الفصل السابع

مقدمة الى البرمجة

1- تقديم :

لو افترضنا وجود مشكلة أو مسألة معينة نريد الوصول الى حل لها فاننا يجب أن نمتلك شيئين أساسيين لإيجاد ذلك الحل وهما :

أ- **المعالج** : وهو الإنسان أو الآلة التي تقوم بتنفيذ سلسلة التعليمات اللازمة لحل المسألة .

ب- **الخوارزمي** : وهي سلسلة التعليمات التي إذا تم تنفيذها من قبل معالج معين فسيتم أنذاك حل المسألة . على شرط أن يتم تنفيذ سلسلة التعليمات تلك ، على التتابع أى واحدة بعد الأخرى ودون حذف إحداها ، أو تكرار تنفيذ اياً من تلك التعليمات .

ولكي يقوم المعالج بحل المسألة فإنه يجب أن يتوفر شرطان هما :

الأول هو أن يتمكن المعالج من فهم كل خطوة من خطوات الخوارزمي، **والثاني** أن يكون المعالج قادر على تنفيذ آى خطوة من خطوات الخوارزمي. أو بعبارة أخرى فإن **الخوارزمي** يجب أن تتم كتابته بطريقة تُمكنْ **المعالج** من فهم وتنفيذ كل خطوة فيه، وإذا كان المعالج هو الحاسبة الألكترونية فإن الخوارزمي يدعى **البرنامج** .أما **البرمجة** فهي عملية وصف التعليمات اللازمة لحل المسألة على شكل برنامج ، ويكتب البرنامج بلغة خاصة تدعى **لغة البرمجة** .

2- لغات البرمجة :

أن وسيلة التفاهم بين إنسان وآخر كما نعرف ، هي اللغة بأي صورة كانت ، فاللغة قد تكون مسموعة أو مقروءة أو على شكل إشارة أو جفرة ..الى آخره من الطرق الممكن للتعبيرعن شئ ما . ولكل بلد لغة أساسية واحدة على الأقل يمكن أن يتفاهم بواسطتها مواطنو ذلك البلد فيما بينهم، فلو أردنا أن نتفاهم أو نتحادث مع بعضنا البعض كعرب فاننا نتحدث باللغة العربية، ولو ذهبنا الى انكلترا مثلاً فإنه يفترض بناأن نتقن التحدث باللغة الإنكليزية كي نستطيع التفاهم مع سكان ذلك البلد، وهكذا لا بد أن تكون هناك لغة مشتركة بين إنسان وآخر ، وإلا لإنقطع الاتصال بينهما .

ونفس المبدأ يمكن أن نطبقه على التفاهم والتخاطب والإتصال بين الإنسان والحاسبة الألكترونية ، فبدون لغة مشتركة بين الإنسان والحاسبة فإنه لا يوجد اتصال بينهما وبالتالي لا يستطيع الإنسان ، الذي قام بصنع الحاسبة ، من الإستفادة منها . واللغة في معناها العام ، هي عبارة عن مجموعة من الرموز والحروف والإشارات والأرقام إلى آخره، والتي تنتظم فيما بينها لتكوّن كلمات ومقاطع متعارف عليها من ناحية الشكل والمعنى .بحيث أن مجرد سماع الكلمة أو رؤيتها (مكتوبة) يوحي فوراً للسامع (أو القارئ) بالمعنى المقصود من الكلمة . ولكل لغة قواعد خاصة بها ، تنظم العلاقة بين الكلمات المرتبطة في مجموعة واحدة والتي نسميها الجملة او التعبير . لذلك لا بد أن تتكوّن لغة البرمجة من مقاطع وحروف وكلمات تنتظم مع بعضها بقواعد معينة . بشرط أن تكون مفهومة من قبل الحاسبة والإنسان معاً ، ولا يشترط أن تكون اللغة بالطبع مشابهة تماماً للغة التفاهم المستعملة بين إنسان وآخر لأن ذلك غير ممكن عملياً ، بسبب طبيعة الأجزاء الألكترونية (التي تتكون منها الحاسبة)، وطريقة عملها. إضافة إلى قواعد لغتنا الاعتيادية ومفرداتها هي أصعب بكثير من أن تستوعبها وتتعامل بها الحاسبة الألكترونية .

كما قلنا سابقاً فإن خوارزمي حل المسألة ، حين يكتب بإحدى لغات

البرمجة فإنه يدعي البرنامج ، ولغات البرمجة متعددة ومتنوعة طبقاً للغرض الذي تستخدم من أجله تلك اللغة. ولو عدنا إلى مثالنا حول اللغة العربية نجد أن مفردات اللغة ، التي نستعملها في المدرسة تكثر فيها المصطلحات العلمية والتي تتعلق بالدروس والمناهج ؛ ولكننا لو ذهبنا إلى أحد مراكز الشباب للتدرب على أنواع الرياضة، فعند ذاك لا يمكننا بالطبع أن نستعمل نفس المفردات التي نتعامل بها في المدرسة فالمفردات والتعابير التي نستعملها ستحتوي بالتأكيد على مصطلحات في الرياضة بنسبة كبيرة . وهكذا إن لكل موضوع مفردات خاصة به أو لغة خاصة به . وذات الشيء ينطبق على الحاسبة ولغة البرمجة فيها . فهناك لغات للأغراض التجارية مثل لغة (كوبول)، وهناك لغات للأغراض العلمية مثل لغة البرمجة (فورتران). والفرق بين لغة وأخرى من لغات البرمجة ، هو انه توجد لكل من هذه اللغات مفردات خاصة تخدم الغرض الذي من أجله وجدت هذه اللغة، وتصنف هذه اللغات إلى نوعين أساسيين فهناك لغات البرمجة العالية ، ولغات البرمجة الواطئة . ولغات البرمجة العالية هي أقرب إلى لغتنا الاعتيادية ، والحاسبة لا تستطيع فهم اللغات العالية مباشرة بل يجب تحويلها أو ترجمتها إلى لغات واطئة تستطيع الحاسبة فهمها والتعامل معها مباشرة . وتتم عملية ترجمة لغات البرمجة العالية إلى لغات البرمجة الواطئة عادة بواسطة أجهزة خاصة تكون من ضمن تركيب الحاسبة تدعى (المترجم) .

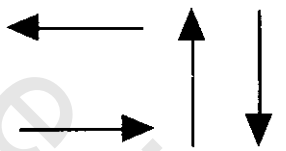
أما لغات البرمجة الواطئة فتتكون من مفردات تستطيع الحاسبة التعامل معها مباشرة مثل لغة الماكنة ، وهي طريقة معقدة ومطولة بالنسبة للإنسان الاعتيادي ،،تحتاج إلى شخص متخصص كي يتقنها ، إضافة إلى أن (لغة الماكنة) تختلف من حاسبة ألى أخرى ، لذلك سوف نحاول في هذا الكتاب أن ندرس بالتفصيل إحدى لغات البرمجة العالية وهي لغة بيسك وهي لغة متعددة الأغراض للمبتدئين .

3- المخطط الانسيابي:

لقد تعودنا أن نكتب خوارزمي حل المسألة على شكل خطوات

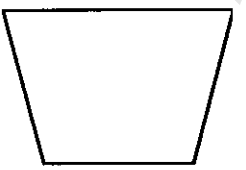
مكتوبة . ونستطيع أن نعبر عن الخوارزمي بطريقة أخرى وهي المخطط الأنسيابي . والمخطط الانسيابي هو خوارزمي يتم تمثيله بواسطة أشكال ورسوم هندسية معينة . وكل شكل من هذه الأشكال يدل على خطوة أساسية من خطوات الحل ، والتي يتم فيما بعد ترجمتها إلى تعابير بإحدى لغات البرمجة . وأهم هذه الأشكال التي تستخدم في المخطط الانسيابي هي ما يلي:

1- **السهم** :ويمثل اتجاه سير البرنامج أو الحل حيث يبين رأس السهم الخطوة التالية في الحل .

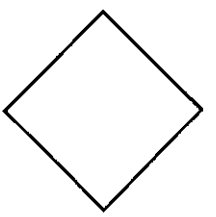


2- **الشكل البيضوي** : يستعمل لتمثيل **البداية** أو **النهاية** في البرنامج

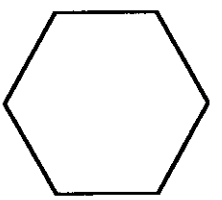
3- **شبه المنحرف** : يمثل **ادخال** البيانات والمعلومات أو يمثل **إخراج** النتائج



4- **المعين** : يمثل **نقطة اتخاذ قرار** ما ، أو تحويل تتابع سير البرنامج، وفقاً لحالة معينة ، إلى خطوات أخرى خارج التتابع الأصلي لسير البرنامج .

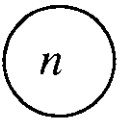


5- **الشكل السداسي** : **استدعاء برنامج ثانوي** ، يتم بواسطته تنفيذ مجموعة من الخطوات التي يتطلبها البرنامج



6- **المستطيل : التعابير الرياضية والمنطقية** أو ما شابه .

7- **دائرة الربط :** وهي دائرة فيها رقم، وتوضع على مسار البرنامج وتعني أن لهذا المسار تتمة في مكان آخر ، يحتوي على دائرة تحمل نفس الرقم .



مثال :

أكتب الخوارزمي اللازم لحل المسألة التالية . ثم ارسم المخطط الانسيابي له :

ص = س^2 + 2 س + 4

الحل :

يمكن أن نكتب الخوارزمي كما يلي :

1- البداية

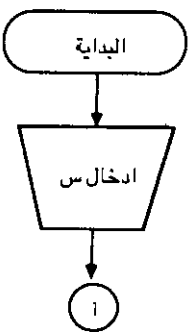
2- أدخال قيمة س .

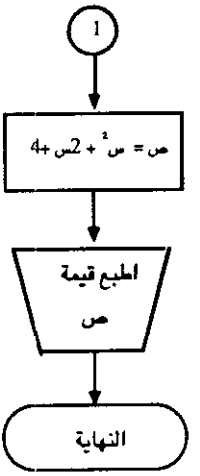
3- ص = س^2 + 2 س + 4

4- أطبع قيمة ص .

5- النهاية .

وفيما يلي المخطط الانسيابي المكافيء للخوارزمي اعلاه :





وبالطبع ، فإننا عندما نرسم المخطط الانسيابي عادة لا نحتاج الى كتابة الخوارزمي أيضاً ، ولكننا فعلنا ذلك في هذا المثال للتوضيح فقط.

مثال :

ضع الخوارزمي المناسب لما يجب على سائق السيارة (أو المركبة) أن يفعله حين يقود مركبته ويصل قرب إشارات المرور الضوئية (ترافك لايت) . ثم ارسم المخطط الانسيابي لذلك .

الحل :

يمكن وضع الخوارزمي التالي الذي يمثل سلوك السائق النموذجي عندما يواجه إشارات المرور الضوئية ، وكما يلي :

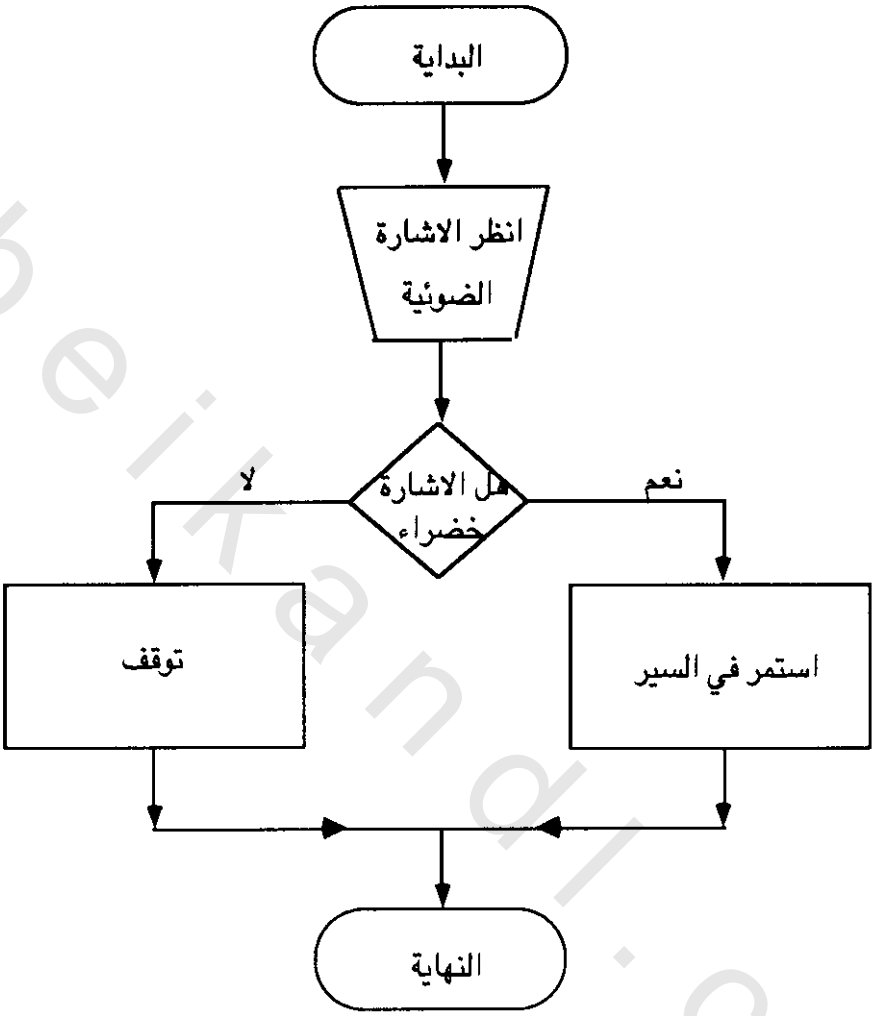
1- البداية.

2- أنظر أي إشارة ضوئية مفتوحة أمامك .

3- هل الأخضر مفتوح ← نعم ← استمر في السير .
لا ← توقف .

4- النهاية .

أما المخطط الانسيابي فيمكن وضعه بالصورة التالية :



مثال :

أرسم المخطط الانسيابي الملائم لخوارزمي سلوك سائق السيارة عندما يجد إشارات المرور الضوئية (ترافك لايت) عاطلة عن العمل وسلوكه إذا وجدها صالحة للعمل معاً في نفس المخطط .

الحل :

يمكن وضع الخوارزمي كالتالي :

1- البداية .

2- هل الإشارات المرورية الضوئية تعمل أم لا .

3- نعم الإشارات تعمل .

اذن هل الأخضر مفتوح ← نعم ← استمر في السير .

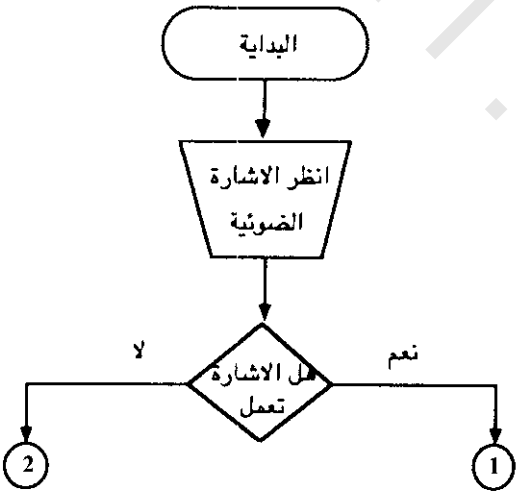
لا ← توقف .

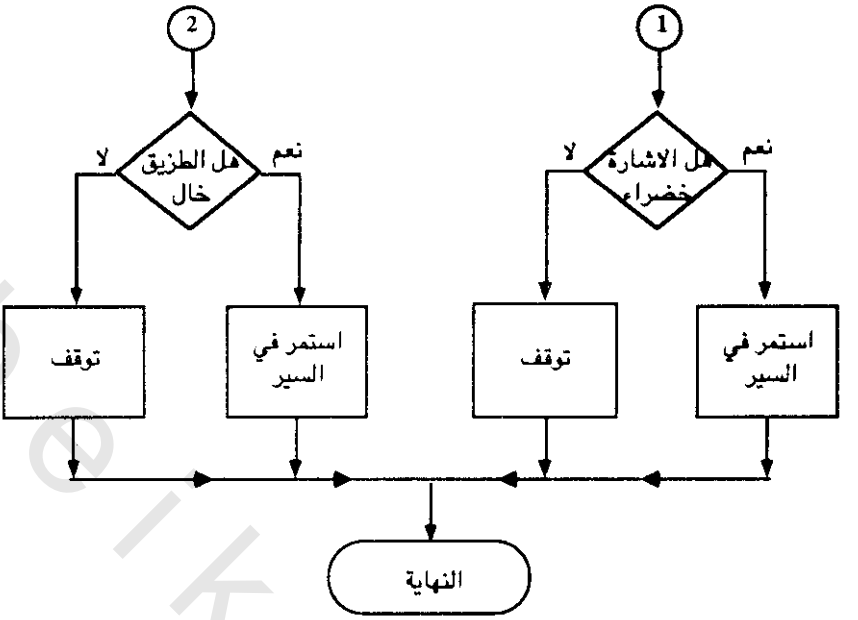
4-الإشارات لا تعمل . انتبه جيداً (هل الطريق خالي من السيارات التي لا تتقاطع بمسارها مع مسارك) ← نعم ← سر بإتجاهك

لا ← توقف

5- النهاية .

ويكون المخطط الانسيابي كالتالي :





4- التدوين اليائي (أو ألأسّي) :

عندما تكون قيمة عدد ما كبيرة أو صغيرة ، فإنها يمكن ان تكتب بالصيغة الإعتيادية فمثلاً (000 000 000 1) وتعني بليون (الف مليون)، أو (0.00000001) تعني جزء من مائة مليون جزء وسواها من الامثلة الكثيرة . ويمكننا ان نكتب العددين اعلاه **بالصيغة الأسية** على التوالي كما يلي : 1×10^{9} ، 1×10^{-8} . أما في لغات البرمجة بصورة عامه فيجب كتابة القيد ضمن مستوى واحد (أي ضمن نفس السطر) ، ولا يوجد طريقة لكتابته بمستويين ، لذا فإننا يمكننا كتابة العدد 1×10^{9} بالصيغة 1.E+09 (أو 1.E9) . وهذا يعني اننا قمنا بالاستعاضة عن (x10) بالحرف (E) ، وبعد الحرف (E) توجد ثلاثة مواضع ، أقربها الى الحرف (أى الى يمين الحرف) مخصص لإشارة الاس (موجبه أو سالبة) ، علماً بأنه يمكن حذفها أو عدم كتابتها اذا كانت موجبة . أما الموضعين التاليين لموضع الاشارة ، فهما يمثلان مرتبتين للأس (الآحاد والعشرات)، وكحد أقصى لقيمة الأس لا تتجاوز الرقم

(63) ولا تقل عن (64 -) . وفيما يلي أمثله لأعداد مختلفة مع طريقة كتابتها بالصيغة الأسية وبالتدوين اليائي :

$1000\ 000\ 000 = 1 \times 10^{9} = 1.E + 09 = 1.E9$

$0.000\ 00001 = 10 \times 10^{-8} = 1.E - 08 = 1.E - 8$

$0.000\ 415\ 68 = 4.1568 \times 10^{-4} = 4.1568\ E - 04 = 4.1568E - 4$

$5\ 6321\ 000\ 000 = 5.6321 \times 10^{10} = 5.6321E + 10 = 5.6321E10$

5- الدقة :

ان القيمة العددية ومقدار الدقة فيها يؤثر بشكل مباشر على عدد خلايا الخزن في الذاكرة وعلى وقت المعالجة اللازم لأي عملية ، لذا يمكن ان تكون القيم العددية للثوابت أو المتغيرات بمستويين من الدقة في النتائج ، هما **الدقة المفردة** (الاعتيادية) والتي تكون بستة ارقام . أما المستوى الثاني من الدقة فيسمى **الدقة المضاعفة** ، حيث يمكن ان تكون فيه القيمة لغاية أربعة عشر أو حتى ستة عشر رقماً.

6- الحرف :

الحرف هنا لا يعني فقط احد الحروف الابجدية ، بل أي رمز مفرد سواء أكان حرفاً ابجدياً أو رقماً أو اشارة حسابية (-, + , / , * , ^) أو رمزاً خاصاً مثل (#,!,$,%,? ...الى آخره) .

7- الإشارات الحسابية والمنطقية :

يمكن تقسيم الاشارات الحسابية والمنطقية الى نوعين أساسيين هما:

7-1- الاشارات الحسابية :

وهي الاشارات التي تستخدم لتمثيل العمليات الحسابية الاعتيادية (الجمع ، الطرح ، القسمة ، الضرب) ، مضافاً اليها اشارة خاصة للرفع (الى الاسس) . وكما مبين ادناه :

+ لتمثيل عملية **الجمع** .
- لتمثيل عملية **الطرح** .
* لتمثيل عملية **الضرب** .
/ لتمثيل عملية **القسمة** .
^ لتمثيل عملية **الرفع الى الاسس** .

2-7- الإشارات المنطقية :

عدا الإشارات في الفقرة (أ) اعلاه ، توجد هناك إشارات أخرى مهمة نحتاجها في الحساب بين الكميات العددية أو الجبرية أو المنطقية . وفيما يلي قائمة بها ، والتي نستخدمها في لغة بيسك .

= المساواة بين قيمتين أو طرفين لمعادلة أو عبارة . مثل : (X=Y) ، أي (X) تساوي (Y) .

<> عدم المساوة . مثل (X < > Y) ، أي (X) لا تساوي (Y) .

< أقل من . مثل (X < Y) ، أي (X) أقل من (Y) .

> أكبر من . مثل (X > Y) ، أي (X) أكبر من (Y) .

=< أقل من أو يساوي . مثل (X < = Y) ، أي (X) أقل من أو يساوي (Y) .

=> أكبر من أو يساوي. مثل (X > = Y) ،أي (X) أكبر من أو يساوي (Y) .

8- الثوابت :

تعريف **الثابت** بلغة البرمجة (بيسك)، هو الرقم المجرد أو الكمية التي تمتلك قيمة واحدة فقط ولا تتغير أثناء تنفيذ البرنامج . وتكون **الثوابت** على نوعين هما ، **الثوابت الحرفية والثوابت العددية** (أو الرقمية) . وفيما يلي تفاصيل كل منها :

1-8- الثوابت الحرفية :

هي عبارة عن تتابع (أو **سلسلة**) من الحروف ، على ان لا

يتجاوز طولها عدد معين من الحروف (255 حرف) . ويجب ان نضع سلسلة الحروف هذه بين **إشارتي إقتباس** (") . **مثال** :"HELLO" ، "أحمد 635" ، " هاتف 7654321" ، "22ABC", "Vitamin A" .

8-2- الثوابت العددية (أو الرقمية) :

يتكون **الثابت العددي** (الرقمي) من مجموعة من الارقام (سواءاً كانت موجبة أو سالبة) ، ولا تحوي على فوارز اعتيادية ، رغم انها يمكن ان تحوي فاصلة عشرية . وتكون الثوابت الرقمية على عدة أنواع منها :

8-2-1- الثوابت الصحيحة :

وهي جميع **الاعداد الصحيحة** (أي **لا تحوي على فاصلة عشرية**) التي تتراوح قيمها بين (32768-) , (32767) . أما اذا كان العدد موجباً فقط (أو بدون إشارة) فأن قيمته تتراوح بين (0) و (65535) . **مثال** : 12+,-507,54001.

8-2-2- الثوابت الحقيقية (ذات الفاصلة العشرية) :

وهي كافة القيم العددية (الموجبة أو السالبة) ، والتي **تحوي على الفاصلة العشرية** ، سواءاً كانت مكتوبة بالصيغة الاعتيادية أو الصيغة الأسية . **مثال** : 3.57×10^9 , -0.461 , 24.59 E-6 .1.

9- المتغيرات :

يمكن تعريف المتغير بأنه كمية يمكن ان يكون لها أكثر من قيمة خلال تنفيذ البرنامج . وقد تكون الكمية متغيرة كنتيجة للمعالجة التي تتم اثناء تنفيذ البرنامج ، أو يتم جعلها كذلك بواسطة المبرمج .

9-1- تسمية المتغيرات :

يمكن تسمية المتغير بأي عدد من الحروف (على ان لا يتجاوز

(255) **حرف في معظم الحاسبات)** ، **ولكن الحاسبة لا تستطيع أن تميز سوى عدد محدود** (لا يتجاوز عادة (6) حروف) **تقع في بداية الرمز** . ويمكن تلخيص أهم الشروط التي يجب أن تتوفر في تسمية رموز المتغيرات بما يلي :

1) يجب أن **يبدأ** اسم المتغير **بحرف ابجدي** .

2) **لا يحوي** الاسم **أي اشارات حسابية أو منطقية** ، بل **يتكون من حروف ابجدية وأرقام فقط** ويمكن ان يحوي ايضاً بعض العلامات والاشارات الخاصة والتي سنشرحها لاحقا ، وهذه العلامات هي (% , $, ! , #) .

3) **لا يتجاوز عدد الحروف** ، التي تكّون اسم (أو رمز) المتغير ، (255) **حرف و/ أو رقم** .

4) **يمكن أن يكون الحرف الاخير** في الرمز الحرفي للمتغير ، **احد العلامات التالية** ، وكما مبين إزاء كل واحدة منها :

أ) % تستخدم هذه العلامة للتوضيح بأن الرمز الذي ينتهي بهذه العلامة هو **رمز لمتغير ذا قيمة عددية صحيحة** .

ب) ! تستخدم هذه العلامة للتوضيح بأن الرمز الذي ينتهي بهذه العلامة هو **رمز لمتغير ذا قيمة عددية حقيقية بدقة اعتيادية** .

حـ) # تستخدم هذه العلامة للتوضيح بأن الرمز الذي ينتهي بهذه العلامة هو **رمز لمتغير ذا قيمة عددية حقيقية بدقة مضاعفة** .

د) $ تستخدم هذه العلامة للتوضيح بأن الرمز الذي ينتهي بهذه العلامة هو **رمز لمتغير حرفي (سلسلة)** .

ولا يفوتنا ان ان نذكر هنا بأنه **اذا لم نضع أي علامة من هذه العلامات الاربعة** في نهاية رمز المتغير ، فأن **الحاسبة سوف تفترض بأنه رمز لعدد حقيقي** .

5) يجب أن **لا يحوي الرمز على أية فراغات** بين الحروف المكّونه له .

6) يجب أن **لا يبدأ الرمز بأي كلمة من الكلمات المحجوزة اصلاً لإستخدامات الحاسبة** مثل التعابير ، الاوامر ، الوظائف ..الى آخره ، (انظر الفصل التاسع) . ومن هذه الكلمات التي يفترض ان لا يبدأ بها رمز المتغير مثلاً LET, PRINT, GOTO,IF .. الى آخره .

9-2- أمثلة عن تسمية المتغيرات :

فيما يلي بعض الامثلة عن تسمية المتغيرات سوف نناقش كل واحدة منها سواءاً كان ممثلاً بصورة صحيحة أم خاطئة :

L% هذا الرمز يمثل لنا متغير ذا قيمة هي عبارة عن عدد صحيح، وذلك لوجود علامة (%) في نهاية رمز المتغير .

N123JK! هذا الرمز يمثل متغير ذا قيمة هي عبارة عن عدد حقيقي بدقة إعتيادية ، وذلك لوجود علامة (!) في نهاية رمز المتغير .

POP # هذا الرمز يمثل متغير ذا قيمة هي عبارة عن عدد حقيقي بدقة مضاعفة ، لوجود علامة (#) في نهاية رمز المتغير .

SOSU93819$ هذا الرمز يمثل متغير حرفي (سلسلة) . وذلك لوجود علامة ($) في نهاية رمز المتغير .

LM%3= هذا الرمز ممثل بصورة خاطئة ، وذلك لوجود علامة (=) ، وهي إشارة حسابية .

LM%= 3.25 هذه العبارة ممثلة بصورة خاطئة ، وذلك لأن العلامة (%) تعني ان القيمة العددية يجب ان تكون عدد صحيح ، بينما العدد (3.25) يمثل عدد حقيقي .

LETI% = 86 هذه العبارة ممثلة بصورة خاطئة ، وذلك لأن الرمز يبدأ بكلمة (LET) وهي محجوزة كتعبير للحاسبة

NS123456=10.005 هذه العبارة صحيحة ، وذلك لتوافر كافة شروط رمز المتغير فيها ، اضافة الى ان القيمة العددية للمتغير هي ذات دقة مفردة (الوجود علامة (!).

R35+8%=12 هذه العبارة خاطئة لوجود اشارة (+) ضمن رمز المتغير وهذا لا يجوز وذلك لأن الرمز يفترض ان لا يحوي سوى الحروف الابجدية والارقام فقط .

NAATT$=ALT هذه العبارة ممثلة بصورة خاطئة ، وذلك لأن المتغير الحرفي (ALT) يجب ان يكون موجوداً ضمن علامتي اقتباس ، أي ("ALT") . والسبب ان رمز المتغير يعني أنه متغير حرفي وذلك لوجود علامة ($) في اخر الرمز .

NA ATT$="ALT" هذه العبارة ممثلة بصورة خاطئة ، وذلك لوجود فراغ ضمن رمز المتغير (بين حرفي(A)) وهذا لا يجوز .

10- رقم السطر :

يتكون البرنامج في لغة البيسك من سلسلة من السطور (او الخطوات) . ويجب ان نقوم بتعريف كل سطر منها برقم في بدايته وضمن الشروط التالية :

أ- يكون الرقم **عدد صحيح** (أي بدون فاصلة عشرية) **وموجب** .

ب- يكون الرقم بين (0) و (65529). وفي بعض الحاسبات يكون بين (1) و (9999) او بين (0) و (32767) .

جـ- لا يوجد سطران يحملان نفس الرقم ضمن البرنامج نفسه .

د- تسلسل تنفيذ البرنامج في الحاسبة سوف يكون ابتداءاً من الرقم الاقل وصعوداً الى اعلى رقم .

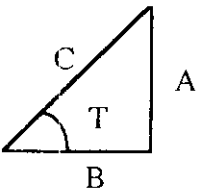
11- الوظائف :

هي مفردات معينة تُنفذ في إيجاد قيم بعض المتغيرات العددية او

الحرفية ، والتي تعبر عن علاقات رياضية او منطقية خاصة ، مثل (الجذر التربيعي) لعبارة او كمية (SQR) ، او (جيب زاوية) (SIN) ، أو ((أو) الاستثنائية) المنطقية (XOR) .. الى آخره من الوظائف . وفي الفصل التاسع نجد تفاصيل اخرى كثيرة عن مثل هذه الوظائف .

مثال :

اكتب عبارة رياضية بلغة البيسك لإيجاد قيمة الوتر (C) في المثلث القائم الزاوية ، والذي ضلعاه هما (A) و (B) .



الحل :

حسب نظرية فيثاغورس فأن :

$C^2 = A^2 + B^2$

$C = \sqrt{A^2 + B^2}$

C = **SQR** (A^2 + B^2)

مثال :

اكتب عبارة بيسك لتمثيل قيمة جيب الزاوية (T) في المثال السابق وضع الناتج باسم المتغير (X) .

الحل :

X= **SIN** (T)

12- العبارة الحسابية او الجبرية :

هي مجموعة من الثوابت او المتغيرات او الوظائف او خليط من

اكثر من واحد منها ، ترتبط مع بعضها بإشارات حسابية أو منطقية ، على ان تكون قيمة كل منها معلومة ومعرفة مسبقاً الى الحاسبة . وقد تحوي العبارة على علامة التساوي (أي =) . وعند ذاك يجب ان تكون كافة الكميات الموجودة الى يمين علامة (=) ذات قيم معلومة ، ولا يوجد الى يسار علامة (=) سوى رمز لمتغير واحد فقط ، ويمكن ان يكون هذا الرمز مجهول القيمة . وفيما يلي بعض الامثلة عن ذلك :

1) B<=5.7*Y
2) N>=(14.3-Z^2.1 +3.007*X)
3) A!=B%-C+D^2
4) C#=SQR(A^2+B^2)
5) R$=A$+"CALL"+C$+LEFT$ (A$, 5)
6) PI!=22.17 .
7) (X OR Y) >0

13- المتغيرات ذوات الرموز الدليلية :

بعض المتغيرات تأخذ اكثر من قيمة في البرنامج وتحت نفس الاسم (او الرمز) مثل X_1 و X_2 و X_3 .... X_n . واذا كان عددها محدوداً فيمكن ان نسميها (او نرمز لها) كذلك . ولكن عندما يكون عددها كبيراً فأن ذلك غير ممكن علمياً ، وذلك لأن عدد الرموز في اي حاسبة هو عدد محدود . كما ان المعالجة المتكررة للرموز من قبل الحاسبة في البرنامج تتطلب منا ان نكرر نفس الخطوات عدة مرات وذلك لأن (X) يتخذ عدة قيم . لذلك نلجأ الى طريقة اخرى في تسمية المتغير (او الرمز له) ، وذلك بأن نضع **إسم المتغير** ، ونضع بعده **رمز دليل داخل قوسين** . و**رمز الدليل** هذا سوف يتغير كي يأخذ رقم الحد المطلوب ، فيكون واحد في الحد الاول ، واثنان للرمز الى الحد الثاني . اي يمكننا ان نرمز الى المتغير السابق بالرمز X(I) ، حيث أن **الدليل** (I) يمكن أن يأخذ القيم 1, 2, 3, ..., n .

إن **المتغيرات ذوات الرموز الدليلية** تتكون من جزئين هما

الرمز الحرفي ، سواءاً كان عددياً او حرفياً يليه بين قوسين رمز متغير (او عبارة) هو **الدليل** والذي نؤشر بواسطته رقم الحد . ولا يفوتنا ان نذكر ان **الدليل** يجب ان تكون قيمته تساوي **عدد موجب وصحيح** (أي بدون فارزة او كسور عشرية) . وذلك لأن **الدليل يمثل رقم الحد** ، مثل الحد الاول او الحد الثاني ... الى آخره ، ولا يمكننا القول مثلاً الحد الاول والنصف او الحد الثاني والربع لأن ذلك ليس له معنى .

قد يكون هناك اكثر من **دليل** لنفس المتغير ونقوم بالفصل بين الادلة (جمع **دليل**) بفوارز مثال :

Y#(A%, B%, C%, ...Z%) , X! (K%, L%, F%) , S#(I%, J)