

المستوى الثالث

الوحدة الأولى

مقدمة الى تركيب البيانات والملفات

Introduction to Data Structure

١- مقدمة الى تركيب البيانات والملفات

تتطلب عمليات البرمجة ومعالجة البيانات خوارزميات فعالة للتعامل مع البيانات في كل من الذاكرة الثانوية والرئيسة. وترتبط هذه الفعالية ارتباطاً مباشراً مع تركيب هذه البيانات المعالجة.

مثال: لو قلنا العلامة "A" ماذا يعني ذلك؟ طبعاً ليس له معنى في حين لو قلنا العلامة "A" للمساق بيسك (١) وهي علامة الطالب Ahmad Hasan فان لها معنى كبيراً. ويُعد هذا طريقة لتنظيم البيانات.

تعريف تركيب البيانات: هي طريقة لتنظيم البيانات التي لا تتعامل فقط مع المفردات وانما مع علاقة المفردات مع بعضها.

مثال: على تركيب البيانات (تعريف تركيب البيانات) يوضحه الجزء التالي من برنامج باسكال:

```
TYPE    INTARRAY = ARRAY [1..30] OF INTEGER;
  :       .
  .       .
          .

VAR     N: INTARRAY;
  :       .
  :       .
          .

FOR  I: = 1 TO 30 DO
READ (M[I]);
```

نقول بأن التركيب N يحتوي 30 عنصراً، كل منها عدد صحيح (Integer). والعنصر الأول هو N[1] والعنصر العاشر هو N[10]. وهذا الجزء من برنامج باسكال المعطى سابقاً سيقرأ الارقام من المدخلات ويخزنها بالتتابع في N[1]، N[2]، N[30]، واماكن التخزين متماسكه. وبالتالي فان طريقة التعامل مع هذه البيانات الموجودة في N تعتمد على تنظيم هذه البيانات.

والسجل هو نوع شائع من تركيب البيانات كما في المثال التالي:

```
STUDENTREC = REC
Name: ARRAY [1..30] of char;
TERM: integer;
TEST: integer;
Grade: char;
End;
```

حيث يحتوي سجل الطالب على الاسم الذي يتكون من منظومه 1..30 (Array) من الرموز ويحتوي على الفصل وهو عدد صحيح وكذلك الامتحان عدد صحيح واخيراً يحتوي على العلامة النهائية وهي بالرموز.

٢- التركيب المنطقي والتركيب الفيزيائي

أ- التركيب المنطقي:

وهو وجهة نظر المبرمج في سير البرنامج اي ان هذا ترتيب معلومات البرنامج بشكل معين حتى يتم تنفيذ هذا البرنامج بطريقة صحيحة.

ب- التركيب الفيزيائي:

وهو يعني كيفية ترتيب البيانات على اوساط التخزين مثل الشريط المغناطيسي والقرص المغناطيسي، حيث يتم تخزين البيانات على الشريط

المغناطيسي بطريقة تتابعية، اما تخزين البيانات على القرص المغناطيسي يتم بطرق أخرى مثل الطريقة المباشرة او التتابعية المفهرسة.

٣- المصفوفات: (ARRAYS)

المصفوفة ذات البعد الواحد (One dimensional array)

وتسمى ايضاً شعاعاً أو متجهاً (Vector) ويرجع الى وجود عدد محدود من المواقع في الذاكرة وتكون متتابعة. وهذا العدد هو حجم الشعاع أو المتجه. وبالتالي فان المصفوفة ذات البعد الواحد تمثل بالنسبة للكمبيوتر تمثل صفاً متراصاً من الناس. لاحظ انه في المصفوفة ذات البعد الواحد لدينا القدرة على التعامل المباشر مع اي عنصر (بمفرده) وذلك بتحديد موقعه فقط.

والمصفوفة ذات البعدين او الثلاثة ابعاد يمكن ان تعامل على اساس انها ذات بعد واحد.

والمثال التالي يوضح استخدام المصفوفات أو الجداول:

```
VAR A : Array [1..100] of String;
```

وهي طريقة لتمثيل 100 اسم، فإذا اردنا مثلاً فحص الاسم رقم 40 فلا يوجد داع لفحص الاسماء من 1-39 بل يمكن فحص الاسم 40 مباشرة.

- **القوائم والقوائم المتصلة:** Lists & Linked Lists

القوائم المتصلة (Linked Lists)

يرد هذا التركيب في اكثر عمليات المعالجة ويدعى القوائم المرتبة (Ordered Lists)

مثال: قائمة من الاسماء مرتبة حسب الاحرف الابجدية. او قوائم مرتبة حسب ايام السنة. مثل هذه البيانات يمكن تمثيلها باستعمال المصفوفة أو الجدول ذي البعد الواحد.

وعندما تحتوي هذه القوائم على مدخلات كبيرة جداً فان تطوير وسائل فعالة لعملية التحديث (update) تكون ذات اهتمام كبير.

وبشكل عام نجد ان معالجة القوائم المرتبه تتضمن:

١- التعامل مع العنصر J من القائمة، حيث اذا كان n طول القائمة فان $1 \le J \le n$ (اي ان J محصورة بين 1-n).

٢- ادخال عنصر جديد بين المواقع J و J+1 في القائمة بحيث يبقى الترتيب صحيحاً.

٣- حذف العنصر J مع ابقاء الترتيب صحيحاً في القائمة.

فمثلاً: اذا كانت لدينا قائمة مرتبة حسب رقم الطالب كما في الشكل:

(1121, 1125, 1172, 1180, 1195)

واردنا ادخال الطالب ذي الرقم 1175

وبنفس الوقت اردنا حذف الطالب ذي الرقم 1125 مع الابقاء على الترتيب.

نلاحظ ان الرقم 1175 المراد ادخاله الى القائم يدخل بين الرقمين 1172 و 1180 لان الابقاء على الترتيب هو المطلوب فتصبح القائمة بعد عملية الحذف والاضافة:

(1121, 1172, 1175, 1180, 1195)

لو كان لدينا قائمة رئيسة (Master List) للموظفين مرتبه ترتيباً تنازلياً حسب رقم الضمان. وهذه القائمة تم تحديثها بواسطة الحذف والاضافة فان المصفوفات (Arrays) تصبح عملية غير فعالة (وذلك بسبب عمليات التحريك التي تتم لهذه البيانات).

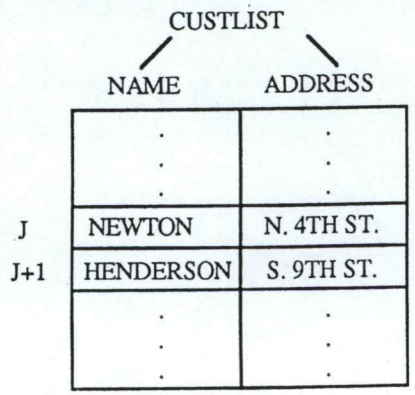
طريقة اخرى: وهي طريقة القوائم المتصلة (Linked Lists): وهذه ليست الا تركيباً يؤدي الى خوارزميات تقلل من تحريك البيانات حينما تتم عملية ادخال او حذف بيانات من القوائم المرتبة.

في القوائم المتصلة يسمى كل عنصر عقدة (NODE) وهذه العقدة تحتوي على حقلين الاول يسمى حقل البيانات (data field) والثاني مؤشر (pointer) يشير حسب عنوان معين الى العقدة الاخرى.

ويغني استخدام القوائم المتصلة عن عملية تحريك بيانات الذاكرة وما يلزم هو الاحتفاظ فقط بموقع العقدة وتتم عملية تحديث للمؤشرات.

المؤشرات (POINTERS):

```
TYPE
CUSTREC = RECORD
    NAME: PACKED ARRAY [1.. MAXNAMELEN] OF CHAR;
    ADDRESS: PACKED ARRAY [1.. MAXADDRESSLEN] OF CHAR END;
```



واذا كنا مهتمين فقط بانتاج قائمة مرتبة حسب الاحرف للاسماء والعناوين للشكل السابق فهل من الضروري ترتيب هذه القائمة فيزيائياً على ذاكرة الكمبيوتر؟ الجواب لا. قد يكون ذلك محتملاً بطريقة منطقية بدون الحاجة الى الترتيب الفيزيائي على الذاكرة ولتحقيق ذلك يجب ان نستخدم حزمه من المؤشرات– (Array of pointers).

وهذه المؤشرات تحتفظ بالترتيب المنطقي للبيانات دون الحاجة الى نقلها فيزيائياً وفي النهاية فان المؤشر الأول يبين موقع الاسم والعنوان الواجب ان يكون الاول في الترتيب، والمؤشر الثاني يحتوي على موقع السجل الذي يكون الثاني في الترتيب وهكذا ...

- **التركيب الشجري (الهرمي) للبيانات** (Trees)

شجرة العائلة تبين علاقة الاقارب، الابوين وغيرهم.... ومثال هو علاقة الطفل بابويه Parent-child والتي تمثل علاقة هرمية (hierarchical) حيث ان علاقة السبق واضحه بين هذه العناصر. وفي الحاسوب فان الشجرة (tree) هي عبارة عن تركيب بيانات تمثل علاقة هرمية بين المفردات.

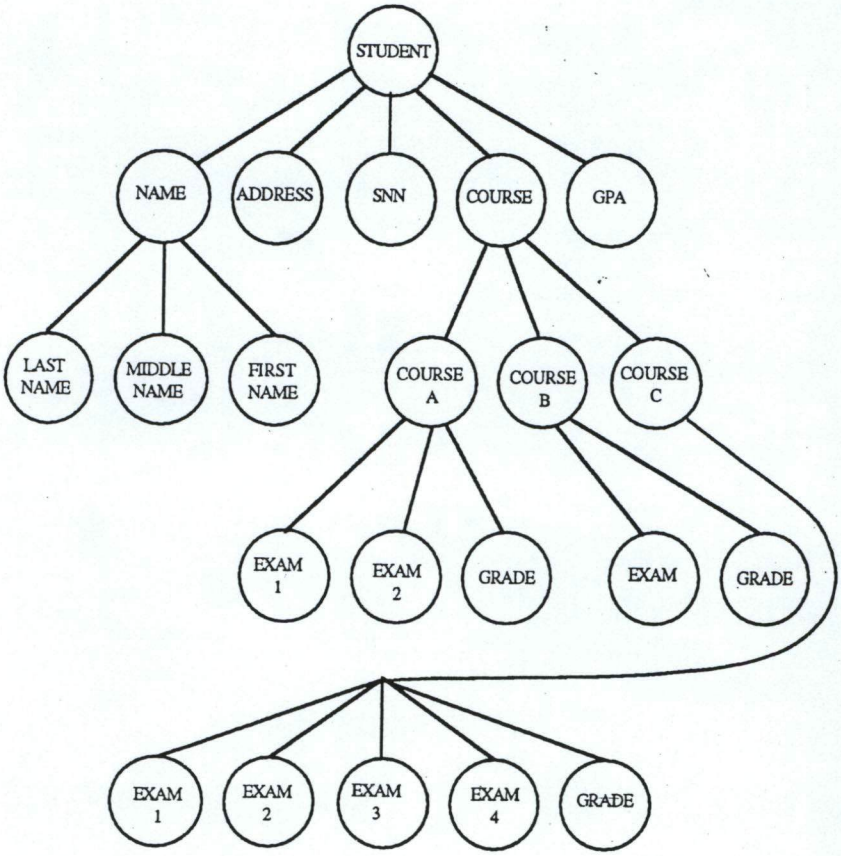
مثال: سجل الطالب في الجامعة يحتوي على:

(SSN Social Security Number) رقم الضمان الاجتماعي، (name) الاسم، (address) العنوان، (Number of courses) عدد الفصول، (Final Grades) معدل كل فصل، وكذلك (Grade) المعدل العام.

والشكل 1-1 يبين هيكل الشجرة (Tree Structure) للطالب حيث المستوى الاعلى في الشجرة يكون في (Level 0) أي المستوى صفر ويسمى (root) أي الجذر، وكذلك يسمى عقدة الجذر او عقدة الاصل (root node) وعقدة الجذر في الشكل هي الطالب (STUDENT) والعقد الاخرى في الشجرة هي address, SSN, course ، Name وكذلك GPA والمرتبطة مباشرة مع الطالب التي تشكل عقدة الاصل تدعى العقد الحفيدة (child nodes) لما يسمى بالعقدة الاصلية parent node لشجرة الطالب. والعقدة الوليدة او الحفيدة (child nodes) تدعى مجموعة الاخوة (Siblings) لذلك فان العقد GPA، SSN, address, Name course تسمى الاخوات siblings. لذلك تكون هذه العقد في المستوى الاول Level 1 والربط بين الاصيل والحفيد يسمى التفريع branch.

والجذر للشجرة هو الاصل (Ancestor) لكل العُقَد. والعقدة التي لا يوجد لها تفرعات (children) تسمى عقدة الورقة (leaf node).

وفي الشكل 1-1 فان GPA تسمى عقدة ورقة وكذلك EXAM 1, 2, ... تسمى عقد ورقية.



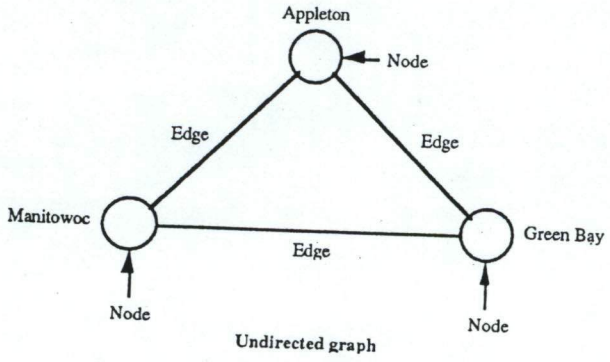
شكل (١-١)

- **المخططات (Graphs)**

تُعد الشجرة (tree) نوعاً خاصاً من المخططات حيث تكون العلاقة بين الاصيل (parent) والابن (child) علاقة هرمية.

والمخطط العام يمثل علاقة ذات تحديدات اقل بين مفردات البيانات (data Items). والمثال الشائع للمخططات هو تجمع من المدن والطرق التي تربط بينها. وفي هذا المثال فان اي مدينتين تكونان مرتبطتين اذا كان بينهما طريق وهذه العلاقة ليست هرمية لان الطريق بين المدينتين يكون باتجاهين.

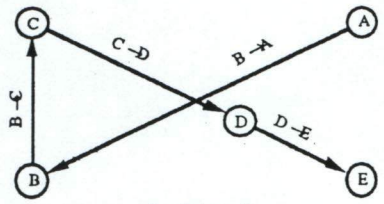
تعريف المخططات (Graph): هي مجموعة من العناصر تدعى العُقَد (nodes) والحواف (edges) وبالتالي فان العقدة (Node) عبارة عن عنصر من المخطط بينما الحافة (edge) هي طريق (path) بين عقدتين من المخطط كما في الشكل ادناه.



Undirected graph

تعريف المخطط غير الاتجاهي (undirected graph): اذا كان AG من A الى G هو حافة فان GA من G الى A هو ايضاً حافة وهكذا...

في حين يُعرّف المخطط الاتجاهي (directed graph) بأنه المخطط الذي يكون فيه الاتجاه موجهاً ولا يجوز عكسه (حافة باتجاه واحد)، مثل الطريق ذي الاتجاه الواحد. كما في الشكل ادناه.



Directed graph

C ⟶ D هو حافة

ولكن D ⟶ C ليست حافة

٤- الخيط أو السلسلة (String):

سلسلة من الرموز موضوعة في منطقة متماسكه منطقياً ويمكن ان تكون بأي حجم والخيط الذي لا يحتوي على رموز يسمى خيط الرموز الفارغ (empty string أو null string) ويكون طولها صفري. وبالتالي فان طول الخيط هو عدد الرموز التي تكون داخله.

خيوط الرموز أو سلاسل الرموز (Character Strings):

تلعب معالجة خيوط الرموز دوراً مهماً في كثير من تطبيقات الحاسوب، مثل عملية التفاعل بين الحاسوب والمستخدم باللغة العادية كما في لغة كوبول ولكن في لغات اخرى مثل فورتران أو باسكال يجب ان يكتب المستخدم ذلك بشكل اجراءات (procedures) أو اقترانات (Functions).

العمليات المسموح بها على خيوط الرموز:

١- عملية النسخ (Assignment) وهي نسخ متغير رمزي في متغير رمزي آخر. فاذا كان STR1، STR2 متغيرين رمزيين فان العملية

STR1: = STR2 تنسخ محتويات STR2 في مخزن STR1 في ذاكرة دون اضاعة محتويات STR2.

٢- عمليات البناء والوصل (Concatination) وهي بناء خيوط من الرموز وتعني عملية الربط والوصل لخيطين من الرموز معاً. وعملية البناء من الرموز التالية ROOM وكذلك MATE تعطي الناتج ROOMMATE.

٣- عمليات مطابقة الخيوط أو السلاسل(pattern matching) : وتفيد في التعديل والبحث: وتعني هنا البحث في خيط معين عن وجود خيط آخر. فمثلاً خيط رموز مثل JERUSALEM يمكن ان يوجد في السطر (الذي يمثل خيط رموز آخر). وهو

MODERN JERUSALEM HISTORY

ويبدأ في الموقع الثامن من العبارة الأخيرة

(٤-١) (Insertion): الاضافة بالادخال:

يمكن في عملية الاضافة ادخال خيط رموز في منتصف خيط رموز آخر. مثال: عملية ادخال AND في الموقع 7 من خيط الرموز BASIC PASCAL ينتج خيط الرموز التالي:

BASIC AND PASCAL

(٥-١) (Deletion): الالغاء:

وهو عكس الاضافة وبه يتم مسح تعبير خيطي. مثال: حذف التعبير الخيطي الموجود في المثال السابق من الموقع 7 الى الموقع 15 ينتج الخيط BASIC فقط.

٥- الحزَم والطوابير STACKS AND QUEUES

قمنا سابقاً بشرح القوائم المتصلة التي ليست الا تركيب بيانات صمم خصيصاً لمعالجة عمليات الحذف والاضافة في القوائم المرتبة. وسنعرض هنا نوعين خاصين من القوائم التي تتميز بتجديدات في طريقة ادخال (اضافة) وطريقة حذف المدخلات.

١- **الطوابير** (queues) وهي قائمة على اساس الداخل أولاً هو الخارج أولاً (FIRST-IN-FIRST-OUT: FIFO) حيث الادخال محدد فقط بطرف من القائمة ولكن الحذف يتم في النهاية. (اي يوجد للقائمة نهايتان واحدة للحذف والاخرى للاضافة). وهاتان النهايتان لهما اسماء هي المقدمة (FRONT) وهي للحذف والمؤخرة (REAR) وهي للاضافة. وشكل الطابور هو شكل خط الانتظار، كما في حالة الاعمال التي تنتظر الخدمة من الحاسوب.

مثال آخر هو معالجة الحُزَم (batch processing).

٢- **الحزم** (Stacks) وهذا التركيب قائم على اساس آخر الداخلين هو أول الخارجين. (LAST-IN-FIRST-OUT: LIFO) حيث تتم الاضافة والحذف تتم من طرف واحد فقط، لذلك يكون لها مؤشر واحد في اعلى (top) القائمة.

وهذه الحزم تستخدم في الحياة العملية أيضاً، فمثلاً (ترتيب الامتعة في الحقيبة) حيث يكون آخر ما نضعه في الحقيبه عند ملئها هو أول ما يخرج منها عندتفريغها.

وتسمى عملية الحذف من الاعلى (Popping the stack) أي الاخذ من أعلى الحزمة.

وعملية الاضافة على الحزمة تسمى (Pushing the stack) الدفع الى الحزمة.

٦- العلاقة بين الزمن وحجم مكان التخزين

يختار المبرمجون عادة بين كتابة البرامج سريعة التنفيذ والتي تستخدم كثيراً من الذاكرة وبين البرامج البطيئة التي تستخدم قليلاً من الذاكرة، هذا الاختيار ليس دائماً مقصوداً. وقليلاً ما تحدث على اساس انه يتجه لاختيار تطبيق معين، وعلى سبيل المثال اذا كان لدينا قائمة مرتبة ابجدياً لمستخدمي الحاسوب وكذلك ارقام حساباتهم، نستطيع بسهولة ان نأخذ المعلومات عن طريق الرقم، واذا اردنا معرفة اذا كان شخص موجوداً في القائمة ام لا (لنفترض ان الشخص غير موجود) فليس بالضرورة ان نقرأ القائمة كلها وذلك لانها مرتبة ابجدياً. وبالرغم من ذلك اذا عرفنا رقم الحساب لأحد الاشخاص ونريد ان نعرف من هو، فيجب ان نبحث القائمة كلها. ويكمن حل هذه المشكلة في وجود قائمتين، قائمة مرتبة حسب الاحرف وقائمة مرتبه حسب الرقم. لذلك فانه من السهل الحصول على المعلومات، غير أننا قمنا هنا بمضاعفة احتياجاتنا من التخزين. اذا كانت الزيادة فقط من 200-100 موقع، فان الزيادة في حجم الذاكرة المطلوب يمكن ان لا يكون له تأثير يذكر او قد يكون مهملاً. ولكن اذا كانت الزيادة في حجم الذاكرة من 10,000-20,000، فيجب ان نضع بعين الاعتبار فيما اذا كنا نحتاج الى ذاكرة كبيرة (اي نضحي بالذاكرة) مقابل ان نحصل على المعلومات بسرعة. طريقة أخرى لتوضيح ذلك وهي اننا بحاجة الى قائمتين-قائمة مرتبه حسب الاحرف الأبجدية، واخرى مرتبة حسب الارقام-وهي استخدام عملية الربط (LINKS) كما هو موضح في الشكل:

	NAME	USER number	LINK
1	BAKER J	201	1
2	GRAIGE	167	1
3	FRAMER M	121	2
4	JONES A	348	-
5	WALKER T	313	4

3
FIRST

هذا الشكل يبين قائمة الترتيب حسب رقم المستخدم

القائمة مرتبة ابجدياً، والروابط (LINKS) تبين موقع المفردة التالية في القائمة اذا كانت مرتبة حسب الرقم.

المتغير FIRST يبين موقع رقم اول مستخدم في القائمة. وبما ان FIRST = 3، فيكون رقم المستخدم في القائمة هو الاصغر.

الربط للرقم 121 هو 2، والتي تبين ان الرقم 167 (الموقع 2 في القائمة) هو رقم المستخدم التالي. ويتبع 167 الرقم 201 (في الموقع 1)، ثم 313 (في الموقع 5) واخيراً 348 (في الموقع 4). لاحظ أن الرابط في 348 عبارة عن شرطة (-) وهي تبين رقم اخر مستخدم. وفي الحاسوب يمكن ان نستخدم 0، أو رقماً سالباً، او رموزاً خاصة لتبين نهاية القائمة. وهذا المثال يبين ويؤكد العلاقة بين الزمن وحجم مكان التخزين، وتأخذ الروابط مساحة غير أنها توفر لنا تعاملاً مع القائمة بطرق مختلفة.

وتقع عمليات الاضافة والحذف (اضافة وحذف عناصر) من مجموعة من البيانات ضمن عملية التحديث (updating). والتحديث كذلك يتضمن تصحيح العناصر الموجودة، وحينما نريد اضافة عنصر، يجب ان نبين فيما اذا كان هنالك

محتويات الوحدة الثانية

الوحدة الثانية

القوائم والجداول (المصفوفات)

Lists & Arrays

القوائم (Lists) والجداول (المصفوفات) (Arrays)

١- مقدمة وتعريف القوائم والجداول: شرحت سابقاً.

٢- الجداول: شرحنا في الدروس السابقة عن الجداول ذات البعد الواحد وقد اعطيت اسم الشعاع أو المتّجه (Vector).

- **المصفوفات أو الجداول ذوات البعدين (Two dimensional Array):**

وهي قائمة متماسكه من الاشعة (المتجهات) (Vectors) وتسمى مصفوفة (matrix) ذات بعدين N, M حيث تمثل N عدد الاسطر وتمثل M عدد عناصر كل سطر أو عدد الاعمدة.

- **الجداول ذوات الثلاثة ابعاد (3-dimensional Array):**

وهي ليست الا مجموعة من الجداول ذوات البعدين وتتميز بالاسطر والاعمدة والرتبة (row, column, rank) . ونستطيع القول منطقياً بأن الجدول ذا البعدين يمكن تمثيله بمثلث مكون من صفوف واعمدة.

فمثلاً جدول ٢×٣ (2x3 array) يمكن تمثيله على النحو التالي:

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)

ويخزن الجدول أو المصفوفة فيزيائياً بطريقة تتابعية. وهناك طريقتان لتخزين الجدول:

أ- على اساس الاعمدة (by column) أو حسب رتبة العمود.

ب- على اساس الاسطر (by rows) أو حسب رتبة السطر.

والاشكال التالية توضح عملية التخزين لمصفوفة أبعادها 2x3.

(1,1)
(1,2)
(1,3)
(2,1)
(2,2)
(2,3)

ترتيب حسب السطور

by row

(1,1)
(2,1)
(1,2)
(2,2)
(1,3)
(2,3)

ترتيب حسب الاعمدة

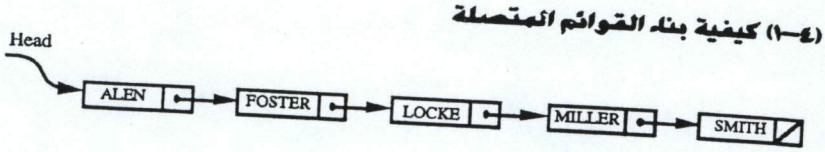
by column

٣- القوائم المتصلة (Linked Lists):

شرحت سابقاً.

٤- العمليات المطبقة على القوائم المتصلة

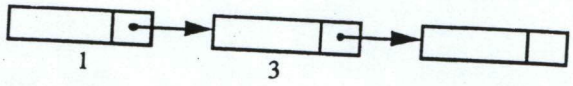
(٤-١) كيفية بناء القوائم المتصلة



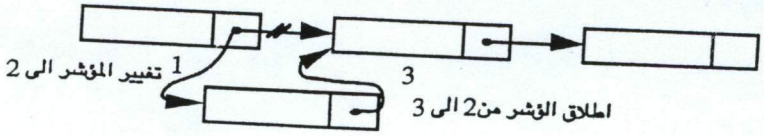
وهي تبين التمثيل المنطقي للقائمة المتصلة

(٤-٢) اضافة والغاء عناصر الى القوائم

الشكل التالي يبين قائمة متصلة



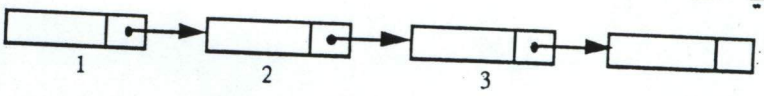
إذا أردنا اضافة عقدة 2 بين 1,3 فما يحصل هو تغيير اتجاه المؤشر (POINTER) من 1 الى 2 وتوجيه مؤشر العقدة رقم 2 ليؤشر على العقدة 3.



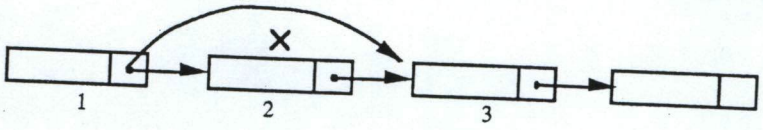
عملية الاضافة

وسوف نتطرق الى عمليات الحذف والاضافة بالتفصيل.

عملية الحذف



إذا أردنا حذف العقدة 2 فكل ما نقوم به هو جعل المؤشر في العقدة 1 مساوياً للمؤشر في عقدة 2 (اي يصبح مؤشر العقدة الاولى في العقدة 3) كما في الشكل.



عملية الحذف

٥- انواع القوائم المتصلة

(٥-١) القوائم المتصلة المفردة والعمليات المطبقة عليها

(Singly Linked Lists)

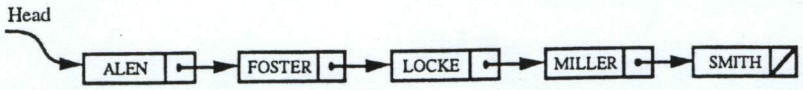
القائمة المتصلة المفردة عبارة عن قائمة متصلة حيث تحتوي كل عقدة (Node) على حقل اتصالي (Link field) واحد يشير الى العقدة التالية التي تليها في القائمة. والعقدة الاولى يشار اليها بمؤشر يدعى الرأس (Head) والعقدة الاخيرة في القائمة لها حقل اتصال (Link field) يحتوي على العلامة الصفرية (Null flag) وتبين نهاية القائمة (end of list).

ويبيّن الشكل التالي التمثيل الفيزيائي للقائمة المتصلة

	DATA	LINK
1	FOSTER	5
2	SMITH	NULL
3	MILLER	2
4	ALLEN	1
5	LOCKE	3

HEAD
4

حيث يتبين لنا من التركيب الفيزيائي الترتيب المنطقي على اساس معرفة الرأس (Head) ومعرفة علاقة النهاية Null . فيكون التمثيل المنطقي للشكل الماضي على النحو التالي:



حيث تم ترتيب البيانات في التمثيل المنطقي حسب الاحرف الابجدية لان القائمة يجب ان تكون مرتبة.

ويبيّن التمثيل الفيزيائي للقائمة المتصلة (كما هو موضح في الشكل ٢-١) جدول من سجلات بلغة باسكال وبالاعتماد على اللغة المستعملة فان الذاكرة المستعملة لهذه السجلات يجب ان تعطى في وقت الاتمام (Compilation time) ويسمى التوزيع الساكن للذاكرة (Static allocation) أو وقت التنفيذ (excution time) ويسمى التوزيع الحركي للذاكرة (dynamic allocation) كما في لغة باسكال.

سوف نبين المعالجة في قائمة متصلة مفرده من خلال فحص المثال التالي لايجاد معدل المفردات العددية المخزنه في قائمة متصلة وذلك من خلال اجراء.

الوحدة الثالثة

الحـــزم والطـوابيـر

Stacks & Queues

الحزم والطوابير Stacks and Queues

١- تعريف ومقدمة الى الجزم والطوابير (امثلة عملية تطبيقية للحزم والطوابيز وكذلك المؤشرات في الطوابير).

الطوابير (Queues):

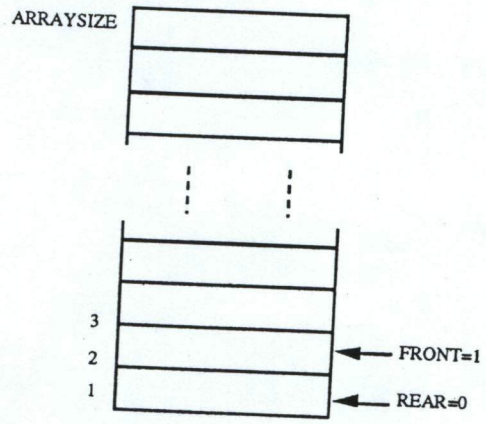
شرحنا في الدروس السابقة عن الطوابير وقلنا انها تحتوي على مؤشرين، مؤشر لاضافة العناصر ومؤشر آخر لالغاء هذه العناصر وقد بينا امثلة عملية على ذلك.

من الامثلة العملية هو اعمال الحاسوب التي تجدول في حزم المعالجة (batch processing) ، ونفترض ان اسماء الاعمال تتكون من ستة احرف او اقل وتتم جدولة هذه الاعمال حسب ترتيب وصولها. وبالتالي فاننا سنستخدم جدولاً ومؤشرين لتطبيق جدولة الطابور كما في الجزء التالي من برنامج باسكال التالي:

```
TYPE DATASTRING = Packed Array [1..6] of char;
   Queue Array = Array [1.. Array size] of Integer;
Var
   FRONT, REAR: INTEGER;
   QUEUE: Queue array;
```

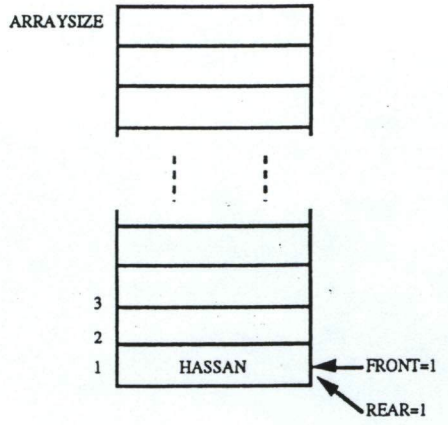
وهنا فان حجم الجدول (Array size) سوف يوضع في ثابت (CONST) ليحدد اكبر رقم للمدخلات التي يمكن ان يحتويها الطابور. ونجد هنا ان الطابور ممثل بالجدول. وسوف نرى ان هذا الرقم يختلف عن اقصى عدد للمدخلات التي يمكن ان يحتويها الطابور في وقت محدد من المعالجة.

اذا وضعنا في البداية قيمة المؤشر الامامي = (FRONT = 1)، والمؤشر الخلفي = صفر (REAR = 0)، نجد في البداية ان حالة الطابور (قبل اي عملية اضافة او حذف) ستكون كما في الشكل (٢-٧).



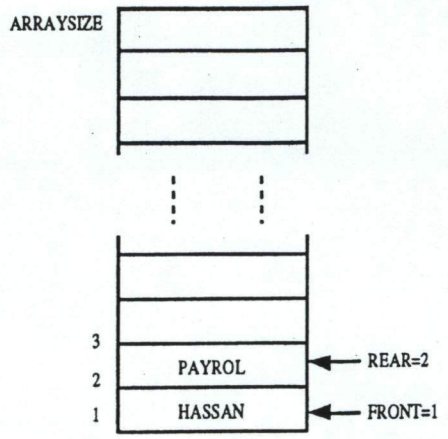
شكل ٢-٧

ونلاحظ ان عملية الاضافة في الطوابير تتم من خلال المؤشر REAR لنفرض أننا نريد ادخال عمل معين باسم Hassan والتي وصلت الان للمعالج فان الطابور سوف يتغير كما في الشكل (٢-٨)



شكل ٢-٨

واذا أتبع العمل HASSAN بعمل اخر هو PAYROL فان الطابور سوف يتغير ليصبح كما في الشكل (٢-٩).



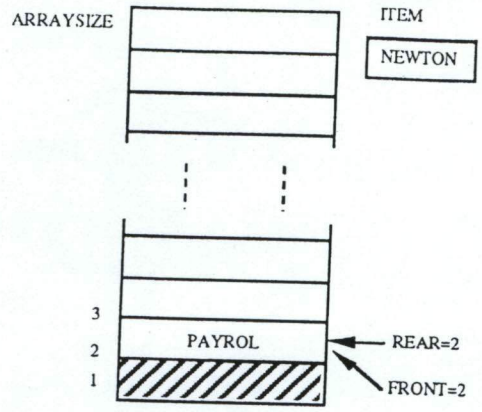
شكل ٢-٩

والتي توضح ان عملية الاضافة لاي عمل محتوى في عنصر (ITEM) للطابور تتطلب خطوتين وهما:

REAR: =REAR +1

Queue [REAR]: = I TEM

واذا كان النظام حالياً جاهزاً لمعالجة HASSAN فيجب تحريك او ازالة مدخلات البداية (FRONT) من الطابور الى موقع مناسب موضح في العنصر (ITEM) كما في الشكل (٢-١٠).



شكل ٢-١٠

واجراء ذلك يتم على النحو التالي:-

ITEM: = Queue [FRONT];

FRONT: = FRONT + 1

من الشرح السابق تتضح الشروط التالية:

الشرط	المعنى
1- REAR < FRONT	الطابور فارغ (queue enpty)
2- REAR = FRONT	مدخل واحد في الطابور (one entry Queue)
3- REAR = Arraysize	(No more entries may be added to queue) لا يمكن اضافة مدخلات إلى الطابور

٢- طوابير الاولويات: (Priority queues)

حتى الآن استخدمنا جدولة الحزم كمثال في استخدام الطابور في نظام التشغيل، وبالتحديد فإن جدولة الحزم (batch) يمكن أن تعطي اولويات عالية لبعض الاعمال.

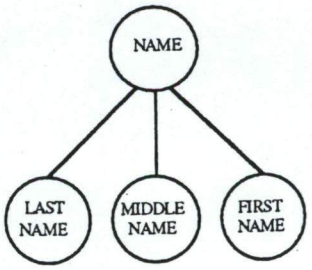
مثال: في الجامعة نجد أن طلاب السنة الاولى يأخذون المادة ذات الاولوية العالية حتى تشجعهم على التخصص أو البقاء في التخصص وبعدها يأخذون المادة التي تليها في الاولوية، ولكن الاعمال المرتبطة بالابحاث التي تقدم في القسم والتي تتطلب عمليات كبيرة يكون لها الاولوية الاقل، مثل هذه الاعمال يمكن أن تسمى A, B, C بالتتابع، أي عمل من A سوف يُخدم قبل أي عمل من B أو C وذلك بصرف النظر عن وقت دخولها إلى الطابور. وبالمثل فإن أي عمل من B سوف يُخدم قبل أي عمل من C. وتركيب البيانات القادر على تمثيل هذا الطابور يحتاج فقط إلى مؤشر أمامي (FRONT) واحد فقط وثلاثة مؤشرات خلفية (REAR) واحد لكل من A, B, C.

الوحدة الرابعة
الهياكل الشجرية المتصلة والمخططات
Trees & Graphs

١- مقدمة وتعريف للهياكل والمخططات

تعرفنا في السابق على الهياكل والمخططات وكذلك تعرفنا على شكل الهيكل الشجري (Tree) كما ورد في المثال عن الطالب وهو شكل (١-١).

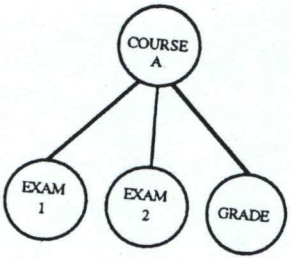
تعريف الهيكل الفرعي (Subtree): هو مجموعة جزئية من الهيكل الاصلي وهي نفسها تشكل هيكلاً (Tree). وبالاعتماد على شكل (١-١) الذي يحتوي على عقدة الجذر وهو الاسم (Name) في شكل (١-١) نجد ان الشكل التالي شكل (١-٤).



شكل ١-٤

يمثل هيكلاً فرعياً للهيكل الاصلي في شكل (١-١)

وكذلك الشكل التالي:

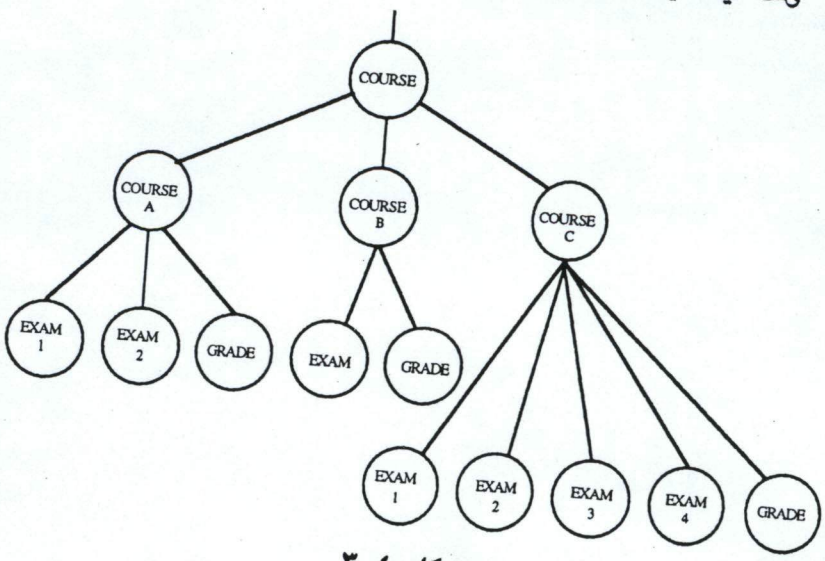


شكل ٤-٢

يمثل ايضاً هيكلاً فرعياً للهيكل الاصل في شكل (١-١).

وكذلك الشكل التالي شكل (٤-٣)

وهذا ايضاً يمثل هيكلاً فرعياً للهيكل الاصل (١-١).



شكل ٤-٣

بالاعتماد على ما سبق، نستطيع تعريف الهيكل (Tree) بدقة بطريقة متداخلة على اساس انها مجموعة من العقد التي اما تكون:

١- فارغة.

٢- لها عقدة مميزة تسمى الجذر (root) حيث بالامكان ان يتفرع منها عدد من الهياكل الفرعية او لا يتفرع منها هياكل جزئية. (اي لا تحتوي على هياكل جزئية وتسمى الهياكل الصفرية Zero Subtrees).

كل هيكل فرعي لوحده يطبق عليه تعريف الهيكل.

٢- الهياكل المرتبة والمصطلحات المتعلقة بها

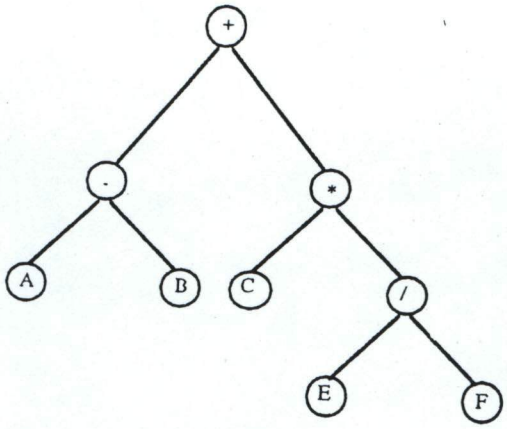
(٢-١) العناصر الاساسية للهياكل.

(٢-٢) البناء الهيكلي للهياكل.

شرحت من خلال تعريف الهياكل وكذلك الهيكل الفرعي.

٣- الهياكل الثنائية واشكالها المختلفة

وهي من اهم تركيبات الهياكل. اذ ترتبط كل عقدة في الهيكل الثنائي على الاكثر بعقدتين، وكل عقدة منها يمكن ان لا ترتبط مع عقد اخرى (leaf node) وبكلمات اخرى، في الهيكل الثنائي يكون لكل عقده هيكلان فرعيان (اما ان تكون فارغة او لا). يسمى احدهما هيكلاً متفرعاً ايسر والآخر هيكلاً متفرعاً ايمن، والشكل التالي (٤-٤) يبين هيكلاً ثنائياً.

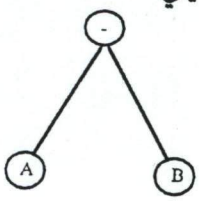


شكل ٤-٤

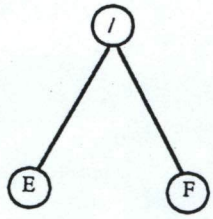
لاحظ من الشكل ان كلا العقدتين المرتبطتين مع العقدة التي تحتوي على اشارة "-" هي عقدة فرعية (leaf nodes) (اي العقدتين A و B) وهاتان العقدتان غير مرتبطتين مع عقد اخرى. وتكون من الجهة اليسرى.

بينما في الجهة اليمنى التي تحتوي على اشارة فان العقدة اليسرى منها هي عقدة فرعية.

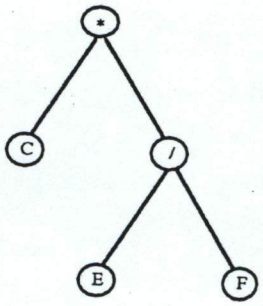
والهيكل الفرعي الايسر للاشارة "+" (العقدة التي تحتوي على اشارة "+" تسمى عقدة الاصل) تظهر كما يلي:



والهيكل الفرعي الايمن للاشارة "*" يظهر كما يلي:



من المثال السابق يجب ان يكون من الواضح انه اذا قمنا بالربط الفرعي بين العقدة الوليدة (child node) والعقدة الاب (أو الأصل)، فان العقدة الوليدة ستصبح عبارة عن اصل للشجرة الفرعية. وهذا واضح بالنسبة لتعريف الهيكل (عملية التداخل في التعريف). كما هو في الشكل التالي المعتمد على الهيكل (٤-٤).



شكل ٤-٤

حيث قمنا بتجزئة الهيكل الى هيكل فرعي للتعبير عن تركيب الهيكل، وتتضح عقدة الاصل فيها، حيث هي العقدة التي تحتوي على الاشارة "*". وكذلك فان الهيكل الثنائي ليس له تركيب متماثل (متناسق)، لأنه اذا حاولنا تبديل الهيكل الفرعي الايمن والايسر، سوف يؤدي ذلك الى تركيب جديد، وكذلك هيكل مختلف. وهذا واضح من شكل الهيكل الثنائي (التجزئة الى الهيكل الايمن والهيكل الايسر).

تركيب الهيكل يظهر بوضوح في كل من علم الحاسوب، وكذلك تطبيقات معالجة البيانات (مخطط فهرس الملفات File index schemes وكذلك أنظمة ادارة قواعد البيانات الهرمية مثل نظام ادارة المعلومات Information management system (IMS) المتوفر في أجهزة IBM ويستخدم تركيب الهيكل).

الهيكل في شكل ٤-٤ عندما يوضح بالطريقة الصحيحة يمثل التعبير الرياضي التالي:

$$(A - B) + C * (E/F)$$

وهذا التعبير حصلنا عليه بواسطة المرور (traversing) بعُقَد الهيكل في شكل ٤-٤ بطريقة معينة.

وتستخدم المترجمات الحديثة تركيبات الهيكل للحصول على اشكال للتعابير الرياضية من اجل التحليل الفعال. وهذه الاشكال هي:

Infix, Prefix and Postfix.

جميع هذه الاشكال متوفرة لدينا، فإذا عرفنا بالضبط كيف يمكن معالجة الهيكل المعني. فعملية المرور بالهيكل تعني معالجته بطريقة معينة، بحيث اننا نمر مرة واحدة على اي عقدة فيها.

ويعطينا (المرور المتناسق) Inroder traversal للهيكل الثنائي التعبير بشكل ثابت (Infix).

كما يؤدي (مرور الاتساق) Preorder traversal لنفس الهيكل الى شكل ما قبل الثبات (Prefix) للتعبير الرياضي.

ويؤدي (المرور ما بعد الاتساق) Postorder traversal لنفس الهيكل الى شكل ما بعد الثبات (Postfix) للتعبير الرياضي.

٤- العمليات المختلفة على الهياكل الثنائية وخوارزمياتها

العمليات المختلفة هي:

١- Preorder (ما قبل الاتساق) NLR

٢- Postorder (ما بعد الاتساق) LRN

٣- Inorder (متسق) LNR

ونعني بالاحرف R و L و N ما يأتي:

N: عقدة الجذر (الاصل)

L: العقدة اليسرى

R: العقدة اليمنى

١- عملية (ما قبل الاتساق) تكون Preorder الخوارزمية لها هي:

التمركز في الجذر a- visit the root

المرور بالفرع الايسر b- traverse the left subtree

المرور بالفرع الايمن c- traverse the right subtree

٢- وتكون الخوارزمية (لما بعد الاتساق) Postorder هي:

المرور بالهيكل الفرعي الايسر a- traverse left subtree postorder

المرور بالهيكل الفرعي الايمن b- traverse right subtree postorder

التمركز بالجذر c- visit the rot

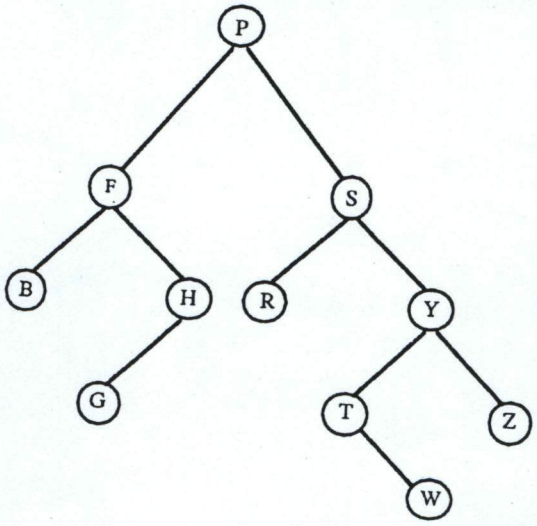
٣- أما عملية (التناسق او الاتساق) Inorder فخوارزميتها هي:

a- traverse left subtree in order — المرور بالهيكل الفرعي الايسر

b- visit the root — التمركز بالجذر

c- traverse right subtree inorder — المرور بالهيكل الفرعي الايمن

والشكل التالي يوضح صورة هيكل ثنائي:



اكتب ترتيب الاحرف (كل العقد) بالطرق الثلاث:

١- ما قبل الاتساق.

٢- ما بعد الاتساق.

٣- الاتساق.

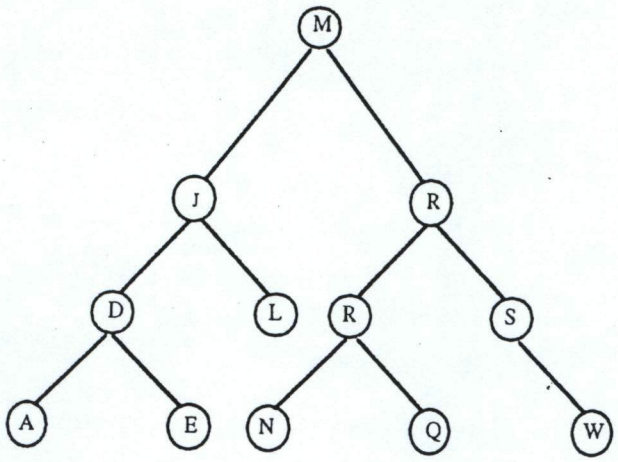
١- العقد ما قبل الاتساق — PFBHGSRYTWZ

٢- العقد في ما بعد الاتساق — BGHFRWTZYSP

٣- الاتساق

BFGHPRSTWYZ

مثال:



اجب عن الاسئلة التالية:

١- دوّن جميع العقد في مرحلة ما قبل الاتساق.

الحل:

MJDAELRPNQSW

٢- دوّن جميع العقد في مرحلة الاتساق.

الحل:

ADEJLMNPQRSW

٣- دوّن جميع العقد في مرحلة ما بعد الاتساق

الحل:

AEDLJNQPWSRM

٤- دوّن عقد الاصل للعقدة P

الحل:

M, R

٥- دوّن جميع العقد في المستوى 3:

الحل:

A, E, N, Q, W

مدخل إلى لغة سي Intoduction to C Lannguage

المقدمة :

ترجع بدايات لغة السي إلى الفترة ما بين 1969 و 1973 حيث قام دينيس ريتشي Dennis Ritchie بتطوير هذه اللغة وقد كان ظهور لغة سي متزامنا مع بدايات تطوير نظم التشغيل الشهير يونكس ، Unix وقد شهدت لغة السي تطورا كبيرا في عام 1972. كما كانت الفترة فيما بين 1977 و 1979 فترة تطور كبير في لغة السي ، ففي منتصف هذه الفترة ظهر أول وصف للغة السي والذي أصبح شائعا فيما بعد وهو . The C programming language حتى بداية الثمانينات وعلى الرغم من وجود العديد من المترجمات Compilers التي تتيح العمل على أنواع مختلفة من الحاسبات ونظم التشغيل إلا أن لغة السي كانت مرتبطة وعلى وجه الحصر بنظام التشغيل يونكس.

وأخيراً في منتصف الثمانينات أصبحت لغة السي رسميا لغة معيارية وقد تم ذلك بواسطة اللجنة ANSI X3J11 والتي قامت أيضاً بإدخال بعض التغيرات في اللغة. بعد ذلك أصبحت هذه اللغة تستخدم بشكل كبير حتى أصبحت اللغة الأكثر انتشارا لفترة طويلة من الزمن.

لا تعتبر لغة السي من اللغات الصعبة ولكنها في ذات الوقت لا تخلو من بعض الصعوبة حيث تحتاج اللغة إلى الكثير من الممارسة والمثابرة حتى يتمكن الدارس من إتقانها جيدا.

فوائد لغة سي:

1. تعتبر لغة السي هي اللغة الرئيسة المستخدمه في تطوير نظم التشغيل وهي اللغة الرئيسية المستخدمه في برمجة نظم التشغيل الشهيرة مثل: نظام تشغيل مايكروسوفت ويندوز، نظام التشغيل يونكس، نظام التشغيل لينيكس.
2. لا يوجد شيء لا تستطيع لغة السي عمله فهي تتميز بعدم ارتباطها بنظام تشغيل معين او نوع معين من الحاسبات وان كان استخدامها لتطوير برامج ضخمه يحتاج إلى مجهود جبار لاداء ذلك.

مميزات لغة سي:

- يعود سبب اشتهار لغة السي إلى كونها تعمل بسرعة وهي في نفس الوقت سهلة نسبيا للتعلم والاستخدام. وتعتبر لغة السي لغة مركبة، تتطلب من المبرمج أن يفكر بكل برنامج كمجموعة مقاطع اي ان البرنامج يتم تقسيمه تقسيما وظيفيا من قبل المبرمج كل جزء منه يسمى إجراء Function تشكل مجتمعة برنامجا متكاملا .

يمكننا ان نجمل ما يميز لغة C بما يلي :

- **لغة عامه:** أي تصلح لعمل برامج قواعد البيانات والرسومات والحسابات ونظم التشغيل.
- **لغة تركيبية Structured Language :** اي ان البرنامج المكتوب بهذه اللغة عبارة عن اجراء Function رئيسي ينادي جميع الإجراءات الاخري .
- **لغة متنقلة Portable Language :** البرنامج المكتوب بلغة C يعمل على اكثر من نوع من انواع الأجهزة مثل IBM و Apple و ..
- **لغة قياسية :** المترجم الخاص بلغة C يتوافق مع اللغة القياسية ANSI

Turbo C

البرنامج Turbo C++ من إنتاج شركة بورلاند انترناشونال وهي نسخة شائعة الاستعمال للبرمجة بلغتي C++/C وهي مصممة للعمل مع الحاسبات الشخصية صنع شركة IBM والحاسبات المتوافقة معها..

تحتوي المجموعة Turbo C++ على مجموعتين مترابطتين من الأدوات

اللغة وبيئة التطوير

- اللغة C/C++: توفر العمليات وأوامر التشغيل التي تستعملها لإنشاء برامجك الخاصة.
- بيئة التطوير للغة: Turbo C++ وهي مجموعة من قوائم الاختيار والنوافذ والمزايا المفيدة التي تبسط عملية إنشاء برامج بلغة البرمجة C أو بلغة البرمجة C++.

تحميل البرنامج على الكمبيوتر :

يوجد البرنامج على قرص ضوئي، أو عادة ما يكون على أقراص مرنة (3 أقراص)، وسنخوض في هذا البند كيفية تحميل البرنامج من الأقراص المرنة الثلاث:

أولاً: أدخل القرص المرن #1 INSTALL/HELP في محرك الأقراص المرنة A ثم قم بفتحه ستجد مجموعة من الملفات من ضمنها ملف INSTALL ، قم بالنقر عليه مرتين للبدء في تنصيب البرنامج.

ثانياً: ستظهر لك رسالة حقوق الطبع للبرنامج INSTALL على الشاشة.

لتشغيل البرنامج إذا كنت تستخدم التيربو سي نحتاج للقيام بخطوتين:

1. ترجمة compilation للبرنامج ، ليتحول إلى صيغة ، file_name.obj من قائمة Compile اختر Compile واختصاره Alt + F9 وسوف تظهر لك رسالة بالأخطاء الموجودة في البرنامج إذا وجدت أو رسالة بعدم وجود أي أخطاء في حال كان برنامجك خاليا من الأخطاء.
2. Linking للبرنامج ليتحول إلى ملف تنفيذي بامتداد EXE من قائمة Compile اختر .Link وستظهر لك رسالة تفيدك بان البرنامج تحول إلى صيغة exe بشكل سليم وبدون أي أخطاء.

تبقي لنا تشغيل البرنامج بعد أن أصبح على صيغة ، exe اختر أمر التشغيل Run من القائمة Run واختصاره Ctrl+F9 وستظهر لك نافذة سوداء تختفي بسرعة كبيرة، وهذا امر طبيعي في الـ Turbo C++ وسنبين فيما يلي كيف سترى نتائج برنامجك..

مشاهدة المخرجات

هناك طريقتين لمشاهدة المخرجات أو نتائج تشغيل برنامجك..

الطريقة الأولى: من قائمة Window اختر Output وسوف تظهر لك نافذة فيها مخرجات برنامجك الذي قمت بتشغيله مسبقا.. وتستطيع القيام بذلك بسرعة من لوحة المفاتيح بالضغط على مفتاحي Alt+F5.

الطريقة الثانية: تستطيع إضافة السطر **getch();** أو السطر **getchar();** في نهاية الجزء الرئيسي main() من برامجك وهذا يؤدي لثبات نافذة المخرجات فور استخدام الأمر ، Run

ويصبح البرنامج السابق بالشكل التالي :

```
/* Welcome program */
# include < stdio .h >
void main ( )
{
    printf ( " welcome to C Language " ) ;
    getch();
}
```

أنواع البيانات *Data Type*

في لغة سي يمكن ان تكون البيانات اما أرقام أو حروف :

- الأرقام يمكن أن تكون صحيحه (أي ليس بها علامة عشرية) integer
  حقيقية (أي بها علامة عشرية) float .
- والحروف يمكن أن تكون حرف واحد أو أكثر من حرف.

الجدول التالي يوضح هذه الأنواع وكذلك عدد البايت byte التي يشغلها :

نوع المتغير	الوصف	طوله بالبايت	المدي المسموح
(char)	حرف	1	حرف او رمز واحد
(int)	صحيح قصير	2	32768 إلى 32768
(long)	صحيح طويل	4	2014704830648 إلى 2014704830648
(float)	حقيقي	4	e + إلى 38 e- 38
(double)	حقيقي مضاعف	8	e+ إلى 308 e -308

وفي ما يلي المقصود بكل هذه الأنواع:

- **متغير من نوع حرف:** أي متغير يصلح لتخزين حرف فقط.
- **متغير من نوع صحيح:** أي متغير يصلح لتخزين رقم صحيح (ليس به علامة عشرية).
- **متغير من نوع صحيح طويل (Long):** أي يستطيع أي يخزن رقم صحيح ضعف المتغيرالصحيح العادي ويستعمل هذا النوع إذا كانت الأرقام التي تتعامل معها أكبر من المساحة المخصصة وإلا سنحصل على نتائج خاطئة بالرغم من إن البرنامج سليم.
- **متغير حقيقي:** أي متغير يصلح لتخزين رقم حقيقي يقبل الكسور العشرية مثل 5.33 .
- **متغير حقيقي مضاعف :** أي يستطيع أن يخزن رقم حقيقي ضعف المتغير الحقيقي العادي.

تسمية المتغير:

يخضع اسم المتغير لشروط معينه:

- يجب أن يبدأ المتغير بحرف.
- يفرق المترجم بين الحروف الصغيرة والكبيرة فالمتغير X يختلف عن المتغير x (Case sensitive).
- الا يكون المتغير بإسم كلمه من الكلمات المحجوزة (ليس امر من اوامر السي).

الإعلان عن المتغيرات:

في لغة ال C لابد الإعلان عن المتغيرات Variables في بداية البرنا او في إي مكان بالبرنامج ، والصيغة العامة لتصريح متغير تكون بالشكل التالي

```
Data_type variable_name ;
```

Data_type : يقصد بها نوع البيانات ، حيث يمكن ان تكون رقم صحيح عشري او حرف او غير ذلك .

Variable_name : ويقصد بها اسم المتغير .

واليك بعض الامثلة لتصريح المتغيرات :

```
int a ;
float x ;
char char1;
```

التعابير الحسابية والمنطقية

تبنى التعابير الحسابية والمنطقية على اساس مجموعة من المعاملات أو يسمى بالمشغلات ، والمعاملات (المشغلات) هي الرموز التي تربط بين المتغير والثوابت لإنشاء معادلة ، تختلف أنواع المشغلات باختلاف وظيفة كل منها.

1- المشغلات الحسابية arithmetic operators

+ علامات الجمع addition
- علامات الطرح Subtraction

* علامات الضرب multiplication

/ علامات القسمة division

وتستخدم مع المتغيرات والثوابت الرقمية

2- مشغلات المقارنة Relational operators:

وتستخدم لمقارنة قيمتين، وتعطي نتيجة منطقية اما صح true (1) او خطا false (0).

المؤثر	الرمز	مثال	النتيجة
أكبر من greater than	>	10>8	1
أصغر من less than	<	10<8	0
يساوي equal to	==	10==8	0
لا يساوي not equal to	!=	10!=8	1
أقل من أو يساوي less than or equal to	<=	10<=8	0
أكبر من أو يساوي greater than or equal to	>+	10>=8	0

3- المشغلات المنطقية Logical operator

العملية (&&) وتعني (و and)

العملية (||) وتعني (أو or)

العملية (!) وتعني (لا not)

وتعطي نتيجة منطقية اما صح true (1) او خطا false (0) ، كنتيجة منطقية بين عمليتي مقارنة (للعمليتين && و ||) وكنتيجة لعملية مقارنة واحدة في عملية النفي المنطقية (!) ، كما تلخصها الجداول التالية : (1 هي true ، 0 هي false)

العملية (&&) وتعني (و and)

العملية X	العملية Y	X && Y
0	0	0
0	1	0
1	0	0
1	1	1

يتضح لنا من خلال الجدول بأن نتيجة العملية المنطقية && تكون true (1) فقط عندما يكون نتيجة كلا العمليتين هو true وعدا ذلك فان النتيجة تكون false ، اي عندما يتحقق الشرطين ، كأن يكون شرط نجاح الطالب في المادة هو نجاحه في الامتحانين وليس احدهما .

العملية (||) وتعني (أو or)

العملية X	العملية Y	X && Y
0	0	0
0	1	1
1	0	1
1	1	1

يتضح لنا من خلال الجدول بأن نتيجة العملية المنطقية || تكون false (0) فقط عندما يكون نتيجة كلا العمليتين هو false وعدا ذلك فان النتيجة تكون true ، اي عندما يتحقق احد الشرطين تكون النتيجة true ، كأن يكون شرط نجاح الطالب في المادة هو نجاحه في أحد الامتحانين وليس كلاهما .

العملية (!) وتعني (لا not)

العملية X	X && Y
0	1
1	0

وتستخدم لنفي نتيجة عملية معينة .

والجدول التالي يوضح امثلة بسيطة على العمليات المنطقية:

المؤثر	الرمز	مثال	النتيجة
و AND	&&	1<0 && 5>1	1
أو OR	||	1<2 || 5<0	1
لا NOT	!	!(8==8)	0

4- مشغلات التخصيص Assignment Operators

وهي مشغلات اسناد قيمة إلى متغير (تخزين قيمة في متغير ما او تغيير قيمته)، والمشغلات هي: (=+) ، (=-) ، (=*) ، (=/)

المؤثر	التخصيص البديل	العملية
addition assignment operators	X+=2	X=X+2
Subtraition assignment opertors	X-=2	X=X-2
Multipication assibnment operators	X*=2	X=X*2
Division assignment operators	X/=2	X=X/2

5- مشغلات الزيادة والنقصان بمقدار 1 Decrement &increment

مشغلات تستخدم للزيادة بمقدار واحد (++) او النقصان بمقدار واحد (--)

حيث ان : العملية X=X+1 يمكن اختصارها بـ X++ ، والعملية X=X-1 تستبدل بـ X—

X++	X=X+1
X--	X=X-1

6- مشغل باقي القسمة %

يستخدم لمعرفة باقي القسمه (فمثلا قد تستخدم في معرفة العدد الزوجي من الفردي.

مثال:

إذا كانت قيمة a = 5 فان C = a %2 يكون باقي الرقم 2 / 5=1 اي ان القيمة النهائية للمتغير C تكون 1

أولويات التنفيذ للمشغلات الرياضية precedence of Arthmatic operators

ترتيب الاولوية	المشغلات	عند المشاركه
1	()	
2	* / %	Left to right
3	+ -	Left to right
4	< <= > >=	Left to right
5	!= ==	Left to right
6	=	
في حال التساوي في الأولوية يتم التنفيذ من اليسار إلى اليمين .		

مثال : اذا كانت لديك المعادلة التالية :

$$R = \frac{a+b+c+d+e+f}{6}$$

فان C تفهمها بالشكل التالي :

$$R = (a+b+c+d+e+f) /6$$

ويقوم المترجم بتطبيق أولويات التنفيذ ، حيث ان الولوية الاولى تكون لما بين الاقواس (a+b+c+d+e+f) ونلاحظ ان ما بين الاقواس هو اكثر من عملية جمع أي المشاركة بنفس العملية ، وعندها يتم التنفيذ من اليسار الى اليمين أي جمع a+b والناتج يتم اضافته الى c وهكذا ، ثم يقوم بقسمة ناتج ما بين الاقواس على 6 ، وناتج عملية القسمة يتم اسناده الى المتغير R .

مثال : اليك المعادلة التالية ، ما هو الناتج ؟

$$R= 2 * 5 * 5 + 3 * 5 - 9$$

تتبع معي كيف يمكن معرفة الناتج

اولا عليك ان تلاحظ ان المشغلات الموجوده هي الجمع (+) والضرب (*) ، ومن المؤكد ان الاولوية الولى تكون للضرب وعليك الا تنسى أنه عند المشاركة فان الولوية تكون للمشغل الموجود لأقصى اليسار .

الخطوة الاولى : 2 * 5 والناتج هو **10** فتصبح المعادلة بالشكل التالي :

$$R= \underline{10} * 5 + 3 * 5 - 9$$

الخطوة الثانية : 10 * 5 والناتج هو **50** فتصبح المعادلة بالشكل التالي :

$$R= \underline{50} + 3 * 5 - 9$$

الخطوة الثالثة : نلاحظ فيها ان عملية الضرب يجب ان تسبق الجمع ، وعليه :

3 * 5 والناتج هو **15** فتصبح المعادلة بالشكل التالي :

R= 50 + 15 - 9

الخطوة الرابعة : نلاحظ أن الشغلات المتبقية جميعها جمع ، وبالتالي نطبق قاعده المشاركة (من اليسار لليمين)

50 + 15 والناتج هو **65** فتصبح المعادلة بالشكل التالي :

R= 65 - 9

الخطوة الخامسة : 65 - 9 والناتج هو **56** فتصبح المعادلة بالشكل التالي :

R= 56 ← — — الناتج النهائي

Input and Output الإدخال والإخراج

توفر لغة سي مجموعة من الاجراءات التي يمكن من خلالها القيام بعملي
ادخال البيانات للبرنامج ، وكذلك اخراج النتائج على الشاشة.

كل اجراء مرتبط بملف توجيه معين حيث يستدعي هذا الملف في أو
البرنامج بالعبارة include # فمثلا الاجراء () printf معرف بالملف dio.h:
ويكتب العبارة#include <stdio.h> في أول البرنامج حتي يتعرف المترجم عل
الاجراء وهكذا مع باقي الاجراءات.

إجراء الطباعة على الشاشة printf( )

- تستخدم اجراء الطباعة printf ( ) لطباعة البيانات بجميع أنواعا
(string , char , float , int) على الشاشة فقط.
- يأخذ اجراء الطباعة عدة صور وكذلك معاملات واختصارات تحد
شكل المخرجات

مثال 1

intf (" welcome To C Language ")

هنا يتم طباعة ما بين علامتي التنصيص" "

مثال 2

```
# include < stdio.h >
void main ( )
{    printf ( " \n welcome \n To \n C language " ) ;
}
```

تراكيب التحكم Control Structure

أحيانا يحتاج المبرمج لإعادة كتابة جملة معينة عدة مرات أو قد يحتاج لإجراء عملية حسابية لمجموعة متجانسة من القيم ، وفي أحيان أخري قد يحتاج لتنفيذ مجموعة من الجمل عند تحقق شرط معين ، هنا يتبين لنا انه لا بد من وجود جمل برمجية تساعد المبرمج على أداء ما سبق ذكره ، وهي ما يطلق عليها اسم جمل التحكم ، والتي قد تكون جمل دوران أو جمل شرطية ، وسنسهب فيما يلي بهذا الموضوع .

أولا :التكرار Loop

1- جملة الدوران *For*

تستخدم لتكرار تنفيذ عملية او مجموعة عمليات عدد محدد من المرات

الصورة العامة:

```
For ( initial – value ; condition ; step )
{
    C_statement ;
}
```

حيث:

initial – value : هي القيمة الابتدائية لعداد جملة الدوران

condition : هو شرط الاستمرار في تنفيذ جملة الدوران

Step : هي قيمة الزيادة على العداد للانتقال للدورة التالية

حيث ان الدوران يستمر طالما جواب الشرط condition هو true.

يمكن الاستغناء عن الأقواس { } و { إذا كانت الـ C_statement أي الجمل التي سيتم تنفيذها في جملة الدوران هي جملة وحيدة

مثال : برنامج يطبع الأرقام من صفر إلى 100

```
# include < stdio.h >
void main ( )
{
int i ;
for ( i = 0 ; i <= 100 ; i ++ )
printf ( "i = % d \n " , i ) ;
}
```

في البرنامج السابق احتجنا لجملة دوران عوضا عن طباعة كل رقم بجملة طباعة مستقلة ، حيث سنحتاج لكتابة الجملة printf 101 مرة ، بينما باستخدام جملة الدوران فهي جملة وحيدة فقط ، ونحتاج لعداد يقوم بعد الدورات وايقاف الدوران عندما تصل قيمته إلى 101 .

مثال :

حساب مجموع القيم المحصورة بين 5 و 50

```
# include < stdio.h >
void main ( )
```

```
{
int i , s=0;
for ( i = 5 ; i <= 55 ; i ++ )
        s += i;
printf ( "summation = % d \n " , s ) ;
}
```

وهنا بدأ العداد من القيمة 5 ، ويزداد بكل دورة بمقدار واحد ويقوم بإضافة قيمة العداد إلى مجموع القيم (s) ، وعند وصول قيمة العداد إلى القيمة 56 ينتهي الدوران ويتم الخروج من جملة الدوران وطباعة المجموع .

تنفيذ أكثر من جملة مع *For* :

لتنفيذ أكثر من جملة لعدد محدد من المرات يجب بدأ جملة for بالقوس { وإنهائها بالقوس }

مثل : حساب متوسط قائمة من الأعداد ، حيث يقوم المستخدم بإدخال عدد القيم التي يريد إدخالها ، ثم يقوم بإدخال القيم نفسها وحساب معدلها .

```
# include < stdio.h >
/ * calculate the average of n numbers * /
void main ( )
{
int n , count ;
```

```
float x , av , sum = 0 ;
printf ( " how many numbers ? " ) ;
scanf ( " % d " , & n ) ;
for ( count = 1 ; count < = n ; count ++ )
{
    printf ( " x = " ) ;
    scanf ( " % f " ; & x ) ;
    sum + = x ;
}
av = sum / n ;
printf ( " average = % f  " , av ) ;
getche();
}
```

جملة *For* المتداخلة

الدوارات المتداخلة عبارة عن دوارة كبيرة تشتمل بداخلها على دوارة أو أكثر.

بمعني أن مجموعة التعليمات بال loop الداخلي يتم تكرار تنفيذها طالما لم ينته العداد فإذا انتهى ينتقل التنفيذ إلى الدوارة الخارجية ويتم تكرار التعليمات في ال Loop الخارجي حتى ينتهي.

مثال : طباعة جدول الضرب

```
# include < stdio.h >

void main ( )

{

int i , j ;

for ( i = 1 ; i < = 10 ; i ++ )

        for ( j = i ; j < = 10 ; j ++ )

                printf ( "%d * %d = % d  \n " , i , j , i * j ) ;

}   // end main
```

مثال: طباعة الشكل التالي :

```
****
****
****
****
```

```
# include < stdio.h >
void main ( )
{
int i , j ;
for ( i = 1 ; i < = 4 ; i ++ )
        for ( j = i ; j < = 4 ; j ++ )
                printf ( "*" ) ;
}
```

نلاحظ في البرنامج بأننا بحاجة لطباعة 4 صفوف لذلك قمنا بتجهيز الـ loop الخارجي ليسير بـ 4 دورات ، وكذلك نحن بحاجة لطباعة 4 نجوم في كل صف ، وهنا يجب أن تدور جملة الدوران الداخلية 4 دورات ، وفي كل دورة يتم طباعة نجمة واحدة .

مثال: طباعة الشكل التالي :

```
*
**
***
****
```

```
# include < stdio.h >
void main ( )
{
int i , j ;
for ( i = 1 ; i < = 4 ; i ++ )
        for ( j = i ; j < = i ; j ++ )
                printf ( "*" ) ;
}
```

الدوران الغير منتهي

معناها تكرار تنفيذ الجملة بدون شرط ولا يتوقف التنفيذ حتى يضغط المستخدم CTRL + C وتاخذ الدوارة اللانهائية الصورة:

```
For ( ; ; )
```

2- جملة الدوران *While*

تستخدم الدوارةwhile لتكرار تنفيذ جملة أو مجموعة جمل عدد من المرات غير معلوم العدد ويتوقف هذا على شرط موجود بالدوارة

الصورة العامة:

```
Initial_value
While(condition){
    C_statements;
    ...
    ...
}
```

- يتم تكرار مجموعة الأوامر بين الأقواس { } عدد من المرات يتوقف على الشرط بين الأقواس ( )
- عندما يصبح الشرطFalse . يتوقف تنفيذ هذه الأوامر ويستمر أداء باقي البرنامج.
- حينما يكون الشرط False من بداية التكرار ، لا يتم تنفيذ أي من جمل سي C_statements

مثال : تعديل برنامج حساب متوسط قائمة من الأعداد باستخدام While

```
# include < stdio.h >
void main ( )
{
int n , count = 1 ;
float x , average , sum = 0 ;
printf ( " how many number ? " ) ;
scanf ( " % d , & n ) ;
while ( count < = n )
{
printf ( " x = ") ;
scanf ( " % f " , & x ) ;
sum + = x ;
count ++;
}
average = sum / n ;
printf ( " \ n the average is % f " ; average ) ;
}
```

الفرق ما بين *For* و *While*

- الـ For جملة دوران تعتمد على العداد وينتهي التكرار فيها بانتهاء عدد مرات التكرار
- أما الدوارة while فدوارة شرطية أي تعتمد على الشرط الذي يلي الأمر while حيث تتكرر الجمل التي تليها طالما كان الشرط صحيحاً.

3- جملة الدوران *Do ... while* :

تستخدم do while لتكرار تنفيذ جملة أو مجموعة جمل أكثر من مرة بناء على شرط معين أي يتم تنفيذ الجمل التالية لDo ثم تختبر الشرط فإذا كان صحيحا تعيد التنفيذ وإلا توقف التكرار

إذن يتم التنفيذ مرة واحد على الأقل ثم يتم اختبار الشرط المنطقي ( ) فإذا كانTrue ينفذ مرة أخري وإذا كانFalse يتوقف التكرار ويستمر تنفيذ البرنامج الأصلي

مثال : إيجاد مجموع الأرقام من 0 إلى 9

```
# include < stdio.h >

void main ( )

{

int count = 0 ;

int total = 0 ;

do{

      total + = count ;

      printf ( " count = % d , total = % d \ n " ,
                  count++ , total ) " ) ;

  } while ( count < = 10 ) ;

}
```

الإجراءات FUNCTIONS

الإجراء هو مجموعة من الجمل البرمجية التي لها وظيفة معينة و يكون لـ
اسم مميز حتى يتم مناداته به، وتقسم الإجراءات في C إلى نوعين :

1) إجراءات اللغة الجاهزة Built in Functions

وهي الإجراءات القياسية مثل إجراء printf( ) أو إجراء scanf( ).
وهي إجراءات عامة يستطيع أي مبرمج استخدامها

2) إجراءات المستخدم User Functions

وهي الإجراءات التي يقوم المبرمج ببنائها وكتابتها .

والهدف منها: انه عند تكرار مجموعة من الأوامر أكثر من مرة في
مواضع مختلفة فإن أوامر التكرار لن تكون ذات منفعة . ولذلك يتم كتابة
هذه الجمل بشكل منفصل عن البرنامج الأساسي ، ويتم مناداتها عند الحاج
إليها في البرنامج ، وكذلك فان وجود الـ functions في البرنامج يزيد من
كفاءة البرنامج ويزيدها تنظيما ويسهل عملية تعديله ، حيث أن كل إجرا
يكون واضحا للمبرمج ويسهل عليه الرجوع إليه في أي وقت بدلا من قراء
البرنامج كاملاً.

مزايا استخدام الإجراءات

عدم تكرار التعليمات داخل البرنامج: حيث يتم إنشاء الإجراء مره واحدة
ثم يتم استدعائها أكثر من مرة عند الحاجة إليها. باستخدام الإجراءات يصبح
البرنامج أكثر وضوحًا.

تنظيم البرنامج

تسهيل مراجعة أي جزء من البرنامج من قبل المبرمج

الصيغة العامة :

```
return_value_Type function_name ( parameters )
    {
        C_ STATEMENTS ;
    }
```

نلاحظ بأن رأس الإجراء يتكون من جزاءا ثلاثة أساسية وهي :

Return_value_Type وهي نوع البيانات التي يعيدها الإجراء إلى مكان استدعاء الإجراء في البرنامج ، حيث قد تكون هذه البيانات من نوع int , float , ...

function_name : اسم الإجراء ، ويخضع إلى نفس شروط تسمية المتغيرات .

(parameters) : متغيرات تمثل القيم التي يستقبلها الإجراء .

{ ، رمز بداية الإجراء

C_ STATEMENTS ; ، مجموعة من جمل C والتي تؤدي الوظيفة المطلوبة من الإجراء .

} ، رمز نهاية الإجراء .

وسيتم توضيح ذلك بشكل مركز في الأمثلة اللاحقة .

أما مكان وجود الإجراء في البرنامج بشكل عام ، يوضحه التركيب العام التالي :

```
function declarations;
main( )
{
function1;  // call function1
function2;  // call function2
}
function1
      {
      ------
      ------
      }
function2
      {
      ------
      ------
      }
```

إذن تتكون الإجراء من:

function declarations; الإعلان عن الإجراء

function body جسم الإجراء

call function مناداة الإجراء

[٩-١] مقدمة

رغم أن الحاسب الالكتروني يتميز بقدرته على إنجاز العمليات الحسابية حسب الأوامر والتعليمات المعطاة له بسرعة فائقة وبدقة متناهية وكذلك بإمكانياته الكبيرة في حفظ المعلومات الواسعة والمختلفة التي يعجز الإنسان عن حفظها واستعادتها باستعمال ذاكرته العادية. فهو يعجز عن أن يقوم بشكل ذاتي بحل أية مسألة مهما كانت بسيطة، أي أن عمله ينحصر في إنجاز الحلول للمسائل التي تبرمج له بشكل صحيح يتوافق مع الأسس العلمية الصحيحة التي تعتمد عليها هذه الحلول.

لذا سوق نستعرض في هذا الفصل الخطوات الضرورية اللازمة لحل المسائل باستخدام الكمبيوتر وكذلك توضيحاً مفصلاً لمفهوم الخوارزميات وخرائط سير العمليات التي تشكل العنصر الأساسي لكيفية البرمجة.

[٩-٢] خطوات حل المسألة باستخدام الكمبيوتر

يمكن إيجاز مراحل حل مسألة ما باستخدام الكمبيوتر على النحو التالي:

المرحلة الأولى: وتتمثل في دور الإنسان في حل المسألة وتتكون من الخطوات الآتية:

(١) تعريف وتحليل المسألة: Problem definition and Analysis

الخطوة الأولى هي تعريف المسألة والمقصود بتعريف المسألة هو كتابة موضوع

٦٠

المسألة بصورة جلية وواضحة دون أي غموض.

أما تحليل المسألة واختيار طريقة الحل فهو أصعب المصاعب وأشق الخطوات، ومن أجل الوصول إلى حل صحيح فإن كثيراً من العلاقات الرياضية والقوانين المناسبة لحل المسألة يجب أن تستعمل ولربما تحتاج أيضاً إلى تطوير هذه القوانين والعلاقات لتجعلها تناسب الحل باستخدام الحاسب الالكتروني وفيما يلي العناصر الأساسية التي يمكن التعرض إليها أثناء تحليل المسألة:

(أ) تحديد أهداف المسألة وذلك بتحديد عناصر المخرجات (النتائج المطلوب التوصل إليها) وتحديد نوعها (بيانات حرفية أو رقمية أو حرفية رقمية) وتحديد أسلوب عرضها والوسط الخارجي المناسب.

(ب) تحديد عناصر المدخلات (المعطيات) وتشمل المتغيرات والمعلمات (parameters) ذات التأثير واللازمة للوصول إلى قيم عناصر المخرجات وكذلك يجب تحديد نوع عناصر المدخلات وتوصيفها بدقة.

(جـ) تحديد العلاقات الرياضية (Relations) المختلفة التي يشملها الحل وحصر طرق الحل المختلفة وتقييمها من أجل اختيار الحل الأفضل الذي يتلاءم مع كيفية تنفيذه بالحاسب الالكتروني.

ويشترك في هذه الخطوة أخصائي في موضوع المسألة كأن يكون محاسباً إذا كانت المسألة في المجال المحاسبي أو مهندساً إذا كانت المسألة هندسية بالإضافة إلى محلل أنظمة (System Analyst) المختص في علم الكمبيوتر ومهمته تهيئة المسألة ووضع أسلوب الحل بما يتناسب مع متطلبات المستخدم وإمكانية جهاز الكمبيوتر.

(٢) برمجة الحل تخطيطياً:

بعد اختيار طريقة الحل الأمثل وتحديد كل ما تشمله من علاقات رياضية، يتم تنظيم طريقة الحل المقترحة على شكل خطوات متسلسلة ومترابطة منطقياً، دقيقة الوصف، يعبر عنها باللغة العادية إلا أنه يحكمها المنطق الرياضي، ومجموع هذه

الخطوات يسمى بـ **خوارزمية المسألة**[1] (Algorithm of the problem) وقد يكون للحل الواحد أكثر من خوارزمية واحدة إلا أن جميع هذه الخوارزميات تؤدي إلى نفس النتيجة ولكن بطرق مختلفة ومتفاوتة. وقد يكون في كثير من الأحيان وضع مخطط وصفي تسلسلي يسمى **مخطط سير العمليات**[2] (Flowchart) وذلك باستخدام مجموعة من الأشكال الرمزية الاصطلاحية التي تكون أكثر وضوحاً وأنسب فهماً.

وبواسطة خريطة سير العمليات يمكن ملاحظة كيفية تتبع الحل والعلاقات المنطقية التي تظهر في الحل. وقد تقوم خريطة سير العمليات مقام الخوارزمية وتعتبر وثيقة مهمة من وثائق حل المسألة يمكن الرجوع إليها عند الضرورة وفي غياب المبرمج أو المحلل الذي قام بتصميمها ومن ناحية أخرى فإن استخراج الخوارزمية من خريطة سير العمليات يكون أسهل بكثير من كتابة الخوارزمية نفسها.

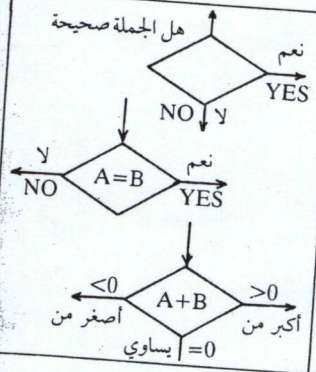
والشكل (٩-١) يبين أهم الرموز والأشكال الاصطلاحية العامة المستخدمة في تخطيط خرائط سير العمليات لحل مسألة معينة.

	رمز البداية أو النهاية Terminal symbol لبيان البداية أو النهاية	START STOP
	رمز الإدخال أو الإخراج (بدون تحديد وسيلة الإدخال أو الإخراج) Input/output symbol لبيان إدخال المعطيات إلى الحاسب أو إخراج المعلومات.	Read A,B Print C

(١) كلمة خوارزمية (Algorithm) مشتقة نسبة إلى العالم العربي المشهور الخوارزمي الذي قام بوضع أسس حل المسائل بشكل تتابعي وقد تسمى الخوارزميات بالمنطقيات أحياناً.

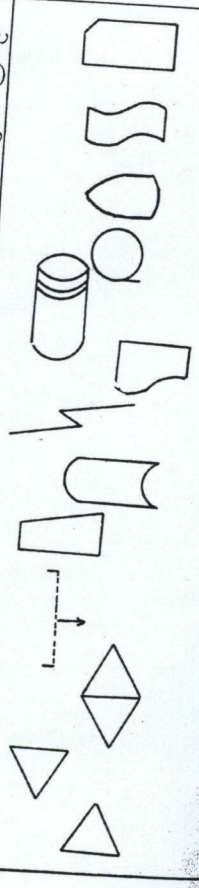
(٢) خريطة سير العمليات (وقد تسمى مخطط الانسياب) وهي عبارة عن مجموعة من الأشكال الرمزية البيانية يمثل كل شكل منها عملية أو حالة معينة ضمن الحل ويوصل بين هذه الأشكال بأسهم تبين عملية تتابع الحل.

تابع الشكل (٩-١)

رمز العملية الحسابية Processing symbol لبيان وجود عملية حسابية رياضية يراد انجازها		LET C = A + B
رمز القرار المنطقي logical decision symbol لبيان اتجاه التسلسل المنطقي لتنفيذ العمليات حسب جواب الشرط المتضمن في عملية منطقية داخل الرمز وقد يكون له مخرجين الأول إذا كان الجواب نعم والثاني إذا كان الجواب لا، أو ثلاثة مخارج تعتمد على قيمة التعبير A+B أقل أو أكبر أو يساوي صفراً.		
رمز تكرار أو دوران loop symbol		I=1, 10, 2
رمز الاتصال connection symbol رمز اتجاه مسار التنفيذ		
رمز استدعاء برنامج فرعي		CALL SUBROUTINE

شكل (٩-١)

والشكل (٩-٢) يبين الرموز والأشكال الاصطلاحية التخصصية المستخدمة في رسم خرائط سير العمليات.

الرمز	الحدث
	رمز إدخال المعطيات أو إخراج المعلومات بواسطة البطاقات CARDS (INPUT/OUTPUT)
	رمز الإدخال أو الإخراج بواسطة الشريط الورقي PAPER TAPE (I/0)
	رمز الإدخال أو الإخراج بواسطة وحدة الاتصال المرئية Terminal (I/0)
	رمز الإدخال أو الإخراج بواسطة الشريط المغناطيسي Magnetic Tape (I/0)
	رمز الإدخال أو الإخراج بواسطة القرص المغناطيسي Magnetic Disk (I/0)
	رمز توثيق printed document (output)
	إدخال أو إخراج البيانات عن بعد Communication Link
	رمز الإدخال / الإخراج بواسطة الاسطوانة الغناطيسية Magnitic Drum
	رمز الإدخال اليدوي للحاسب Manual Input
	رمز التعليق أو الإيضاح Remmarks
	رمز الفرز: عملية ترتيب مجموعة من العناصر وفقاً لمفتاح تعريفي خاص Sort
	رمز الدمج، دمج مجموعتين من العناصر أو أكثر للحصول على مجموعة جديدة Merge
	عكس عملية الضم حيث يتم فصل مجموعة من مجموعتين أو أكثر Extract

شكل (٩-٢)

(٣) اختيار لغة البرمجة:

الخطوة التالية لتخطيط الحل هي إختيار لغة البرمجة المناسبة إذ تختلف إمكانيات وأغراض لغات البرمجة المختلفة وقرار اختيار لغة البرمجة غالباً ما يتخذه محلل أنظمة.

(٤) كتابة البرنامج:

تلي خطوة اختيار لغة البرمجة خطوة كتابة البرنامج على وسط الإدخال المناسب، وغالباً ما تكون مهمة كتابة البرنامج من مهام المبرمج، ويعتمد المبرمج في كتابة البرنامج على خريطة سير العمليات (المخطط الانسيابي) التي يعدها محلل الأنظمة أو ربما يعتمد على مخطط انسيابي أكثر تفصيلاً يعده كبير المبرمجين أو المبرمج نفسه. والبرنامج في شكله الأخير يسمى **بالبرنامج المصدري Source program**

المرحلة الثانية: وتتمثل في دور الكمبيوتر نفسه في حل المسألة وتتكون من الخطوات الآتية:

(١) **إدخال (Enter) البرنامج المصدري إلى ذاكرة الكمبيوتر الرئيسة** عن طريق جهاز الإدخال المناسب لوسط الإدخال المسجل عليه البرنامج المصدري.

(٢) ترجمة البرنامج المصدري

يقوم برنامج الترجمة أو التجميع (Translator or Assembler program) بترجمة البرنامج المصدري إلى لغة الآلة الثنائية ويسمى البرنامج الجديد المترجم **بالبرنامج الهدفي** (Object Program).

وفي مرحلة الترجمة يتم:

(١) اكتشاف الأخطاء المعجمية واللغوية.

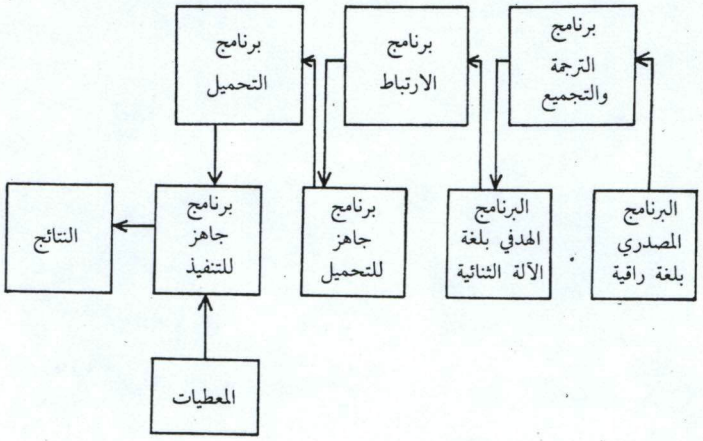
(٢) ترجمة البرنامج المصدري وتحويله إلى لغة الآلة.

(٣) يقوم أحد برامج التحكم (برنامج التوصيل) بالربط بين أجزاء البرنامج الهدفي ونخص بالذكر ربط البرنامج الرئيسي بالبرامج الفرعية المرتبطة به.

(٤) في حالة نجاح الخطوتين السابقتين (٢) و(٣) يقوم أحد برامج التحكم المعروف باسم برنامج التحميل (المحمل) (Loader) بفحص البرنامج الهدفي ونقله إلى موضعه في ذاكرة الكمبيوتر ليصبح البرنامج جاهزاً للتنفيذ.

(٥) تجربة البرنامج وتنفيذه (Test and execution) بعد الانتهاء من خطوة ترجمة البرنامج تبدأ مرحلة اختبار وتجربة البرنامج على عينة من البيانات التجريبية لمختلف العمليات التي يقوم بها البرنامج للتأكد من خلوه من الأخطاء المنطقية والرياضية والحسابية. . . الخ، فإذا ما ثبتت صحة طريقة الحل المقترحة يمكننا تنفيذ البرنامج على المعطيات الحقيقية (Actual data) وطباعة النتائج المطلوبة.

ويبين الشكل (٩-٣) خطوات المرحلة الثانية.

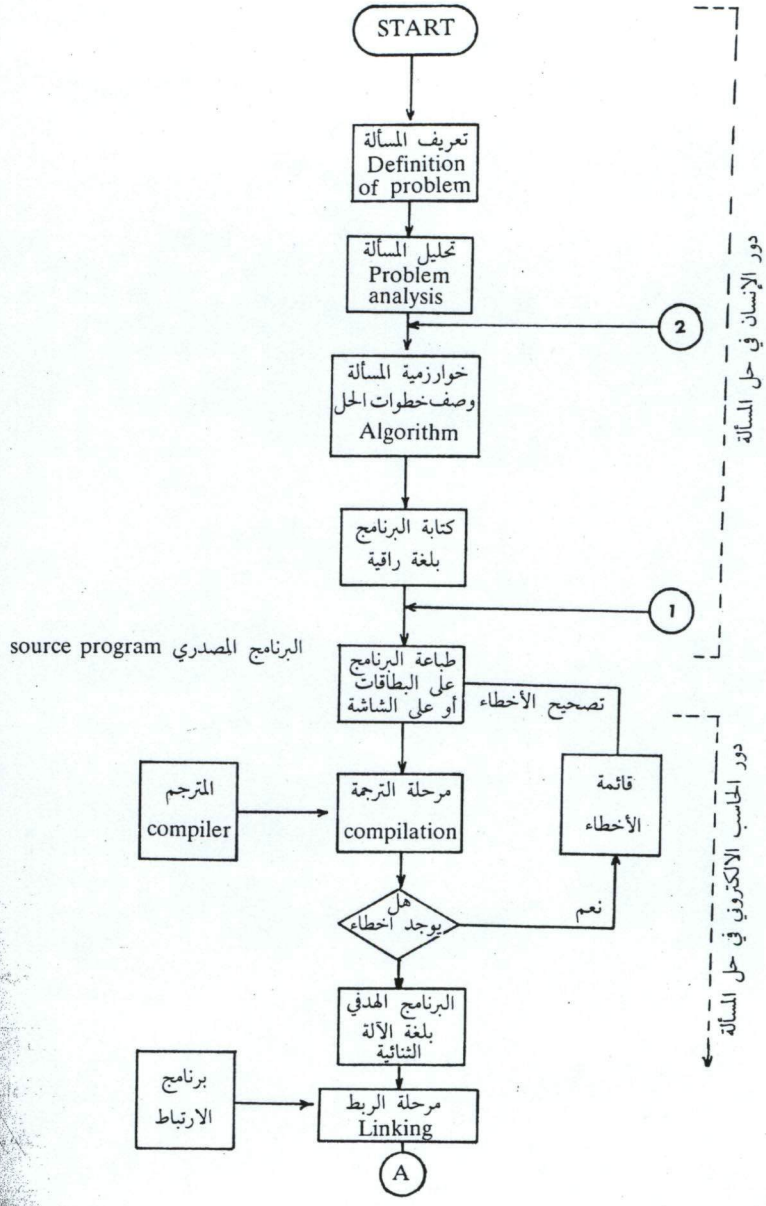


شكل (٩-٣)

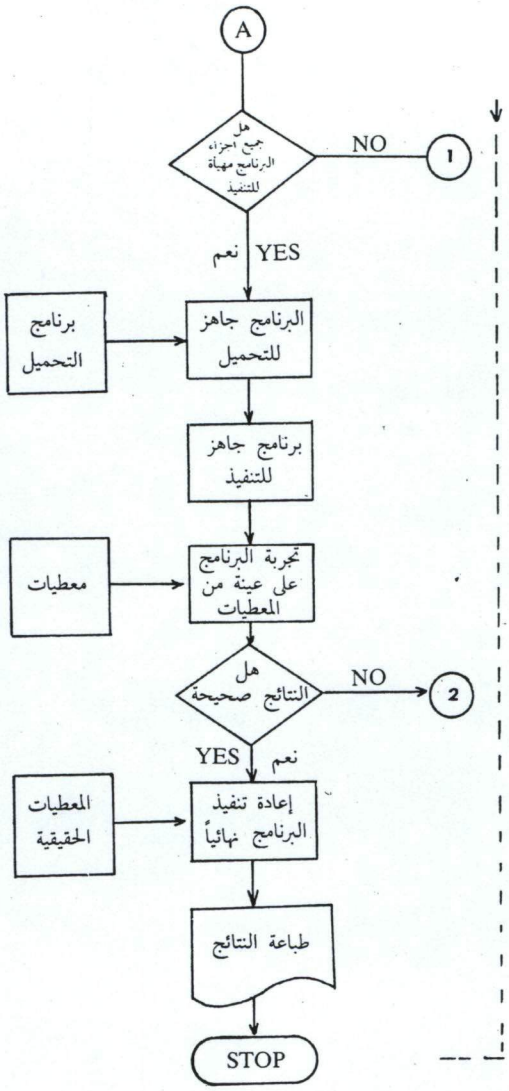
ويلخص الشكل (٩-٤) خطوات حل المسألة بالكمبيوتر.

المرحلة الثالثة: -

تسمى هذه المرحلة بمرحلة توثيق البرنامج حيث يقوم محلل الأنظمة مع المبرمج بإعداد دليل استخدام البرنامج ويشرح فيه خطوات عمل البرنامج وبيان أساليب إدخال المعلومات ونماذج من النتائج.


START
تعريف المسألة
Definition of problem
تحليل المسألة
Problem analysis
2
خوارزمية المسألة
وصف خطوات الحل
Algorithm
كتابة البرنامج
بلغة راقية
1
البرنامج المصدري source program
طباعة البرنامج
على البطاقات
أو على الشاشة
تصحيح الأخطاء
المترجم
compiler
مرحلة الترجمة
compilation
قائمة
الأخطاء
هل
يوجد أخطاء
نعم
البرنامج الهدفي
بلغة الآلة
الثنائية
برنامج
الارتباط
مرحلة الربط
Linking
A
دور الإنسان في حل المسألة
دور الحاسب الالكتروني في حل المسألة

شكل (٩-٤)


A
هل جميع أجزاء البرنامج مهيأة للتنفيذ
NO
1
نعم YES
برنامج التحميل
البرنامج جاهز للتحميل
برنامج جاهز للتنفيذ
معطيات
تجربة البرنامج على عينة من المعطيات
هل النتائج صحيحة
NO
2
YES نعم
المعطيات الحقيقية
إعادة تنفيذ البرنامج نهائياً
طباعة النتائج
STOP

تابع شكل (٩-٤)

[٩-٣] خرائط سير العمليات

(٩-٣:١) مفهوم خرائط سير العمليات:

الخوارزمية هي عبارة عن مجموعة من الخطوات المتسلسلة التي تصف بصورة مضبوطة وبدون أي غموض جميع الخطوات الرياضية والمنطقية اللازمة لحل مسألة ما. ولكن هذا الوصف في كثير من الأحيان يكون معقداً وصعب الملاحظة والتتبع لذلك فإن خريطة سير العمليات التي تمثل وصفاً تصويرياً لخطوات الخوارزمية تكون أكثر وضوحاً. وخريطة سير العمليات تقوم مقام الخوارزمية ويمكن بواسطتها ملاحظة تتبع التسلل المنطقي لحل المسألة بكل سهولة، وغالباً ما يكون استخراج الخوارزمية من خريطة سير العمليات أسهل بكثير من كتابة الخوارزمية مباشرة.

(٩-٣:٢) أنواع خرائط سير العمليات

تنقسم خرائط سير العمليات بشكل عام إلى قسمين رئيسيين هما:

(أ) خرائط النظم (Systems Flowcharts):

وهذا النوع يحدد بالرسوم العلاقات المنطقية التي تربط بين مجموعة العمليات المتتالية والمترابطة للنظام المطلوب ويتكون من عدد من الخرائط مهمة كل منها توضيح وظيفة أو مرحلة معينة داخل النظام بحيث توضح بعض الأمور الآتية:

١ ـ سير المعطيات من حيث طريقة نقلها وتغذية الحاسب بها وكذلك تناولها ومعالجتها.
٢ ـ طريقة ربط برامج النظام بعضها ببعض.
٣ ـ العلاقة بين الدوائر المستخدمة للحاسب والنظام المقترح.
٤ ـ تحديد شكل وطبيعة النتائج الخاصة بكل برنامج.

ومن الأمثلة على خرائط النظم ما تستخدمه بعض الشركات أو المؤسسات لأنظمتها المختلفة.

(ب) خرائط سير البرامج (Programs Flowcharts):

وهي تحدد بالرسم العلاقة المنطقية، والتسلسل العام بين مجموعة من الأحداث والعمليات المترابطة والمتتالية التي يتكون منها البرنامج.

(٩-٣:٣) أهمية خرائط سير العمليات

١ - تساعد المبرمج على الإلمام الكامل بالمسألة المراد حلها، والسيطرة على كل أجزائها بحيث تساعده على اكتشاف الأخطاء المنطقية (logical errors) والتي غالباً ما تجهد المبرمج.

٢ - في المسائل ذات الأوضاع المختلفة، والتي تكثر فيها التفرعات يصعب على المبرمج تذكرها ومتابعتها أثناء كتابة البرنامج بدون الاعتماد على خرائط سير العمليات.

٣ - تساعد المبرمج على تعديل البرامج الموضوعة بمجرد النظر مما يزيد في فاعلية هذه البرامج.

٤ - يعتبر الاحتفاظ بخرائط سير العمليات لحل مسألة (مسائل) معينة مرجعاً هاماً يمكن الرجوع إليها عند إجراء تعديلات عليها أو استخدامها لحل مسائل أخرى مشابهة دون التقيد بوجود المبرمج الذي قام بحل المسألة باعتبار أن الحل / الحلول الأولى قد تمت صياغتها في خطوات واضحة وبسيطة ومفهومة يسهل متابعتها وفهمها من قبل مبرمج آخر.

[٩-٤] خرائط سير البرامج

تمثل خرائط سير العمليات في البرنامج وصفاً تصويرياً لخطوات حل المسألة (الخوارزمية) بشكل أكثر وضوحاً وأنسب فهماً باستخدام مجموعة من رموز الأشكال الاصطلاحية المبينة في شكل (٩-١)، هذا ويمكن تصنيف خرائط سير البرامج على النحو التالي:

١ ـ خرائط التتابع البسيط (Simple sequential Flowcharts).

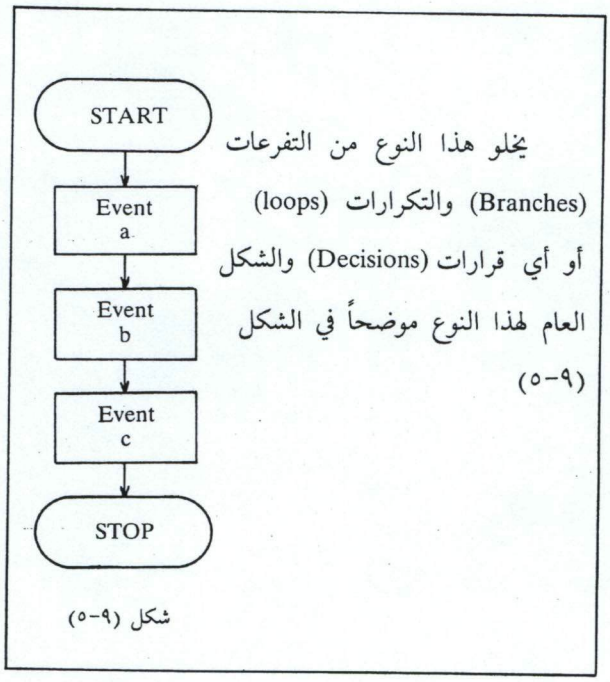
٢ ـ خرائط التفرع (Branched Flowcharts).

٣ ـ خرائط التكرار البسيط (Loop Flowcharts).

٤ ـ خرائط التكرار المتداخلة (Nested-loop-flowcharts).

ويمكن للبرنامج الواحد أن يشتمل على أكثر من نوع واحد من هذه الأنواع. وسنتناول فيما يأتي شرح هذه الأنواع بشيء من التفصيل.

(٩-٤:١) خرائط التتابع البسيط *(Simple Sequential Flowcharts)*:



شكل (٩-٥)

يخلو هذا النوع من التفرعات (Branches) والتكرارات (loops) أو أي قرارات (Decisions) والشكل العام لهذا النوع موضحاً في الشكل (٩-٥)

وكلمة Event الواردة في شكل (٩-٥) تعني الحدث أو العملية المطلوب تنفيذها.

مثال (٩-١): اكتب خريطة سير برنامج لإيجاد مجموع ثلاثة أعداد معطاة .

الحـــل:

عناصر المدخلات
A: العدد الأول
B: العدد الثاني
C: العدد الثالث

عناصر المخرجات: S

العلاقات: $S = A+B+C$

خطوات الحل كما هي مبينة في الشكل (٩-٦) هي:

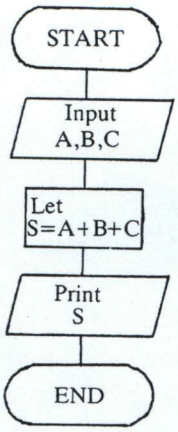
١ ـ ابدأ

٢ ـ اقرأ قيم الأعداد B,A و C

٣ ـ احسب قيمة S طبقاً للمعادلة: $S=A+B+C$

٤ ـ إطبع قيمة S (النتيجة)

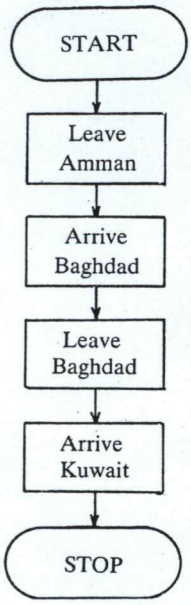
٥ ـ توقف



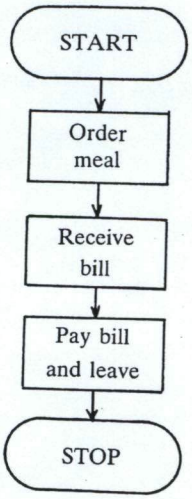
شكل (٩-٦)

مثال (٩-٢): لو أراد شخص أن يسافر براً من عمان إلى الكويت، ارسم خريطة سيره موضحاً العواصم العربية التي يمر بها.

الشكل (٩-٧) يوضح خريطة سير العمليات المطلوبة والممثلة في الخطوات الآتية:



شكل (٩-٧)



شكل (٩-٨)

١ ـ ابدأ الرحلة
٢ ـ غادر عمان
٣ ـ صل بغداد
٤ ـ غادر بغداد
٥ ـ صل الكويت
٦ ـ توقف.

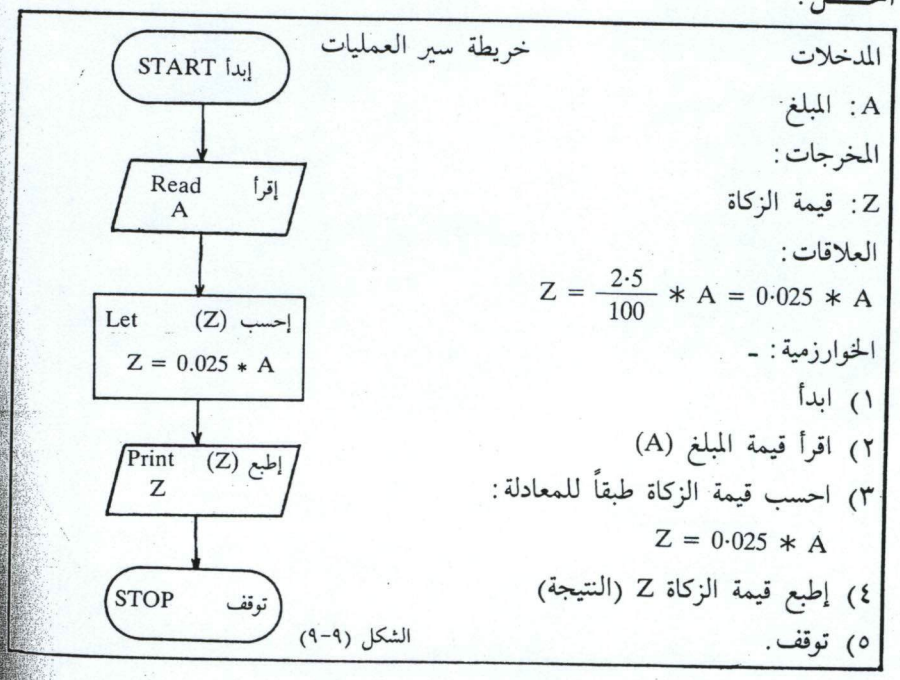
مثال (٩-٣): ارسم خريطة سير العمليات التي تمثل إجراءات تناول وجبة غذائية في مطعم.

الحـل: خريطة سير العمليات في الشكل (٩-٨) تمثل الخطوات الآتية:

١ ـ ابدأ
٢ ـ اطلب الوجبة
٣ ـ استلم الفاتورة
٤ ـ ادفع الفاتورة وغادر
٥ ـ توقف.

مثال (٩-٤) إكتب خريطة سير العمليات لإيجاد قيمة الزكاة مبلغ بلغ النصاب الشرعي بعد مرور عام عليه.

الحـــل:



الشكل (٩-٩)

مثال (٩-٥): ارسم خريطة سير العمليات لإيجاد مساحة ومحيط دائرة نصف قطرها معلوم (R).

الحـــل:

المتغيرات

R نصف القطر

A مساحة دائرة نصف قطرها (R) $= \pi R^2$

C محيط دائرة نصف قطرها (R) $= 2\pi R$

حيث $\pi=3.14$ (النسبة التقريبية)

وتكون خطوات الحل المبينة في شكل (٩-١٠) كما يلي

١ ـ ابدأ

٢ ـ اقرأ قيمة R

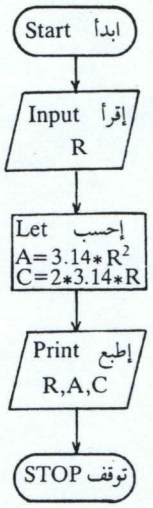
٣ ـ اجعل قيمة PIE = 3.14

٤ ـ احسب المساحة (A) من المعادلة $A = \pi R^2$

٥ ـ احسب المحيط (C) من المعادلة $C = 2\pi R$

٦ ـ اطبع قيم كل من C,A,R

٧ ـ توقف.



شكل (٩-١٠)

مثال (٩-٦): ارسم خريطة سير العمليات لحساب قيمة كل من المتغيرات C,B,A في المعادلة الآتية:

$$A = X^2 + y^2 \ldots (1)$$

$$B = X + 3A \ldots (2)$$

$$C = XY + A - B^2 \ (3)$$

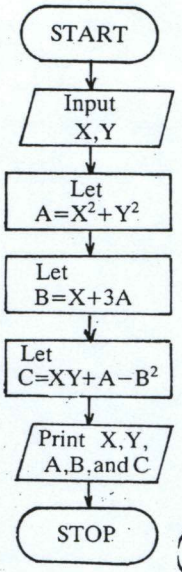
حيث أن X و Y قيم معلومة

الحـــل:

المدخلات في المسألة هي X و Y والمطلوب حساب قيمة المتغير A في المعادلة

(1)، وحساب قيمة المتغير B في المعادلة (2) بالاعتماد على قيمة X المعلومة لدينا وقيمة المتغير A المحسوبة في الخطوة السابقة، أما قيمة المتغير C فيتم حسابها في المعادلة (3) بالاعتماد على قيم كل من المتغيرات B,A,Y,X وكلها معلومة.

وتكون خطوات حل المسألة كما هو مبين في الشكل (٩-١١) كما يأتي:



الشكل (٩-١١)

١ ـ ابدأ

٢ ـ اقرأ قيم كل من Y,X

٣ ـ احسب قيمة A من المعادلة (1)

٤ ـ احسب قيمة B من المعادلة (2)

٥ ـ احسب قيمة C من المعادلة (3)

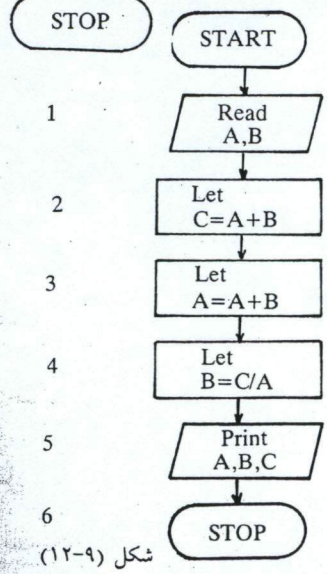
٦ ـ اطبع قيم كل من C, B, A, Y, X

٧ ـ توقف.

مثال (٩-٧): الشكل (٩-١٢) يمثل خريطة سير العمليات لمجموعة من العمليات الحسابية، ادرس ناتج العمليات بعد تتبع الخريطة إذا علمت أن:

A = 10

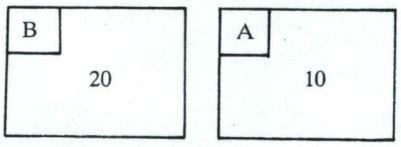
B = 20



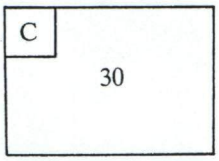
شكل (٩-١٢)

الحـــل:

عند تتبع خريطة سير العمليات في الشكل (٩-١٢) نلاحظ أن الخطوة الأولى تمثل قراءة قيم المتغيرات A و B وبالتالي تخصص القيم في ذاكرة الحاسب على النحو التالي:

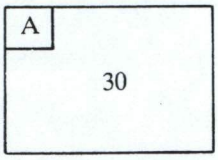


أمّا الخطوة الثانية فهي معالجة وتتلخص بحساب قيمة المتغير C وفقاً للمعادلة:



$$C = A + B$$
$$= 10 + 20$$
$$= 30$$

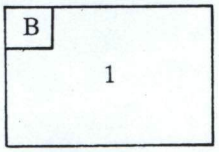
أما الخطوة الثالثة فهي أيضاً معالجة وتتلخص بحساب قيمة A وفقاً للمعادلة:



$$A = A + B$$
$$= 10 + 20$$
$$= 30$$

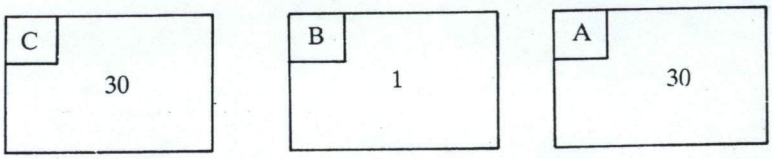
نلاحظ في هذه الخطوة أن قيمة المتغير A الجديدة هي قيمة المتغير A القديمة مضافاً إليها قيمة المتغير B وبالتالي فإن قيمة المخزن A تصبح تساوي 30 بدلاً من 10 وذلك بإلغاء القيمة السابقة واستبدالها بالقيمة الجديدة فقط.

أما الخطوة الرابعة فهي تعني تغير قيمة المتغير B لتصبح:



$$B = C/A$$
$$= 30/30$$
$$= 1$$

لاحظ أننا أخذنا آخر قيمة للمتغير A لحساب قيمة B. وبالتالي فإن القيمة النهائية للمتغيرات C,B,A تكون على النحو التالي:



أما الخطوة الخامسة فهي طباعة قيم المغيرات C,B,A أي أن:

$$A = 30$$
$$B = 1$$
$$C = 30$$

أما الخطوة السادسة فهي توقّف.

ويمكن تلخيص خطوات تنفيذ خريطة سير العمليات في المثال (٩-٧) بالجدول التالي:

رقم الخطوة	A	B	C	المخرجات
1	10	20	-	
2	10	20	30	
3	30	20	30	
4	30	1	30	
5				30 1 30
6	STOP			

ونلاحظ أن كل خطوة تمثل وضع الحاسب في تلك اللحظة.

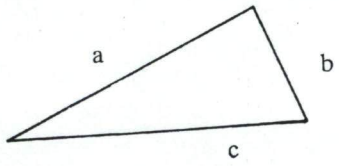
تمارين للتدريب

(١) اكتب خوارزمية وخريطة سير العمليات لإيجاد الوسط الحسابي للأعداد X، Y و Z.

(٢) اكتب خريطة سير العمليات لحساب قيمة Y من المعادلة:

$$Y = X^2 + ZX + 14$$

(٣) اكتب خوارزمية وخريطة سير العمليات لإيجاد مساحة مثلث أطوال أضلاعه الثلاثة معلومة



$$A = \sqrt{s\,(s-a)\,(s-b)\,(s-c)}$$

$$S = (A + B + C) / 2$$

حيث:

(٤) اكتب خوارزمية وخريطة سير العمليات لقراءة a و t وحساب v و d طبقاً للمعادلات الآتية:

$$d = \frac{1}{2}\,at^2$$

$$v = at$$

(٥) اكتب خوارزمية وخريطة سير العمليات لتحويل قياس درجات الحرارة من النظام المئوي إلى النظام الفهرنهايتي حسب القاعدة:

$$F^o = \left(\frac{9}{5}\right) C^o + 32$$

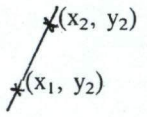
(٦) اكتب خوارزمية وخريطة سير العمليات لإيجاد مساحة وحجم كرة معلومة باستخدام العلاقات الآتية:

$$V = \frac{4}{3}\,\pi r^2$$

$$A = 4\,\pi\,r^2$$

حيث r تمثل نصف القطر.

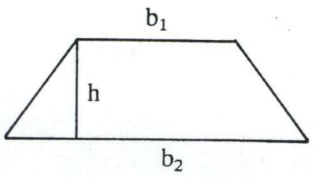
(٧) اكتب خوارزمية وخريطة سير العمليات لإيجاد ميل خط مستقيم يمر من خلال نقطتين معلومتين حيث:



$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

(٨) اكتب خوارزمية وخريطة سير العمليات لحساب طول وتر مثلث قائم الزاوية.

(٩) اكتب خوارزمية وخريطة سير العمليات لحساب مساحة شبه المنحرف التالي:



حيث أن مساحة شبه المنحرف (A) هي:

$$A = \frac{(b_1 + b_2)}{2} * h$$

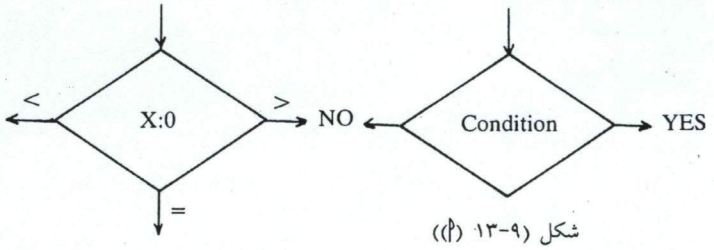
(١٠) توفي رجل وترك مبلغاً قيمته x وترك عدداً من الأبناء (N) وعدداً من البنات (M) وزوجة، إكتب خوارزمية وخريطة سير العمليات لتوزيع الميراث على الورثة طبقاً للقواعد التالية:

أ) الزوجة لها ثمن ما ترك

ب) للذكر مثل حظ الانثيين

(٩-٤:٢) خرائط الاختيار

هذا النوع من الخرائط يتضمن اتخاذ قرار أو مفاضلة بين اختيارين أو أكثر، وهناك أسلوبان في تنفيذ القرار (انظر شكل (٩-١٣)).

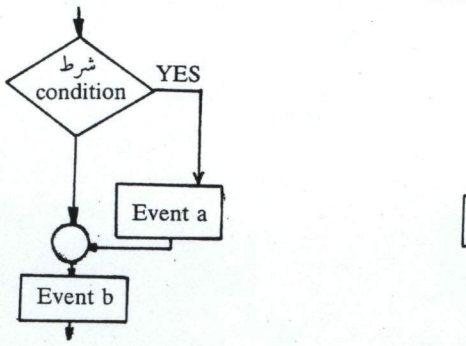


شكل (٩-١٣ (أ))
قرار ذو تفرعين

شكل (٩-١٣ (ب))
قرار ذو ثلاثة تفرعات

الشكل (٩-١٣)

وبشكل عام فإن خرائط التشعيب يمكن أن تأخذ إحدى الصورتين الآتيتين (انظر شكل (٩-١٤)).



شكل (٩-١٤ (أ))

إذا كان جواب الشرط (YES)
فإن الحدث التالي في التنفيذ هو (a)، أما إذا كان
الجواب (NO)
فإن الحدث التالي في التنفيذ هو (b)
ويعرف هذا الشكل بصيغة إذا المزدوجة أي :
IF - THEN - ELSE

شكل (٩-١٤ (ب))

إذا كان جواب الشرط (YES)
فإن الحدث التالي في التنفيذ هو الحدث(a) ثم يتبعه
الحدث (b) أما إذا كان الجواب (NO)
فإن الحدث التالي في التنفيذ هو (b) مباشرة
ويعرف هذا الشكل بصيغة إذا البسيطة أي :
IF - THEN - statement

شكل (٩-١٤)

أمثلة على الصيغة- IF - THEN - ELSE

مثال (٩-٨): لو أردت السفر برّاً من عمان إلى الكويت مارّاً ببغداد إن كان الطريق مفتوحاً وإلا فعن طريق الرياض، ارسم خريطة سير الرحلة.

خطوات الحل المبينة في الشكل (٩-١٥) هي:

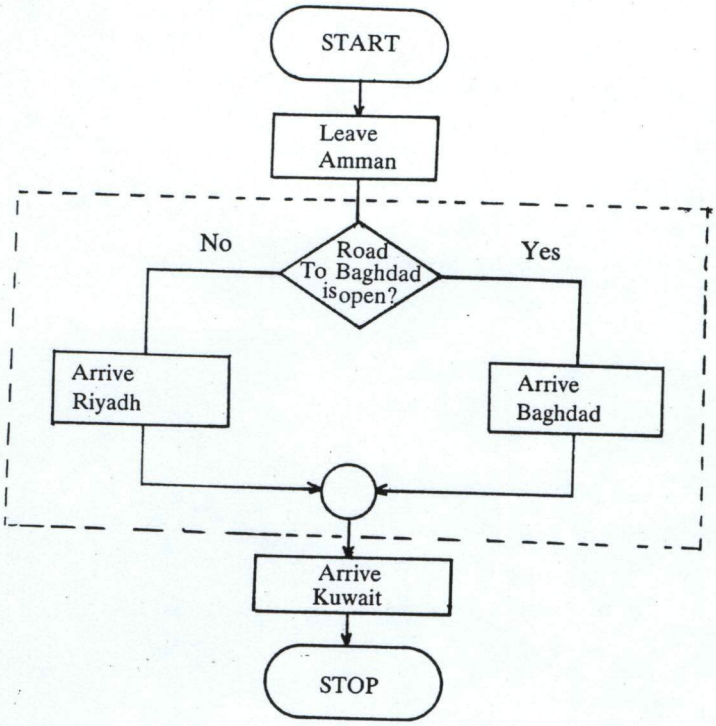
١ ـ ابدأ

٢ ـ إذا كان الطريق إلى بغداد مفتوحاً:
سافر إلى بغداد
وإلا
سافر إلى الرياض

جملة إذا المزدوجة

٣ ـ سافر إلى الكويت

٤ ـ توقف



شكل (٩-١٥)

مثال (٩-٩) ارسم خريطة سير العمليات لإيجاد قيمة اقتران القيمة المطلقة المعرف رياضياً حسب القاعدة التالية:

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

خطوات الحل المبينة في شكل (٩-١٦) تكون:

١ ـ ابدأ

٢ ـ اقرأ قيمة المتغير x

٣ ـ إذا كانت x أكبر من أو تساوي صفراً إحسب قيمة الاقتران طبقاً للقاعدة:

$$F(x) = x$$

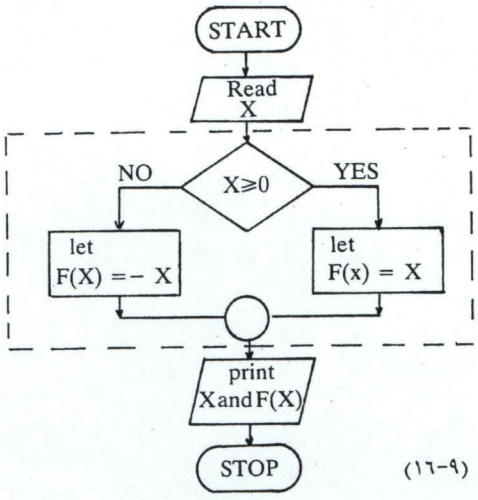
وإلا إحسب قيمة الاقتران طبقاً للقاعدة:

$$F(x) = -x$$

٤ ـ إطبع قيمة كل من:

x و F(x)

٥ ـ توقف



شكل (٩-١٦)

مثال (٩-١٠): ارسم خريطة سير العمليات لقراءة قيمتين معلومتين واختيار أكبرهما.

الحـــل:

المدخلات: A القيمة الأولى

B القيمة الثانية

المخرجات: MAX القيمة العظمى

العلاقات: MAX = max {A,B}

أي أن:

إذا كانت A>B

فإن

MAX = A

وإلا

MAX = B

وعليه تكون خطوات الحل كما هو مبين في الشكل (٩-١٧) كما يأتي:

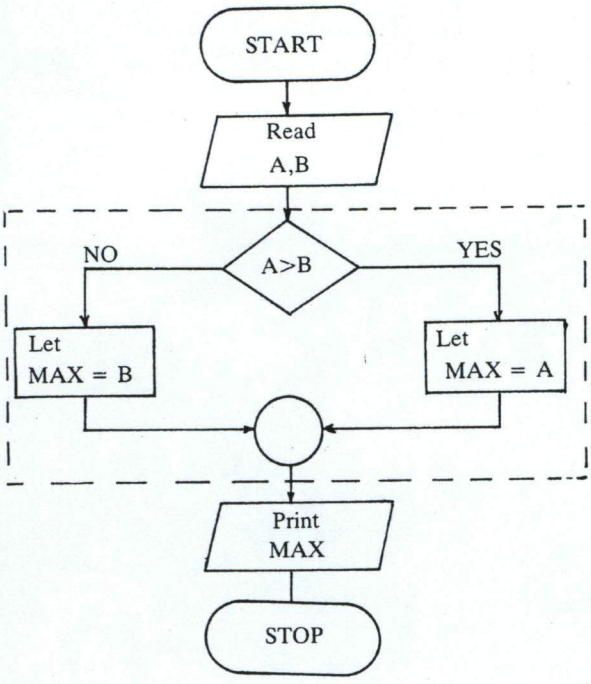
١ ـ ابدأ

٢ ـ اقرأ قيم كل من B,A

٣ ـ إذا كانت A>B
اجعل MAX = A
وإلا اجعل MAX = B

٤ ـ اطبع MAX (النتيجة)

٥ ـ توقف.



شكل (٩-١٧)

أمثلة على صيغة إذا البسيطة IF - THEN - statement

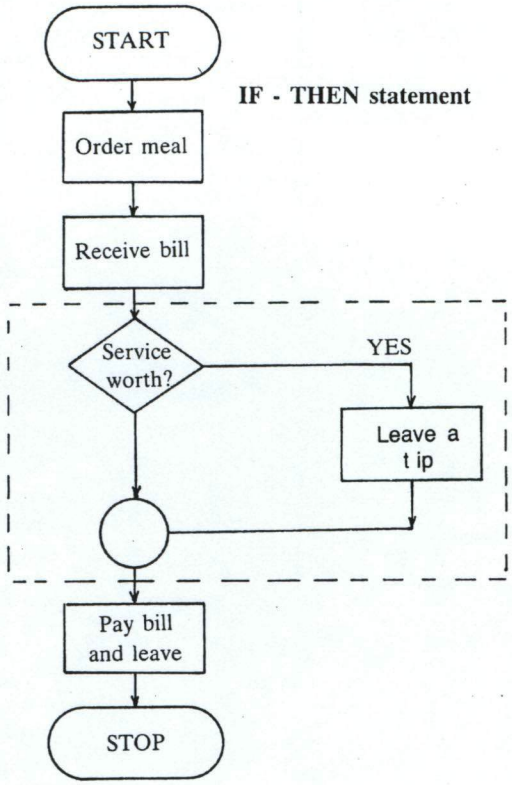
مثال (٩-١١) في المثال (٩-٣) إذا أراد الشخص أن يدفع للعامل هبة (بخشيشاً) بمعنى (tip) فإن ذلك يعتمد على طبيعة الخدمة التي قدمها له العامل وبالتالي تعدل خريطة سير العمليات لتأخذ الشكل رقم (٩-١٨).

وخطوات الحل المبينة في الشكل (٩-٨) هي:

١ ـ ابدأ

٢ ـ اطلب الوجبة

٣ ـ استلم الفاتورة



IF - THEN statement

٤ ـ إذا أعجبتك الخدمة اترك هبة (بخشيشاً) للعامل

٥ ـ ادفع الفاتورة وغادر

٦ ـ توقف.

مثال (٩-١٢): أعد كتابة خريطة سير العمليات في الشكل (٩-١٧) باستخدام الصيغة المبينة في الشكل (٩-١٤ (ب))

الحل: تكون خطوات الحل المبينة في الشكل (٩-١٩) كما يأتي:

١ ـ ابدأ

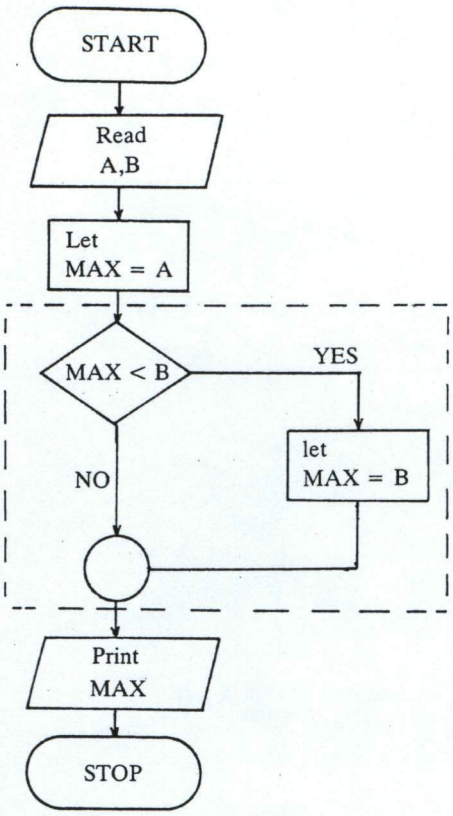
٢ ـ اقرأ قيمة B,A

٣ ـ اجعل MAX = A

٤ ـ إذا كانت MAX < B اجعل MAX = B

٥ ـ اطبع MAX

٦ ـ توقف.



شكل (٩-١٩)

مثال (٩-١٣): اكتب خريطة سير العمليات لإعادة ترتيب قيمتين ترتيباً تصاعدياً.

الحـــل:

تكون خطوات الحل كما هو مبين في الشكل (٩-٢٠) كما يأتي:

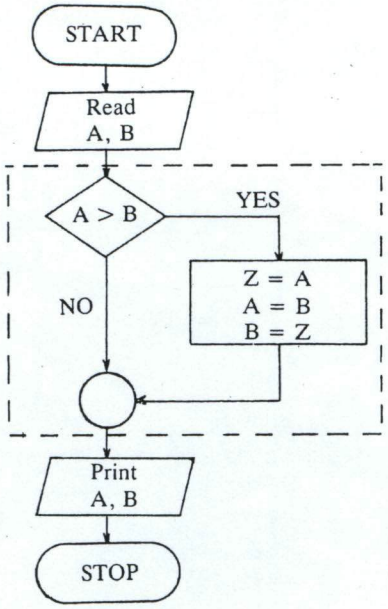
١ ـ ابدأ

٢ ـ اقرأ القيمتين B,A

٣ ـ إذا كانت A > B
استبدل (SWAP) كل من A و B

٤ ـ اطبع A و B

٥ ـ توقف.



شكل (٩-٢٠)

أمثلة على صيغ إذا المتداخلةNested IF statements

مثال (٩-١٤): يعطي أحد البائعين خصماً على مبيعاته وفقاً لما يلي:

١) 20% إذا كانت طلبية العميل 500 وحدة فأكثر.

٢) 15% إذا كانت طلبية العميل 250 إلى أقل من 500 وحدة.

٣) لا يوجد خصم للطلبيات التي تكون أقل من 250 وحدة.

المطلوب تحديد خطوات حساب قيمة الطلبية إذا علمنا سعر بيع الوحدة وحجم الطلبية.

مدخلات:
- Q: حجم الطلبية
- P: سعر بيع الوحدة

مخرجات:
- D: خصم الكمية (٪)
- S: قيمة الطلبية

خطوات حساب قيمة الطلبية المبينة في الشكل (٩-٢١) هي:

١ ـ ابدأ

٢ ـ ادخل قيمة كل من P و Q

٣ ـ إذا كانت $Q \geq 500$

احسب قيمة D وفقاً للمعادلة:

$$D = \frac{20}{100}$$

وإلا

إذا كانت $Q \geq 250$

احسب قيمة D وفقاً للمعادلة:

$$D = \frac{15}{100}$$

وإلا

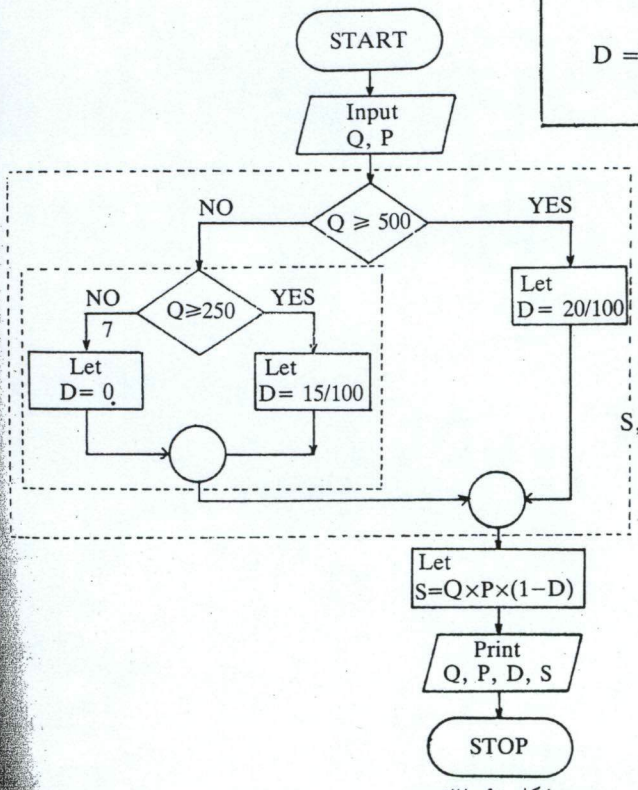
إجعل $D = 0$.

٤ ـ احسب قيمة الطلبية وفقاً للمعادلة:

$$S = Q \times P\ (1-D)$$

٥ ـ اطبع قيمة S, D, P, Q

٦ ـ توقف



شكل (٩-٢١)

مثال (٩-١٥): اذرس ناتج العمليات في المثال (٩-١٤) إذا علمت أن:

Q = 1000 unit

P = 10 J.D / unit

يمكن تلخيص خطوات الحل في الجدول التالي:

الخطوة	Q	P	D	S	Q≥500	Q≥250	المخرجات
2	100	10	-	-	-	-	
3					YES		
	1000	10	.2		-		
4	1000	10	.2	8000			
5							1000 10,.2 8000
6	STOP						

أي أن:

A = 1000

P = 10

D = .2

S = 8000

مثال (٩-١٦) ارسم خريطة سير العمليات لحساب قيمة Y طبقاً للمعادلات الآتية:

$$Y = \begin{cases} x^2 + 1 & , x > 0 \ldots (1) \\ 5 & , x = 0 \ldots (2) \\ 2x + 5 & , x < 0 \ldots (3) \end{cases}$$

خوارزمية الحل هي:

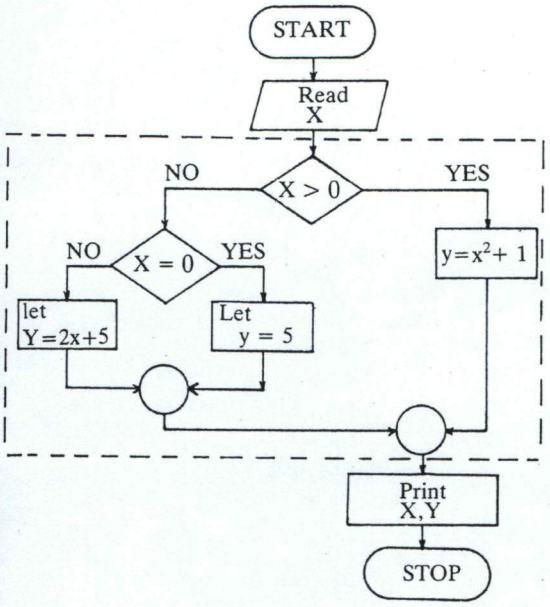
١ ـ ابدأ.

٢ ـ اقرأ قيمة المتغير X.

٣ ـ إذا كانت X أكبر من صفر
احسب Y من المعادلة (1)
وإلا
إذا كانت X تساوي صفراً
احسب Y من المعادلة (2)
وإلا
احسب Y من المعادلة (3).

٤ ـ اطبع قيمة كل من X و Y

٥ ـ توقف.



الشكل (٩-٢٢)

مثال (٩-١٧) : ارسم خريطة سير العمليات لحساب قيمة الاقتران F (X, Y) المعرف حسب القاعدة:

$$F(X,y) = \begin{cases} X + Y & \text{و} \quad X > Y \\ X^2 + 5 & \text{و} \quad X = Y \\ Y + 2X & \text{و} \quad X < Y \end{cases}$$

إذا علمت أن قيم كل من X و Y معلومة.

الحـــل:

يمكن إعادة تعريف الاقتران على النحو التالي:

$$F(X\ Y) = \begin{cases} X + Y & \text{و} \quad X - Y > 0 - (1) \\ X^2 + 5 & \text{و} \quad X - Y = 0 - (2) \\ Y + 2X & \text{و} \quad X - Y < 0 - (3) \end{cases}$$

وبالتالي تكون خطوات الحل كما هي مبينة في الشكل (٩-٢٣) هي:

١ ـ ابدأ

٢ ـ ادخل قيم كل من X و Y

٣ ـ اجعل قيمة D = X − Y

٤ ـ احسب إذا كانت قيمة D أكبر من صفر إنتقل إلى الخطوة ٥
وإذا كانت قيمة D تساوي صفراً انتقل إلى الخطوة ٦
أما إذا كانت D أقل من صفر فانتقل إلى الخطوة ٧

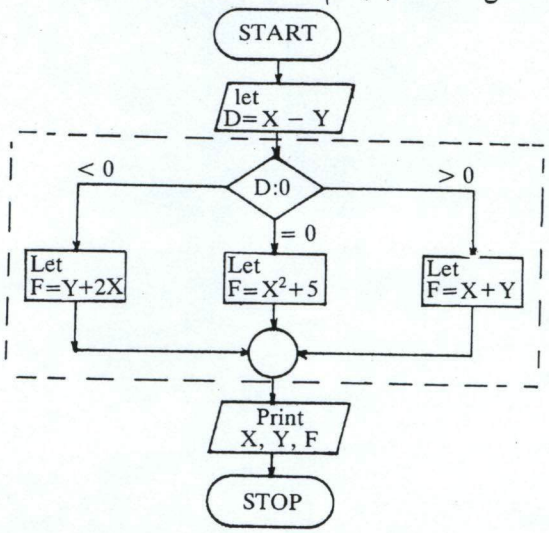
٥ ـ احسب قيمة الاقتران F (X,Y) ثم اذهب إلى الخطوة ٨

٦ ـ احسب قيمة الاقتران F (X,Y) من المعادلة (٢) ثم اذهب إلى الخطوة ٨

٧ ـ احسب قيمة الاقتران F (X,Y) من المعادلة (٣) ثم اذهب إلى الخطوة ٨

٨ ـ اطبع قيمة كل من X و F

٩ ـ توقف.



شكل (٩-٢٣)

تمارين للتدريب

(١) اكتب خوارزمية وخريطة سير العمليات لقراءة قيمتين وإختيار أصغرهما.

MIN = min { A, B}

(٢) يأخذ بائع عمولة مقدارها 20% من قيمة مبيعاته إذا كانت قيمة المبيعات تزيد عن 10000 دينار، اكتب خوارزمية وخريطة سير العمليات لإيجاد مجمل دخل البائع.
حيث مجمل الدخل = الراتب + العمولة.

(٣) اكتب خوارزمية وارسم خريطة سير العمليات لاختبار قابلية القسمة على 2 .

(٤) اكتب خوارزمية وخريطة سير العمليات لإشارة السير الضوئية

(٥) اكتب خوارزمية وارسم خريطة سير العمليات لاختيار القيمة العظمى لثلاثة قيم معلومة.

(٦) اكتب خوارزمية وخريطة سير العمليات لإيجاد قيمة الاقتران F (X) المعرف على النحو التالي:

$$F(X) = \begin{cases} X^2 & ; X < 1 \\ 1 & ; X = 1 \\ X & ; X > 1 \end{cases}$$

(٧) توفر جامعة ما سكناً لأعضاء هيئة التدريس فيها مقابل أجور كما هي مبينة في الجدول التالي:

نوع السكن	الأجرة بالدينار
سكن أ	120 ديناراً
سكن ب	75 ديناراً
سكن جـ	60 ديناراً
سكن د	50 ديناراً

اكتب خوارزمية وارسم خريطة سير العمليات لقراءة اسم الموظف ورقمه ونوع سكنه وراتبه الأساسي وحساب مقدار صافي الراتب بعد اقتطاع أجرة السكن.

(٨) اكتب خوارزمية وارسم خريطة سير العمليات لإيجاد قيمة الاقتران المعرف على النحو التالي:

$$y = \begin{cases} a + bx & \text{if } x = -1 \\ a + bx^2 & \text{if } x = 0 \\ a + bx + 5 & \text{if } x = 1 \end{cases}$$

(٩) اكتب خوارزمية وخريطة سير العمليات لإيجاد المعادلة التربيعية

$$ax^2 + bx + c = 0$$

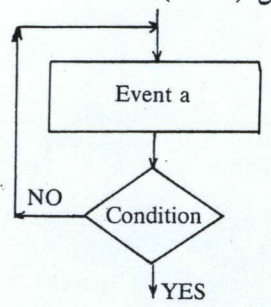
(١٠) إذا كانت العمولة التي يأخذها البائع على مجمل مبيعاته تحسب وفق الجدول التالي:

المبيعـــات	العمولة
إلى أقل من 500 دينار	2%
من 500 دينار إلى أقل من 1000 دينار	5%
1000 دينار أو أكثر	10%

اكتب خوارزمية وخريطة سير العمليات لإيجاد مقدار العمولة بالدينار التي يأخذها البائع.

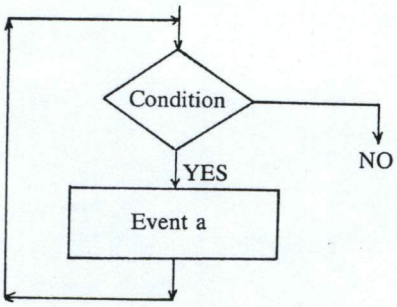
(٩-٤:٣) خرائط التكرار البسيط *(Loop flowchart)*

هذا النوع من الخرائط نحتاج إليها لإعادة تنفيذ مجموعة من الجمل في البرنامج عدداً محدوداً أو غير محدود من المرات، والشكل العام لمثل هذا النوع من الخرائط مبين في الشكل (٩-٢٤):



شكل (٩-٢٤ (أ))

يتكرر تنفيذ الحدث (a) عدداً من المرات إلى أن يصبح جواب الشرط (YES)
(DO...UNTIL)



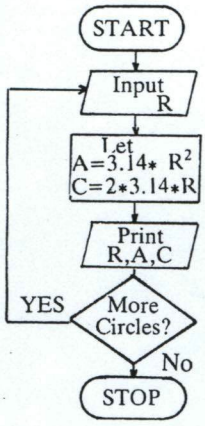
شكل (٩-٢٤ (ب))

طالما كان جواب الشرط YES كرر تنفيذ الحدث (a)
(DO...WHILE)

الشكل (٩-٢٤)

مثال (٩-١٨) اكتب خوارزمية وارسم خريطة سير العمليات لإيجاد مساحة ومحيط مجموعة من الدوائر أنصاف أقطارها معلومة:

تكون خطوات الحل المبينة في الشكل (٩-٢٥) كما يأتي:



شكل (٩-٢٥)

١ ـ ابدأ

٢ ـ إقرأ نصف قطر الدائرة (R)

٣ ـ أوجد مساحة الدائرة (A) ومحيط الدائرة (C)

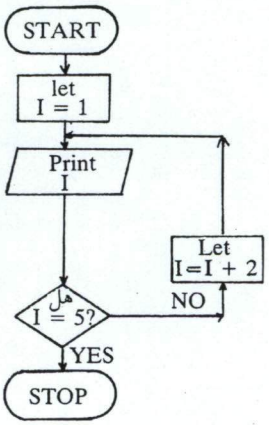
٤ ـ اطبع قيم كل من C, A, R

٥ ـ هل هناك مزيد من الدوائر؟

إن كان الجواب نعم كرر تنفيذ الخطوات من (٢) إلى (٤) بمعنى آخر عُد إلى الخطوة ـ (٢).

٦ ـ توقف.

مثال (٩ ـ ١٩) ادرس ناتج تنفيذ العمليات بعد تتبع أحداث خريطة سير العمليات المبينة في الشكل (٩ ـ ٢٦) .

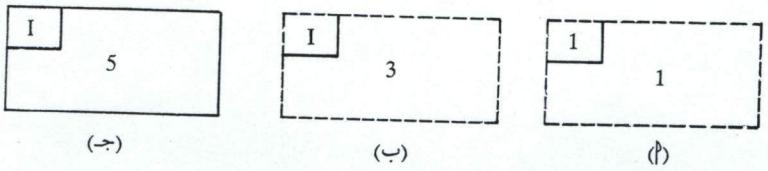


(شكل (٩-٢٦)

يلاحظ من الشكل (٩-٢٦) أن الكمبيوتر يتبع الخطوات التالية:

يبدأ الكمبيوتر بوضع قيمة مبدئية أولى مقدارها 1 في مخزن الذاكرة I كما في

الشكل (٩ ـ ٢٧ (أ)). (٢) يقوم الكمبيوتر بطبع هذه القيمة على الوسط الخارجي (٣) يسأل الكمبيوتر «هل القيمة المخزونة I تساوي 5 » ؟



(أ) (ب) (جـ)

الشكل (٩-٢٧)

(٤) الجواب بالطبع لا (NO) لأن $I \neq 5$. لهذا ينفذ الكمبيوتر الأمر التالي: $I = I + 2$، وهذا يعني أن الكمبيوتر سيضع في المخزن I ما كان فيه سابقاً مضافاً إليه 2 لتصبح القيمة المخزونة في المخزن I تساوي 3 بدلاً من 1 (انظر شكل (٩-٢٧ (ب)) (٥) يعود الكمبيوتر مرة أخرى بعد أن يطبع قيمة I الجديدة على الوسط الخارجي ليسأل هل «$I = 5$»؟

ويتكرر تنفيذ العملية السابقة حتى تصبح القيمة المخزونة في المخزن I تساوي 5، عندما يتوقف البرنامج بالأمر STOP وبذلك يكون قد طبع لنا على الوسط الخارجي القيم التي خزنت في المخزن I وهي على التوالي:

1

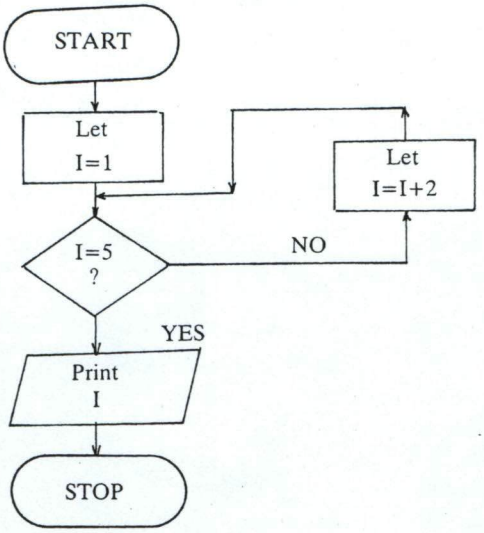
3

5

ملحوظة: هناك مخزن واحد فقط تحت اسم I في وحدة الذاكرة تخزن فيه قيمة واحدة في الوقت الواحد ولذا فإن آخر قيمة تبقى في المخزن I في هذا المثال هي 5.

مثال (٩-٢٠) لو أحدثنا تغييراً بسيطاً في الشكل (٩-٢٦) ليصبح كما هو مبين في الشكل (٩ ـ ٢٨) فما أثر ذلك التغيير على ناتج تنفيذ العمليات ؟

يلاحظ من الشكل (٩-٢٨) أن التغيير الذي حدث يتلخص في أن خطوة كتابة قيمة المتغير I قد تأخرت عن الخطوة (هل I=5؟).



شكل (٩-٢٨)

وهذا يعني طباعة قيمة المتغير I تأتي بعد الانتهاء من عمليات التكرار أي بعد أن يتحقق الشرط (تصبح قيمة I تساوي 5) لذلك يطبع لنا الكمبيوتر على الوسط الخارجي القيمة 5 فقط.

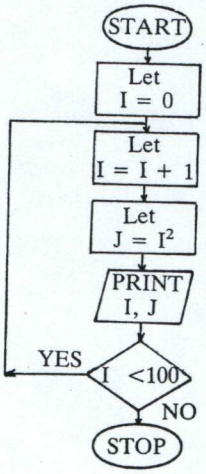
مفهوم العدادCounter:

في كثير من الأحيان نحتاج في برامج الحاسب الالكتروني إلى العد (counting) فقد نحتاج أن نعد عدد كل من الطلاب والطالبات في الصف، وقد تكون هذه العملية سهلة للإنسان لأنها أصبحت ضمن قدراته العقلية التي يكتسبها من الطفولة إلا أن الحاسب يحتاج إلى تصميم خوارزمية للعد (Counting Algorithm) تتضمن خطوات معينة إذا اتبعها استطاع أن يعد.

ويمكن تحديد الخطوات التي يتبعها الكمبيوتر حتى يتمكن من العد في الخطوات الأساسية الآتية:

١) اجعل العداد مساوياً للصفر
٢) اجعل القيمة الجديدة للعداد تساوي القيمة القديمة مضافاً إليها واحد، أي أن: قيمة العداد (الجديدة) = قيمة العداد (القديمة) + 1
(على سبيل المثال (I = I + 1))
٣) نفذ العمليات المطلوب تنفيذها
٤) كرر الخطوات ابتداء من الخطوة (٢).

مثال (٩-٢١): اكتب خوارزمية وارسم خريطة سير العمليات التي يتبعها الحاسب لطباعة الأعداد الطبيعية من 1 إلى 100 ومربعاتها.



شكل (٩-٢٩)

الحل: خطوات الحل المبينة في الشكل (٩-٢٩) هي:

١ ـ ابدأ
٢ ـ اجعل I = 0 (إسناد قيمة ابتدائية للعداد)
٣ ـ اجعل I = I + 1 (زيادة العداد بمقدار الزيادة الدورية)
٤ ـ اجعل $J = I^2$
٥ ـ اطبع J, I
الخطوات المطلوب تكرار تنفيذها.
٦ ـ إذا كانت I<100 عُد إلى الخطوة (٣)
٧ ـ توقف

مثال (٩-٢٢): ارسم خريطة سير العمليات التي يتبعها الكمبيوتر لطباعة الأعداد الزوجية من 1 إلى 100 ومربعاتها.

الحل: خطوات الحل كما هي مبينة في الشكل (٩-٣٠) هي:

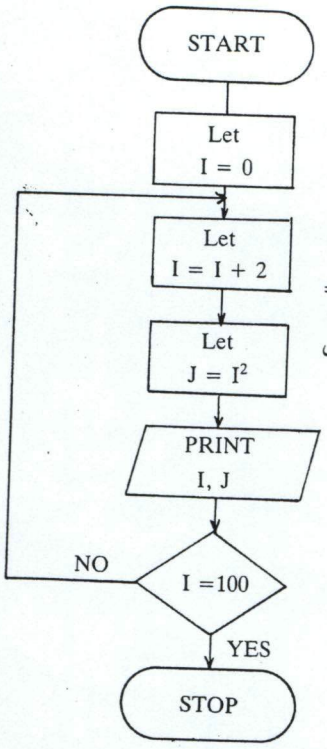
١ ـ ابدأ
٢ ـ اجعل I = 0
٣ ـ اجعل I = I + 2

٤ ـ اجعل $J = I^2$

٥ ـ اطبع I و J

٦ ـ هل تحقق الشرط (I = 100) إذا كان الجواب (NO) عُد إلى الخطوة (٣).

٧ ـ توقف.



شكل (٩-٣٠)

ملحوظة: لاحظ أن القيمة الجديدة للعداد = القيمة القديمة للعداد + 2 أي أن الزيادة الدورية في كل تكرار تساوي 2.

مفهوم المجاميع الإجمالية:

نحتاج في كثير من الأحيان في برامج الكمبيوتر إلى جمع مجموعة كبيرة من الأعداد التي تمثل معطيات ظاهرة معينة، فمثلاً: قد نرغب في إيجاد الوسط الحسابي لأعمار طلاب الجامعة، ولتحقيق هذا يجب أولاً أن نحسب مجموع أعمار الطلاب أي إننا بحاجة إلى إيجاد مجموع N من القيم، ولا يخفى على أحد أنه ليس عملياً أن نعطي رمزاً أبجدياً لكل عدد يمثل عمر طالب، فقد نحتاج إلى أكثر من عشرة آلاف رمز، وفي مثل هذه الحالات، نصمم خوارزمية معينة للتجميع تسمى خوارزمية التجميع (Sum-mers Algorithm) تتضمن خطوات محددة إذا اتبعها الكمبيوتر استطاع أن يجمع أية كمية من البيانات باستخدام متغيرين اثنين فقط أحدهما هو المتغير الذي نجمعه والآخر

هو الجمع الإجمالي (مجموع)، ويمكن تحديد الخطوات التي يجب أن يتبعها الحاسب لتحقيق ذلك في أربع خطوات هي:

١ ـ اجعل المجموع مسلوياً صفراً .
٢ ـ ادخل قيمة واحدة للمتغير.
٣ ـ اجعل القيمة الجديدة للمجموع تساوي القيمة القديمة له مضافاً إليها القيمة المدخلة للمتغير، أي أن:
قيمة المجموع الجديدة = قيمة المجموع القديمة + آخر قيمة مدخلة للمتغير.
على سبيل المثال: S = S + x.
٤ ـ تكرار الخطوات ابتداء من الخطوة (٢)

مثال (٩-٢٣): اكتب خوارزمية وارسم خريطة سير العمليات لإيجاد الوسط الحسابي لأعمار طلاب صفك.

الحـــل:

المتغيرات:
N: يمثل عدد الطلاب في الصف
I: عداد
X: عمر الطالب
S: مجموع أعمار الطلاب
A: معدل أعمار الطلاب

وتكون خطوات الحل كما هو مبين في الشكل (٩-٣١) كما يأتي:

١ ـ ابدأ
٢ ـ ادخل اجمالي عدد الطلاب (N) في الصف
٣ ـ اجعل I = 0
٤ ـ اجعل S = 0
} إسناد قيم مبدئية لـ I و S .

٥ ـ اجعل I = I + 1

٦ ـ ادخل X

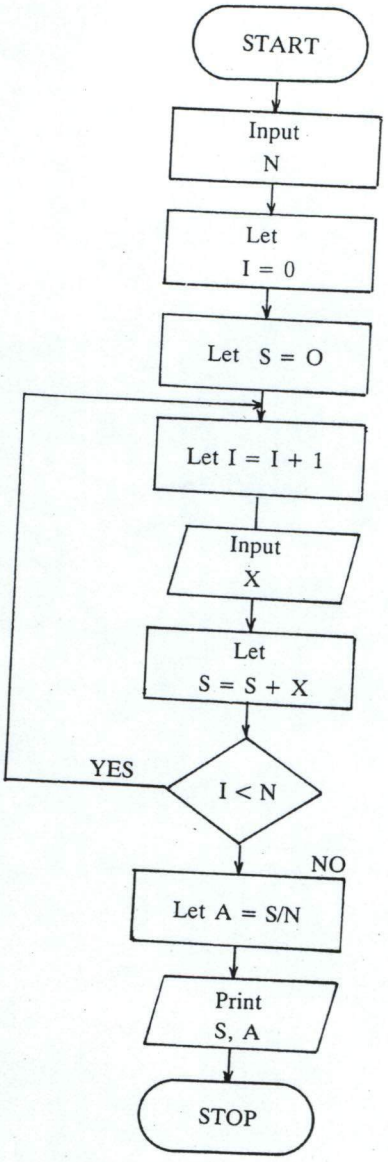
٧ ـ اجعل S = S + X

٨ ـ إذا كانت I < N عُد إلى الخطوة (٥)

٩ ـ اجعل A = S/N

١٠ ـ اطبع A و S

١١ ـ توقف



شكل (٩-٣١)

مثال (٩-٢٤): اسم خريطة سير العمليات لحساب قيمة الاقتران

$$F(X) = X^2 + 1$$

لمجموعة من قيم X الموجبة.

الحل: خطوات الحل المبينة في الشكل (٩-٣٢) هي:

١ ـ ابدأ

٢ ـ اقرأ قيمة X

٣ ـ إذا كانت قيمة $X > 0$ احسب قيمة اقتران F (X) من القاعدة:

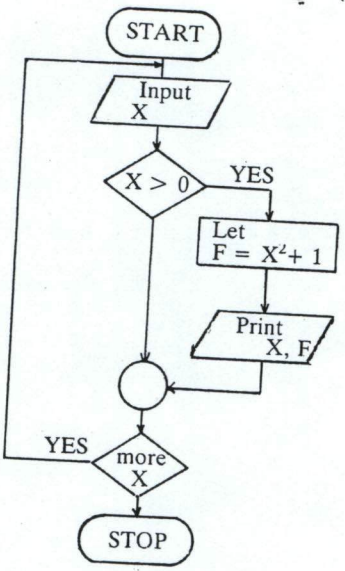
$$F(X) = X^2 + 1$$

اطبع قيم كل من X, Y

٤ ـ هل هناك قيم أخرى لـ X؟

إذا كان الجوابُ نعم عُد إلى الخطوة (٢).

٥ ـ توقف



شكل (٩-٣٢)

مثال (٩-٢٥): ارسم خريطة سير العمليات لإيجاد مجموع الأعداد من 1 إلى 20.

الحل: تكون خطوات الحل كما هي مبينة في الشكل (٩-٣٣) كما يأتي:

١ ـ ابدأ

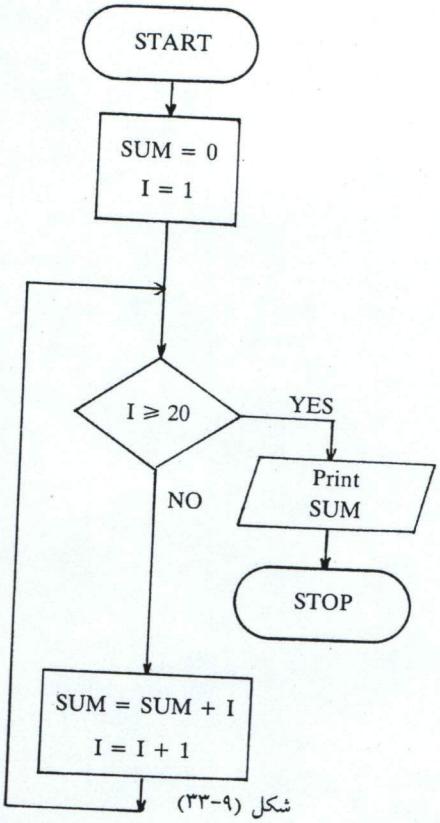
٢ ـ ضع SUM = 0

I = 1

٣ ـ إذا كان جواب الشرط ($1 \geq 20$)

(YES) انتقل إلى الخطوة ٦

٤ - إجعل: SUM = SUM + I
I = I + 1
٥ - عُد إلى الخطوة (٣)
٦ - اكتب المجموع SUM
٧ - توقف



شكل (٩-٣٣)

نلاحظ أن قيمة العداد تبدأ بقيمة ابتدائية مقدارها (1) ثم يتبع ذلك سؤال هل قيمة العداد تزيد القيمة العظمى للعداد (20) فإذا كان الجواب (NO) كرر تنفيذ مجموعة جمل التكرار وهي:

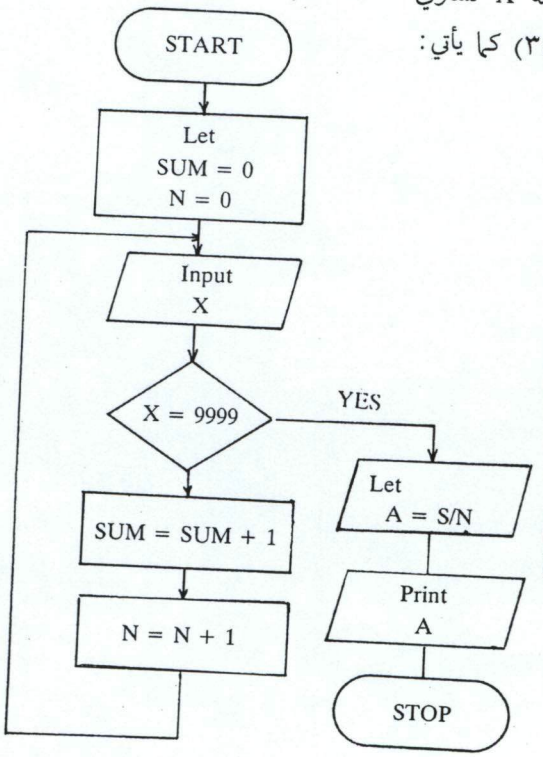
SUM = SUM + I

ثم أضف الزيادة الدورية لقيمة العداد
وإذا كان الجواب (YES) تابع تنفيذ العمليات الواقعة خارج جمل التكرار.

مثال (٩-٢٦): ارسم خريطة سير العمليات لإيجاد الوسط الحسابي لعدد غير محدد من القيم.

الحـــل:

في مسائل التكرار السابقة كان عدد القيم معلوماً، بينما في هذه المسألة العدد غير معلوم، للتغلب على هذه المشكلة يمكننا استخدام علامة مميزة تبين نهاية البيانات (End-of-data) ففي هذه المسألة يمكن أن نفترض العلامة المميزة قيمة عددية تفوق أية قيمة ممكنة كأن نقول إذا كانت قيمة X تساوي 99999 انتهت البيانات وعليه تكون خطوات الحل المبينة في الشكل (٩-٣٤) كما يأتي:



شكل (٩-٣٤)

١ ـ ابدأ

٢ ـ اجعل: SUM=0 (المجموع)

N = 0 (العداد)

٣ ـ ادخل قيمة X

٤ ـ إذا كانت قيمة X تساوي 9999 انتقل إلى الخطوة (٨)

٥ ـ اجعل N = N + 1

٦ ـ اجعل SUM = SUM + X

٧ ـ اذهل إلى الخطوة ـ ٣

٨ ـ احسب الوسط الحسابي

A = SUM / N

٩ ـ اطبع الوسط الحسابي

١٠ ـ توقف

تمارين للتدريب

(١) اكتب خريطة سير العمليات التي يتبعها الحاسب لطباعة الأعداد الفردية من 1 إلى 100 ومربعاتها.

(٢) اكتب خريطة سير العمليات لإيجاد قيمة الاقتران

$$F(x) = x^5 + 2x^3 + x + 5$$

لجميع قيم x من 1 إلى 10 وبزيادة دورية مقدارها 1/2.

(٣) اكتب خريطة سير العمليات لطباعة الأعداد التي تقبل القسمة على 2 وتقع ما بين 10 و 100.

(٤) اكتب خريطة سير العمليات لإيجاد مجموع حدود كل من السلاسل الآتية:

$$1 + 2 + 3 + 4 + \ldots + 100 \quad (1)$$

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \ldots + \frac{1}{50} \quad (2)$$

$$1 + x + x^2 + x^3 + \ldots + x^{10} \quad (3)$$

(٥) حاصل ضرب الأعداد من 1 إلى 100

(٦) إيجاد مضروب العدد N وفقاً للقاعدة

$$N! = \begin{cases} 1 & ; N = 0 \\ 1 \times 2 \times 3 \times 4 \ldots \times N; & N \neq 0 \end{cases}$$

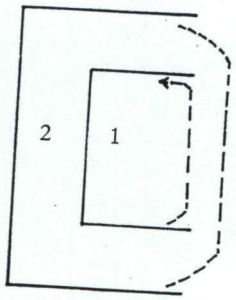
(٧) مجموعة من البطاقات عددها N يحتوي كل منها على عدد صحيح موجب، ارسم خريطة سير العمليات لإيجاد الفرق بين أكبر قيمة وأصغر قيمة.

(٨) إفترض وجود ملف بيانات يحتوي على N من السجلات، يحتوي كل منها على إسم الموظف، عدد ساعات العمل الأسبوعي، أجرة ساعة العمل، ارسم خريطة سير العمليات التي يتبعها الكمبيوتر لقراءة سجلات ملف المدخلات واحداً تلو الآخر وحساب الراتب الأسبوعي لكل موظف بحيث:

الراتب الأسبوعي = عدد ساعات العمل × معدل أجرة الساعة إذا لم يزد عدد ساعات العمل الأسبوعي عن 40 ساعة أسبوعياً. وكل ساعة عمل إضافية تحسب بما يعادل ساعة ونصف.

[٩-٤:٤] خرائط التكرار المتداخلة *(Nested-loop flowcharts)*:

في هذا النوع من الخرائط تكون التكرارات متداخلة تماماً بحيث لا تتقاطع فإذا كان لدينا تكرارين من هذا النوع (شكل ٩-٣٥) فيسمى التكرار (1) تكرار داخلي بينما التكرار (2) يسمى تكرار خارجي ويتم التناسق في عمل هذين التكرارين بحيث تكون أولوية التنفيذ للتكرار الداخلي.



شكل (٩-٣٥)

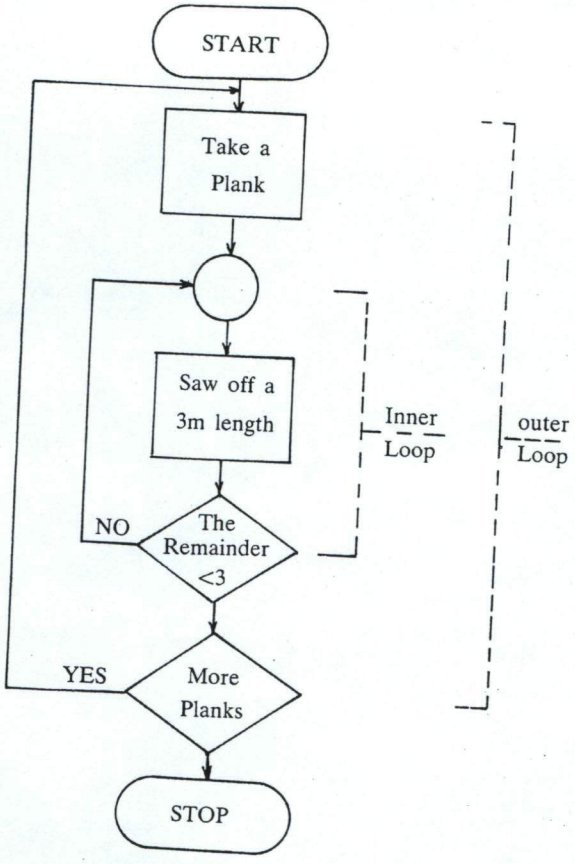
مثال (٩-٢٧): يرغب نجار في تقطيع مجموعة من القطع الخشبية طول كل منها يزيد عن 3 متر إلى قطع صغيرة طول الواحدة منها يساوي 3 متر. ارسم خريطة سير العمليات التي تمثل التسلسل المنطقي والعملي لتحقيق ذلك.

خطوات الحل المبينة في الشكل (٩-٣٦) هي:

١ ـ ابدأ
٢ ـ خذ قطعة
٣ ـ اقطع منها قطعة طولها 3 متر
٤ ـ هل طول القطعة المتبقية أقل من 3 متر؟ إذا كان الجواب (NO) اذهب إلى الخطوة ٣ وإلا اذهب إلى الخطوة ـ ٥ .

٥ ـ هل هناك مزيد من القطع المراد تقطيعها؟ إن كان الجواب نعم فاذهب إلى الخطوة (٢) وإن كان لا فاذهب إلى الخطوة (٦).

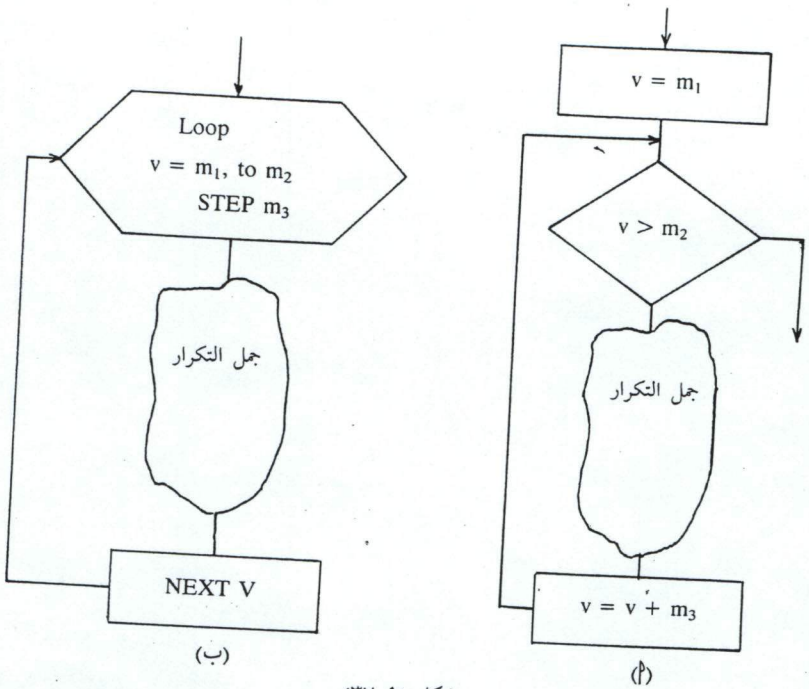
٦ ـ توقف.



شكل (٩-٣٦)

(٩-٤ : ٥) صيغة التكرار باستعمال الشكل الاصطلاحي للتكرار:

تعرضنا في الفقرتين السابقتين لمفهوم التكرار البسيط والتكرارات المتداخلة والآن سنقدم طريقة أسهل لتحقيق معنى التكرار باستخدام مفهوم الشكل الاصطلاحي للتكرار كما هو مبين في الشكل(٩-٣٧).



شكل (٩-٣٧)

نلاحظ في الشكل (٩-٣٧ (أ)) أننا نحتاج إلى العناصر الآتية:

(١) المتغير v ويسمى متغير جملة التكرار

(٢) القيمة الابتدائية لمتغير جملة التكرار (m_1)

(٣) القيمة النهائية لمتغير جملة التكرار (m_2)

(٤) قيمة الزيادة الدورية لمتغير جملة التكرار (m_3)

وكذلك نلاحظ إن إجراءات التكرار كانت تتم طبقاً للخطوات الآتية والمفصلة من قبل المبرمج:

(١) اجعل متغير التكرار يبدأ بقيمة ابتدائية مقدارها m_1 .

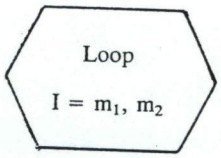
(٢) إذا كانت قيمة متغير جملة التكرار تزيد على القيمة النهائية اذهب إلى أول جملة تلي مجموعة جمل التكرار وإلا اذهب إلى الخطوة - ٣ .

(٣) نفذ مجموعة جمل التكرار.

(٤) أضف الزيادة الدورية للعداد، ثم اذهب إلى الخطوة ـ ٢ .

وباستخدام الشكل الاصطلاحي المبين في الشكل (٩-٣٧ (ب)) يمكننا الاستغناء عن الخطوات المفصلة من ١-٤ حيث تنفذ بصورة أوتوماتيكية من قبل الحاسب وهذا من شأنه تسهيل عملية البرمجة واختصار عدد الجمل في البرنامج وتجنب الأخطاء.

ملحوظة: تعتبر قيمة m_3 تساوي 1 دائماً إذا لم تعط قيمة أخرى بخلاف ذلك، وفي حالة عدم ذكر قيمة m_3 يصبح الشكل الاصطلاحي الوارد في الشكل (٩-٣٧ (ب)) كما يلي:



حيث تكون قيمة m_3 تساوي 1 وبصورة أوتوماتيكية

وفيما يلي مجموعة من **الأمثلة التوضيحية**:

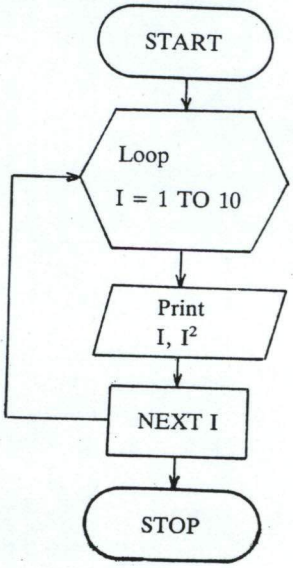
مثال (٩-٢٨): اكتب خريطة سير العمليات لطباعة الأعداد من 1 إلى 10 ومربعاتها:

الحـــل:

القيمة الابتدائية = 1

القيمة النهائية = 10

الزيادة الدورية = 1



شكل (٩-٣٨)

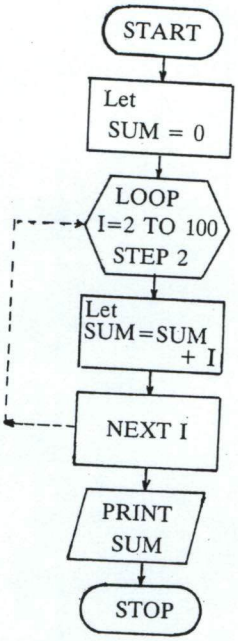
مثال (٩ ـ ٢٩) : اكتب خريطة سير العمليات لإيجاد مجموع الأعداد الزوجية من 2 إلى 100 .

الحـــل :

القيمة الابتدائية 2

القيمة النهائية 100

الزيادة الدورية 2



شكل (٩-٣٩)

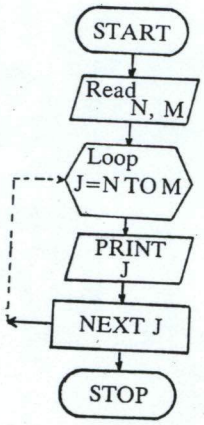
مثال (٩-٣٠): اكتب خريطة سير العمليات لقراءة عدديه وطباعة الأعداد الواقعة بينهما على الترتيب.

الحـــل :

القيمة الابتدائية = قيمة العدد الأول

القيمة النهائية = قيمة العدد الثاني

الزيادة الدورية = 1



شكل (٩-٤٠)

Input and Output الإدخال والإخراج

توفر لغة سي مجموعة من الاجراءات التي يمكن من خلالها القيام بعملية ادخال البيانات للبرنامج ، وكذلك اخراج النتائج على الشاشة.

كل اجراء مرتبط بملف توجيه معين حيث يستدعي هذا الملف في أول البرنامج بالعبارة # include فمثلا الاجراء printf () معرف بالملف stdio.h ويكتب العبارة#include <stdio.h> في أول البرنامج حتي يتعرف المترجم على الاجراء وهكذا مع باقي الاجراءات.

إجراء الطباعة على الشاشة printf( )

- تستخدم اجراء الطباعة printf ( ) لطباعة البيانات بجميع أنواعها (string , char , float , int) على الشاشة فقط.
- يأخذ اجراء الطباعة عدة صور وكذلك معاملات واختصارات تحدد شكل المخرجات

مثال 1

```
printf ( " welcome To C Language " )
```

هنا يتم طباعة ما بين علامتي التنصيص" "

مثال 2

```
# include < stdio.h >
void main ( )
{    printf ( " \n welcome \n To \n C language " ) ;
}
```


١-٩