الفصل التاسع

التوسع في تعلم
مفردات لغة بيسك

1- تقديم :

لقد درسنا في الفصل السابق البرمجة بلغة بيسك ، وقمنا بتعلم كيفية كتابة البرامج بهذه اللغة . وسوف نحاول أن نتعلم المزيد من مفردات لغة بيسك ، وعن أهم أنواعها وإستخداماتها مع الأمثلة المختصرة قدر الإمكان ، وذلك لتعذر شرحها بتوسع ضمن برامج وأمثلة محلولة كما فعلنا في الفصل السابق منعاً للإطالة .

يمكن تقسيم مفردات لغة بيسك إلى سبعة أقسام رئيسية هي : **الأوامر** ، **التعابير** ، **الوظائف العددية** ، **الوظائف الحرفية**، **الوظائف المنطقية** ، **سيطرة الصيغة** ، وأخيراً **متغيرات خاصة**. وسوف ندرس تفاصيل معظم مفردات كل مجموعة منها ضمن هذا الفصل . وسنكتب صيغة عامة لكل من هذه المفردات ، كما سنقوم بإستخدام رموز مختصرة لبعض العبارات التي سنستخدمها بشكل عام في الصيغة العامة ، وسنكتب هذه الرموز بصورة حروف مائلة بينما نكتب مفردات لغة بيسك بشكل حروف كبيرة ومستقيمة . وفيما يلي قائمة برموز هذه المختصرات مع المعنى المقصود بكل رمز :

الرمز المختصر	المعنى
e	عبارة جبرية تتكون من متغير (أو عدد من ض المتغيرات) ، ثابت (أو مجموعة ثوابت) متصلة فيما بينها بعمليات حسابية (مثل + ، - ، × ، ...إلى آخره) .
v	**متغير** (حسب تعريفه بلغة بيسك) .
n	**رقم** (ويمثل عادة بعدد صحيح) .
i	**مقدار الزيادة** (يكون عادة عدد صحيح) .
s$	**سلسلة حرفية** (سواء كانت ثابتة أو متغيرة) .

l **قائمة** (من المتغيرات أو الرموز الحرفية أو الإثنين معاً).

lin **سطر** (رقم السطر في لغة بيسك) .

c **حرف** (حرف أبجدي) .

s تعبير (بلغة بيسك) .

وإذا ظهرت الحاجة لأكثر من واحد من نفس الرمز المختصر ، فسوف نقوم بترقيمه لغرض تمييزه مثل (*e2 , e1*) للدلالة على **العبارة الجبرية الأولى** ، **العبارة الجبرية الثانية** على التوالي. أو *s1$* ، *s2$* ، *s3$* للدلالة على ثلاثة سلاسل حرفية ... وهكذا إلى آخره .

2- مجموعة (الأوامر COMMANDS) :

تتقبل الحاسبة مفردات هذه المجموعة بإدخالها (أي طبعها) بشكل مباشر من لوحة المفاتيح ويلي ذلك الضغط على مفتاح الإدخال (إياب RETURN) (أو لـ) (أو ENTER) . ولانحتاج إلى رقم سطر لأنها عادة لاتدخل ضمن سطور البرنامج .

2-1- الأمر (ذاتي AUTO) :

الصيغة العامة هي :

AUTO *n,i*

ويستخدم هذا الأمر لإعادة ترقيم سطور البرنامج من جديد بشكل ذاتي (أي أوتوماتيكي) ، فتبدأ الحاسبة بترقيم السطر الأول في البرنامج بالرقم (*n*) ، وتزيده في السطر التالي بمقدار (*i*) ، والسطر

الذي يليه بزيادة (i) أخرى على رقم السطر السابق له . ويستخدم هذا الأمر عند ظهور الحاجة لإضافة سطور أخرى للبرنامج الأصلي . أو لإعادة ترقيم البرنامج لغرض دمجه مع برنامج آخر يكون ترقيمه مختلف، أو لأي غرض آخر مشابه .

2-2- الأمر (يزيل CLEAR) :

الصيغة العامة هي :

```
CLEAR e
```

حيث إن (e) هنا إختيارية وليست إجبارية) ، و عند إستخدامها تجعل حيز البيانات أو الحروف في ذاكرة الحاسبة بقدر قيمة الرمز (e) مثل :

```
CLEAR 20
```

أما إذا لم نستخدم الرمز (e) ، فإن هذا الأمر يستخدم لإزالة كافة قيم متغيرات البرنامج من الذاكرة، وجعل قيم كل منها (0) أو (فراغ) .

2-3- الأمر (تحميل شريط CLOAD) :

الصيغة العامة هي :

```
CLOAD s$
```

ويستفاد من هذا الرمز في تحميل البرنامج من الشريط المغناطيسي (الكاسيت) ، ونقله إلى ذاكرة الحاسبة ليصبح جاهزاً للتنفيذ. ويمثل الرمز (s) إسم البرنامج المقصود بعملية التحميل .

4-2- الأمر (حفظ شريط CSAVE) :

الصيغة العامة هي :

CSAVE *s$*

ويستخدم هذا الأمر في نقل البرنامج من ذاكرة الحاسبة إلى الشريط المغناطيسي (الكاسيت) لغرض خزنه (أو حفظه) .ويمثل الرمز (*s$*) إسم البرنامج المقصود بعملية الحفظ .

5-2- ألأمر (مستمر CONT):

الصيغة العامة هي :

CONT

ويستخدم هذا الأمر في الإستمرار في تنفيذ البرنامج بعد توقفه تنفيذاً للأمر (توقف **STOP**) أو تنفيذاً للتعبير (نهاية **END**) .

6-2- الأمر (شطب DELETE)

الصيغة العامة هي :

DELETE *n1 - n2*

بإستخدام هذا الأمر يتم شطب مجموعة من السطور ، من البرنامج الموجود في الذاكرة ، ويتم الشطب من السطر المرقم (*n1*) ولغاية السطر المرقم (*n2*) ، ويمكن أن يكون هذا الأمر بصيغ مختلفة كما في الأمثلة التالية:

1- **مثال** : **DELETE** 45 - 200

أي شطب مجموعة من السطور في البرنامج إبتداءاً من السطر المرقم (45) وإنتهاءاً بالسطر المرقم (200) .

2- **مثال** : **DELETE** 130

يتم بموجب هذا الأمر شطب السطر (130) فقط .

3- **مثال** : **DELETE** 90 -

أي شطب كل السطور في البرنامج إبتداءاً من السطر المرقم (90) ولغاية آخر سطر في البرنامج .

4- **مثال** : **DELETE** -120

أي شطب كل السطور في البرنامج إبتداءاً من السطر الأول فيه ولغاية السطر (120) .

5- **مثال** : **DELETE**

بموجب هذا الأمر يتم شطب كل البرنامج من ذاكرة الحاسبة .

7-2- الأمر (قائمة LLIST) :

الصيغة العامة هي :

LLIST *n1 - n2*

بهذا الأمر نطلب من الحاسبة قائمة (تظهر على الشاشة) بسطور البرنامج الموجود في ذاكرتها ، إبتداءاً من السطر المرقم (*n1*) وإنتهاءاً

بالسطر المرقم (*n2*) . ونفس الأمثلة التي أوردناها في الفقرة السابقة عن الأمر (شطب DELETE) حول الصيغ التي يمكن أن يكون عليها . تنطبق على الصيغ التي يمكن أن يكون عليها هذا الأمر أيضاً .

2-8- الأمر (قائمة LLIST):

الصيغة العامة هي :

LLIST *n1 - n2*

وهو مشابه تماماً للأمر (قائمة LIST) ، والفرق الوحيد هو إن القائمة هنا تظهر بشكل مطبوع على الورق بدلاً من ظهورها على الشاشة .

2-9- الأمر (جديد NEW) :

الصيغة العامة هي :

NEW

هذا الأمر يستخدم لمسح كل البرنامج الموجود في ذاكرة الحاسبة، وكذلك لمسح كافة قيم المتغيرات ، ونستخدمه عادة قبل إدخال برنامج جديد إلى الحاسبة .

2-10- الأمر (أعد الترقيم RENUM) :

الصيغة العامة هي :

RENUM *n1 , n2 , i*

يستخدم هذا الأمر لإعادة ترقيم البرنامج إبتداءاً من السطر المرقم (*n2*) في البرنامج الموجود في ذاكرة الحاسبة ، حيث يستبدل هذا الرقم بالرقم (*n1*) . ومقدار الزيادة بين كل سطر وآخر هو قيمة (*i*) ، وإذا قمنا بحذف (*i*) فإن الحاسبة سوف تفترضه (10) . أما إذا حذفنا (*n2*) فإن الحاسبة سوف تبدأ بإعادة الترقيم إبتداءاً من أول سطر فيه .

11-2- الأمر (نَفِّذْ RUN) :

الصيغة العامة هي :

RUN *n*

بواسطة هذا الأمر نطلب من الحاسبة تنفيذ البرنامج المخزون في ذاكرتها إبتداءاً من السطر المرقم (*n*) .أما إذا كان (*n*) محذوف فإن الحاسبة سوف تنفذ البرنامج إبتداءاً من أول سطر فيه .

12-2- الأمر (إفتح الإقتفاء TRON) :

الصيغة العامة هي :

TRON

عند إستخدامنا لهذا الأمر تقوم الحاسبة بطبع رقم كل سطر يتم تنفيذة أثناء تشغيل البرنامج . أو بمعنى آخر إقتفاء أثر تنفيذ البرنامج.

13-2- الأمر (أطفىء الإقتفاء TROFF) :

الصيغة العامة هي :

```
TROFF
```

تقتصر وظيفة هذا الأمر على إلغاء الأمر السابق (إفتح الإقتفاء TRON) .

14-2- الأمر (عُرض WIDTH) :

الصيغة العامة هي :

```
WIDTH n
```

نستطيع بإستخدام هذا الأمر تحديد عُرض السطر ، في تعبير (إطبع PRINT) ، بعدد محدد من المواضع تساوي قيمته (*n*) حرف .

3- مجموعة (التعابير STATEMENTS) :

1-3- تعبير (إستدعيCALL)

الصيغة العامة هي :

```
n CALL e
```

هذا التعبير يتسبب في تفرع برنامج البيسك الرئيسي إلى برنامج ثانوي مكتوب بلغة الماكنة ، ومخزون في موضع ذاكرة معين ، يتحدد عنوانه بقيمة العبارة (*e*) في الصيغة العامة . وإذا كان البرنامج الثانوي ينتهي بتعبير (إياب RETURN) ، فإنه وبعد الإنتهاء من تنفيذ

البرنامج الثانوي ، سوف ينتقل لتنفيذ السطر التالي للتعبير (إستدعي CALL)، والمرقم (*n*) في الصيغة العامة والموجود في البرنامج الرئيسي المكتوب بلغة بيسك .

3-2- تعبير (تحويل إلى دقة مضاعفة CDBL) :

الصيغة العامة هي :

```
n  CDBL v
```

يفيد هذا التعبير في تحويل قيمة المتغير (*v*) في الصيغة العامة، إلى قيمة بدقة مضاعفة .

3-3- تعبير (تحويل إلى عدد صحيح CINT) :

الصيغة العامة هي :

```
n  CINT v
```

يستخدم هذا التعبير في تحويل قيمة المتغير (*v*) إلى عدد صحيح وذلك بحذف الفارزة العشرية وما بعدها من أرقام إن وجدت .

3-4- تعبير (تحويل إلى دقة مفردة CSNG)

الصيغة العامة هي :

```
n  CSNG v
```

هذا التعبير يستخدم في تحويل قيمة المتغير (*v*) إلى عدد عشري

بدقة مفردة (أو إعتيادية) .

3-5- تعبير (بيانات DATA) :

الصيغة العامة هي :

n* DATA *I

حيث تمثل (*I*) قائمة البيانات المعطاة في هذا التعبير (سواء كانت إعتيادية أو حرفية) ، ويفصل بين كل واحد والآخر فوارز . وتقابل قائمة البيانات هذه رموز المتغيرات التي قمنا بوضع قائمة بها في التعبير (إقرأ **READ**) . كما في المثال التالي :

130 **DATA** 10 , 20 , 48 , "FIRST" , "SECOND"

3-6- تعبير (إقرأ READ) :

الصيغة العامة هي :

n* READ *I

حيث تمثل (*I*) هنا ، قائمة بالمتغيرات التي نضع قيمتها (العددية أو الحرفية) في أول تعبير (بيانات **DATA**) يلي هذا السطر والمرقم (*n*) بحيث أن أول رمز لمتغير في هذا التعبير يقابل أول قيمة في تعبير (بيانات DAT)، وثاني رمز يقابل ثاني قيمة ، وهكذا إلى آخره . ولو أردنا على سبيل المثال وضع تعبير (إقرأ **READ**) مناسب للمثال الوارد في الفقرة السابقة أعلاه ، فيمكننا كتابته كما يلي :

120 **READ** A , B , C , D$, E$

7-3- تعبير (أعد خزن RESTORE)

الصيغة العامة هي :

n **RESTORE**

نتمكن بواسطة إستخدام هذا التعبير من إعادة قراءة البيانات DATA.. وتعبير (إقرأ READ) الذي يلي تعبير (أعد خزن RESTORE)، سوف يأخذ قيم متغيراته من أول تعبير (بيانات DATA) موجود في البرنامج .

8-3- تعبير (عرّف عدد صحيح DEFINT) :

الصيغة العامة هي :

n **DEFINT** *c1-c2*

بإستخدام هذا التعبير فإن كافة رموز المتغيرات التي تبدأ بأحد حروف الأبجدية الأنكليزية ، والواقعة بين الحرف (*c1*) والحرف (*c2*) ، سوف تعتبر من قبل الحاسبة كرموز لأعداد صحيحة . كما في المثال التالي :

40 **DEFINT** B-F

أي إن كافة الرموز للمتغيرات والتي تبدأ أسماؤها بأحد الحروف (B ، C ، D ، E ، F) سوف تعامل كرموز لكميات صحيحة (أي بدون فارزة عشرية) .

9-3- تعبير (عرّف دقة مفردة DEFSNG) :

الصيغة العامة هي :

n **DEFSNG** *c1 - c2*

بإستخدام هذا التعبير فأن كافة رموز المتغيرات التي تبدأ رموزها بأحد الحروف بدءاً من (*c1*) وحتى (*c2*) في الأبجدية الأنكليزية ، سوف يتم إعتبارها كمتغيرات ذات قيمة بدقة مفردة (إعتيادية) .مثال:

60 **DEFSNG** H - P

10-3- تعبير (عرّف بدقة مضاعفة DEFDBL) :

الصيغة العامة هي :

n **DEFDBL** *c1 - c2*

كافة رموز المتغيرات التي تبدأ رموزها بأحد الحروف بدءاً من (*c1*) وحتى (*c2*) في الأبجدية الأنكليزية ، سوف يتم تعريفها كمتغيرات ذات قيمة بدقة مضاعفة . مثال :

80 **DEFDBL** R - W

11-3- تعبير (عرّف سلسلة DEFSTR) :

الصيغة العامة هي :

n **DEFSTR** *c1 - c2*

كافة المتغيرات التي تبدأ رموزها بأحد الحروف بدءاً من (*c1*) وحتى (*c2*) في الأبجدية الأنكليزية ، سوف يتم تعريفها كرموز لمتغيرات حرفية (التي تكون على شكل سلسلة من الحروف والرموز والأرقام) . مثال :

```
100 DEFSTR X- Z
```

12-3- تعبير (أبعاد DIM) :

الصيغة العامة هي :

n DIM v 1(n1,n2,...), v 2(n1,n2,..),..., s$ (n), ...

بإستخدام هذا التعبير يمكننا حجز مواضع فارغة في الذاكرة لمتغيرات ذات رموز دليلية سواءاً كانت أحادية الأبعاد ، أو ثنائية الأبعاد (مصفوفة) أو ذات أبعاد أكثر من ذلك ، وكحد أقصى (200) عنصر لكل بعد (أي دليل) . ولمزيد من التفاصيل عن المتغيرات ذات الرموز الدليلية راجع الفقرة (13) من الفصل السابع . وفيما يلي مثال عن هذا التعبير :

```
10 DIM X(6 , 8 , 2) , Y(4 , 4) , Z$ (20)
```

13-3- تعبير (نهاية END) :

الصيغة العامة هي :

***n* END**

يستخدم هذا التعبير لإنهاء تنفيذ البرنامج . ويكون عادة في السطر الأخير للبرنامج الرئيسي .

14-3- تعبير (الأجل FOR) :

الصيغة العامة هي :

n FOR v =e1 TO e2 STEP e3

هذا التعبير هو بداية حلقة التكرار من نوع (الأجل... التالي... FOR...NEXT) ، حيث توضح قيمه رمز المتغير (*v*) في الصيغة العامة مساوية للعبارة (*e1*) كقيمة إبتدائية ، وتزداد في كل مرة بمقدار قيمه العبارة (*e3*) ، وفي كل مرة يتم تنفيذ كل السطور رحتى نصل إلى التعبير (التالي NEXT) الخاص بهذه الحلقة . ويستمر ذلك حتى تصل قيمة المتغير (*v*) أكبر من قيمة العبارة (*e2*) ، وآنذاك ستتوقف الحاسبة عن تنفيذ السطور الموجودة في حلقة التكرار هذه . مثال :

```
20 FOR RAD = 2 TO 50 STEP 5
```

15-3- تعبير (التالي NEXT) :

الصيغة العامة هي :

n NEXT v1 , v2 , ...

هذا التعبير يستخدم لإنهاء حلقة التكرار ، والتي تبدأ بتعبير (الأجل FOR) ، أما (*v1*) فهي تمثل رمز المتغير المستخدم كدليل لأقرب حلقة تكرار ، وتمثل (*v2*) رمز للمتغير المستخدم كدليل لحلقة التكرار الأبعد... إلى آخره . مثال :

```
315 NEXT RAD
```

أما إذا كان تعبير (التالي NEXT) لايحوي أي رمز لمتغير (مثل

v1 , v2...) في الصيغة العامة ، فأن مثل هذا التعبير سوف يقوم بإنهاء حلقة التكرار الأكثر قرباً اليها في البرنامج .

16-3- تعبير (إذهب ثانوي GOSUB) :

الصيغة العامة هي :

n **GOSUB** ***lin***

يستخدم هذا التعبير للتفرع من البرنامج الرئيسي إلى برنامج ثانوي مكتوب بلغة بيسك . و يبدأ البرنامج الثانوي بالسطر المرقم (*lin*) في الصيغة العامة مثال :

100 **GOSUB** 2000

17-3- تعبير (إياب RETURN) :

الصيغة العامة هي :

n **RETURN**

يكون هذا التعبير هو السطر الأخير في البرنامج الثانوي ، و يتسبب في إعادة تنفيذ البرنامج إلى السطر الذي يلي السطر الذي قمنا بواسطته التفرع إلى البرنامج الثانوي (أي الذي يحوي تعبير (إذهب ثانوي GOSUB) في البرنامج الرئيسي .

18-3- تعبير (إذهب إلى GO TO) :

الصيغة العامة هي :

n **GO TO** ***lin***

15 **IF** Z<5 **THEN** 100 **ELSE** PRINT "ERROR"

21-3- تعبير (إدخال INPUT) :

الصيغة العامة هي :

n* INPUT *l

يستخدم هذا التعبير لإدخال البيانات إلى ذاكرة الحاسبة بواسطة لوحة المفاتيح ، ويتم تحديد كل منها في قائمة المتغيرات (*l*) ، ويتم الفصل ما بين البيانات بفوارز إذا كانت القيم عددية . أما عند إدخال المتغيرات الحرفية (أي من نوع السلسلة) ، فيجب أن نفصل بينها بواسطة الضغط على مفتاح (RETURN) و (ENTER) أو (↵) في لوحة المفاتيح . **مثال :**

20 **INPUT** S,T, X,Y$, Z$

22-3- تعبير (دع LET) :

الصيغة العامة هي :

n* LET *v* = *e

يستخدم هذا التعبير لتحديد قيمة المتغير (*v*) ، سواءاً كانت على شكل قيمة عددية أو على شكل عبارة جبرية (تكون قيم كل عناصرها معلومة) . **مثال :**

20 **LET** X = Y+10.5*Z

90 **LET** W = 20.5 او **مثال :**

هذا التعبير يتسبب في نقل تنفيذ البرنامج إلى
(*lin*) **مثال** : 110 **GO TO** 900

19-3- تعبير (إذا ....آنذاك.....IF...THEN...) :

الصيغة العامة هي :

n **IF** *e* **THEN** *s*

معنى هذا التعبير أنه إذا كانت العبارة (*e*) حقيقية ، آنذاك نفذ التعبير (*s*) . وربما يكون هناك (*lin*) بدلاً من (*s*) في الصيغة العامة . وذلك معناه أنه إذا كانت العبارة حقيقيةَ ، آنذاك ننتقل لتنفيذ التعبير (*lin*) . وإذا لم يتحقق ذلك (أي أن (*e*) تكون زائفة) فيجب أن يستمر التنفيذ الإعتيادي للبرنامج ؛أي ينتقل إلى السطر التالي .

مثال : 20 **IF** X>Y **THEN** PRINT A

أو **مثال** : 90 **IF** X>10 **THEN** 50

20-3- تعبير (إذا...آنذاك...وإلا...IF...THEN...ELSE) :

الصيغة العامة هي :

n **IF** *e* **THEN** *s1* **ELSE** *s2*

هذا التعبير مشابه للتعبير السابق ، ولايوجد فرق بينهما سوى أنه في حالة كون العبارة (*e*) زائفة (غير حقيقية) ، فأنه سوف يتم الإنتقال إلى التعبير الثاني (*s2*) أو إلى رقم سطر آخر (أي *lin 2*) . **مثال** :

23-3- تعبير (إدخال سطر LINE INPU) :

الصيغة العامة هي :

[redacted]

عند استخدام هذا التعبير يظهر لنا("النص") على الشاشة (وهو هنا إختياري) ، ونستطيع بعد ذلك (بواسطة لوحة المفاتيح) إدخال أي سلسلة من الحروف والرموز والارقام (والتي مثلناها بالرمز (*s$*) في الصيغة العامة) .

24-3- تعبير (عندٌ . . . إذهب ثانوي . . . GOSUB . . .ON):

الصيغة العَامة هي :

[redacted]

هذا التعبير يعني أنه إذا كانت القيمة العددية للعبارة (*e*) تساوي (1) فإذهب الى البرنامج الثانوي والذي يكون رقم اول سطر فيه هو(*n1*) . آما اذا كانت القيمة العددية للعبارة (*e*) تساوي (2) فسوف تذهب الحاسبة لتنفيذ البرنامج الثانوي الذي يكون رقم اول سطر فيه (*n2*) - - - وهكذا وصولاً الى آخر متغير موجود في القائمة . **مثال** :

40 **ON J GOSUB** 1000,3000,5000,7000,9000

وهذا معناه أنه إذا كانت قيمة المتغير (J) تساوي (1) فسوف ينتقل البرنامج لتنفيذ البرنامج الثانوي الذي يكون اول سطر فيه مرقم (1000) . اما إذا كانت (J) تساوي (2) فسينتقل الى البرنامج الثانوي (3000) . وإذا كانت (J) تساوي (3) فاذهب الى (5000) . وفي حالة كون

(J) تساوي (4) فإذهب الى (7000) . اما في حالة (J) تساوي (5) فإذهب الى (9000) .

3-25- تعبير (عندَ . . . إذهب الى . . . GOTO . . . ON) :

الصيغة العَامة هي :

n ON e GOTO n1,n2,n3, . . .

وهذا التعبير يشبه التعبير السابق له ، والفرق الوحيد هو إن التنفيذ ينتقل الى سطور في البرنامج الرئيسي والمرقمة *n3,n2,n1* . . .الى آخره . وليس الى برامج ثانوية كما شرحناه في الفقرة السابقة .

3-26- تعبير (عند الخطأ إذهب الى ON ERROR GOTO) :

الصيغة العامة هي :

n ON ERROR GOTO l

في حالة حصول اي خطأ اثناء تنفيذ البرنامج الذي يحوي مثل هذا التعبير ، فان التنفيذ سوف ينتقل الى السطر المرقم (*l*) . **مثال :**

120 **ON ERROR GOTO** 78

3-27- تعبير (إطبع PRINT) :

الصيغة العامة هي :

n PRINT . . . . . .

نستطيع بواسطة إستخدام هذا التعبير ان نطبع القيم للمتغيرات ، سواداً كانت عددية (*v*) ، او كعبارة (*e*) ، او كمتغير حرفي (*s$*) .
مثال :

```
110 PRINT A,B,C+5, D$
```

ولا يفوتنا هنا ان نذكر انه اذا كان السطر لا يحوي سوى تعبير **PRINT** (آي لا توجد رموز متغيرات بعدها) ، فأن الحاسبة سوف تقوم بترك سطر فارغ . **مثال :**

```
20 PRINT
```

28-3- تعبير (إطبع على الورق LPRINT) :
الصيغ العامة هي :

n* LPRINT *v*, - - -, *e*, - - -, *s$

وهذا التعبير مشابه للتعبير السابق . والفرق الوحيد بينهما هو ان الطباعة بموجب هذا التعبير تتم على الورق ، وليس على الشاشة .
مثال :

```
210 LPRINT A,B,C+5 , S$
```

29-3- تعبير (اطبع مستخدماً PRINT USING) :
الصيغة العامة هي :

n* PRINT USING *s$,e

وهذا التعبير يستخدم لاغراض خاصة ، حيث نتمكن بإستخدامه تحديد (صيغة) لطباعة أي متغير او عبارة جبرية ، بحيث تتحدد

الطباعة ضمن عمود ثابت وعدد معين من المواضع ، وتظهر النتائج منسقة على شكل جدول . **مثال :**

```
125 PRINT USING " $$###.## " , A
```

حيث تمثل الاشارة (#) موضع يتسع لرقم واحد فقط ، وتمثل ($) موضع يتسع لحرف واحد فقط . ولو افترضنا ان هذا التعبير استخدم لطباعة قيم المتغير (A) ، والتي تكون على شكل عدد عشري يحوي ثلاثة ارقام قبل الفارزة كحد اقصى واثنين بعد الفارزة . ونفترض ان ($) ستمثل موضع للحرفين KG كرمز للكيلوغرام . فيمكن ان تظهر النتائج بالشكل التالي :

```
KG 315.20
KG8.00
KG15.00
KG999.99
```

4- مجموعة (الوظائف العددية NUMERIC FUNCTIONS):

عند استخدام مفردات هذه المجموعة ، فان الحاسبة تقوم بتنفيذ عمليات معينة تفيدنا في حل معادلة حسابية معينة ، او إنجاز وظيفة رياضية معينة . ، مثل ايجاد الجذر التربيعي لمتغيرٍ ما ، او ايجاد جيب تمام زاوية معينة - - - الى آخره من الوظائف الحسابية الأخرى . وأهم عناصر هذه المجموعة هي :

4-1- (المطلق ABS):

الصيغة العامة هي :

```
ABS (e )
```

وهذه الوظيفة تفيدنا في حساب (او ايجاد) القيمة المطلقة لقيمة المتغير او العبارة (*e*) في الصيغة العامة . والقيمة المطلقة تعني الرقم بدون إشارة (سواءاً سالبة او موجبة) . **مثال** :

```
50 IF ABS (x) > 9 THEN 120
```

4-2- (ظل المماس ATN) :

الصيغة العامة هي :

```
ATN (e)
```

هذه الوظيفة نستطيع بواسطتهاإيجاد ظل المماس (معكوس الظل) للزاوية ، على ان نتذكر ان تلك الزاوية ستكون بالمقياس الدائري، وليس بالمقياس الستيني . وتتراوح الزاوية بين(ط/2) إلى(- ط/2)، حيث تمثل (ط) النسبة الثابتة (اي 22 ÷ 7) . **مثال** :

```
130 PRINT ATN (B)
```

4-3- (جيب تمام COS) :

الصيغة العامة هي :

```
COS (e)
```

بإستخدام هذه الوظيفة نتمكن من حساب جيب تمام الزاوية (*e*) في الصيغة العامة . ويجب ان تكون الزاوية بالمقياس الدائري (وليس الستيني) . كما ذكرنا في الفقرة السابقة عن (ظل المماس ATN) .

4-4- (ثابت FIX) :

الصيغة العامة هي :

FIX (*e*)

نستخدم هذه الوظيفة لاستخراج العدد الصحيح (أي العدد الموجود قبل الفارزة العشرية) من العدد العشري الذي يمثل قيمة العبارة (*e*) في الصيغة العامة . فلو كانت قيمة (*e*) تساوي العدد العشري (2.793) فأن قيمة (*e*) **FIX** تساوي (2) فقط . **مثال** :

100 FIX (A / 17)

4-5- (داخل سلسلة INSTR) :

الصيغة العامة هي :

INSTR (*n*,*s1$* , *s2$*)

هذه الوظيفة تفيدنا في البحث عن متغير حرفي (سلسلة) والذي نرمز له بـ (*s2$*) ، وهذا المتغير موجود داخل (او ضمن) متغير حرفي (سلسلة) اخر هو (*s1$*) ، ويمثل الرقم (*n*) موضع البحث (اي رقم الحرف) في المتغير (*s1$*) والذي نطلب من الحاسبة البدء منه في البحث. فاذا كان (*n*) يساوي (1) فأن البحث سوف يبدأ من الحرف الاول في (*s1$*) واذا كانت (*n*) تساوي (2) فأن البحث سوف يبدأ بالحرف الثاني من (*s2$*) . واذا كانت نتيجة البحث وجود (*s2$*) ضمن (*s1$*) فأن الحاسبة ستضع قيمة عددية (تمثل رقم موضع التطابق) مقابل الوظيفة **INSTR** (*n* , *s1$* , *s2$*) ، اما إذا كانت (*s2$*)غير موجودة ضمن *s1$*)

(فسوف تكون قيمة الوظيفة تساوي (صفر). **مثال:**

```
250 IF INSTR (A,B$, C$)>0 THEN PRINT "GOT IT"
```

4-6- (صحيح INT) :

الصيغة العامة هي :

```
INT (e)
```

هذه الوظيفة تفيدنا في الحصول على اقرب (سواءاً اقل او يساوي) عدد صحيح للمتغير او العبارة (*e*) **مثال** :

```
15 A=INT (B/2.75)
```

4-7- (طول LEN) :

الصيغة العامة هي :

```
LEN (s$)
```

يمكننا بإستخدام هذه الوظيفة من الحصول على طول متغير حرفي (سلسلة حرفية)، اي عد د الحروف المكونة له . مثال :

```
112 IF LEN (A$)> 30 THEN PRINT " TOO LONG"
```

4-8- (الباقي MOD) :

الصيغة العامة هي :

```
e1 MOD e2
```

بإستخدام هذه الوظيفة نحصل على الباقي فقط من قسمة (*e1*) على (*e2*) . فعلى سبيل المثال لو كان لدينا المتغير (A) ، وارد نا ان نعرف ان كان هذا المتغير عدد زوجي ، فنقوم بإيجاد الباقي من قسمته على (2) ، فاذا كان يساوي (0) فهذا يعني أن قيمة المتغير (A) هي كمية زوجية ، ويمكن ان نمثل ذلك كما يلي :

```
200 IF (A MOD 2)=0 THEN PRINT "EVEN"
```

4-9-(إشارة SGN) :

الصيغة العامة هي :

SGN (*e*)

نستخدم هذه الوظيفة في إيجاد الإشارة لقيمة ألعبارة (*e*) . فتكون (-1) اذا كانت (*e*) < 0 ، وتكون 0 اذا كانت قيمة (*e*) كذلك . وتكون (+1) اذا كانت العبارة (*e*) > 0 . **مثال :**

```
100  A=INT (A+0.5 * SGN (A))
```

4-10- (جيب SIN) :

الصيغة العامة هي :

SIN (*e*)

هذه الوظيفة تفيدنا في إيجاد جيب الزاوية (*e*) ، والتي يجب ان تكون مقاسة بالمقياس الدائري وليس بالمقياس الستيني . **مثال :**

```
150 IF SIN(x)>0.2 THEN 310
```

11-4- (الجذر التربيعي SQR) :

الصيغة العامة هي :

SQR (*e*)

بإستخدام هذه الوظيفة نستخرج قيمة الجذر التربيعي للعبارة (*e*) .

مثال :

```
100 A=SQR(B*C)
```

12-4- (ظل TAN) :

الصيغة العامة هي :

TAN (e)

هذه الوظيفة تفيدنا في إستخراج قيمة ظل الزاوية (*e*) ، على ان لا ننسى ان (*e*) تكون بالمقياس الدائري ، وليس الستيني ، كما في الوظائف المشابه الاخرى التي سبق ان شرحناها في فقرات سابقة . **مثال**:

```
20 IF TAN(A)>1 THEN 130
```

13-4- (قيمة VAL) :

الصيغة العامة هي :

VAL (s$)

هذه الوظيفة تفيدنا في إستخراج القيمة العددية لمتغير حرفي (سلسلة)؛تكون من أرقام فقط بالطبع.وإذا كان المتغير الحرفي(السلسلة) يحوي أي شيء آخر غير الارقام (كالحروف والاشارات) . فأن قيم(s)VAL عند ذاك تساوي (صفر) . **مثال** :

110 **IF VAL**(C$)<35 **THEN** 180

5- مجموعة (الوظائف الحرفية STRING FUNCTIONS)

ان مفردات هذه المجموعة تفيدنا في التعامل مع المتغيرات الحرفية (اي سلسلة من الحروف والارقام والرموز) وتسهل علينا معالجتها والتعامل معها ، ومن اهم مفردات هذه المجموعة ما يلي :

5-1- (حرف سلسلة CHR$):

الصيغة العامة هي :

CHR$ (e)

هذه الوظيفة نستخدمها لإيجاد الحرف المقابل لقيمة العبارة (e) حسب **جفرة آسكي** ASCII . (راجع الجدول الفصل) ولهذه الوظيفة استخدامات اخرى معينة لا يسعنا تعدادها . **مثال** :

370 **PRINT CHR$** (A+5)

5-2- (يسار سلسلة LEFT$) :

الصيغة العامة هي :

LEFT$(S$, e)

هذه الوظيفة تستخدم لإستخراج عدد محدد من الحروف قيمته (*e*) وتقع الى اقصى اليسار من متغير حرفي (*s$*) معين **مثال** :

```
420 B$=LEFT$ (A$,5)
```

ولو فرضنا ان (A$) تساوي (RSTUVWXY) ، فالسطر(420) اعلاه يعني (B$) يساوي اول (5) حروف من يسار المتغير (A$) . أي (RSTUV)

5-3- (وسط سلسلة MID$) :

الصيغة العامة **الاولى** هي :

```
s2$ =MID$(s1$ , e1,e2 )
```

تستخدم هذه الوظيفة لإيجاد متغير حرفي جديد طوله (*e2*) من الحروف ، إبتداءاً من الحرف المرقم (*e1*) في المتغير الحرفي الاساسي والذي رمزه هو (*s1$*) . **مثال** :

```
100 X$= MID$("ABCDEFGHIJKLMN",Y+3,5)
```

فلو فرضنا ان (Y) تساوي (1) فأن هذا معناه إن (*e1*) يساوي (4) وهو يمثل رقم الحرف الاول في السلسلة الحرفية التي تكون الرمز (Y$) . أما طول هذه السلسلة الحرفية فيتحدد بالرقم (5) والذي يقابل الرمز (*e2*) في الصيغة العامة . أي أن (X$) يساوي ("DEFGH")

الصيغة العامة **الثانية** هي :

```
MID$(s1$ ,e1,e2)= s2$
```

هذه الوظيفة تقوم بعكس العمل الذي شرحناه في الصيغة العامة

الاولى اعلاه ، فهي تقوم بإستبدال جزء من الرمز الحرفي (*s1$*) إبتداء من الحرف (*e1*) فيها ، بطول (*e2*) من الحروف ، بالمتغير الحرفي (*s2$*). فلو أفترضنا أن (*s1$*) يساوي ("ABCDEFGHIJKLMN") ، وقيمة (*e2*) تساوي (6) وقيمة (*e1*) تساوي (2) وكان المتغير الحرف (*s2$*) يساوي ("PQRSTU") فعند ذاك يكون ناتج العبارة التالية :

```
110 MID$(S1$,2,6)=S2$
```

ويكون بذلك كما يلي :

"APQRSTUHJIJKLMN"

5-4- (يمين سلسلة RIGHT$) :

الصيغة العامة هي :

RIGHT$(s$,e)

وهذه الوظيفة مشابهة جداً للوظيفة (يسار سلسلة **LEFT$**) (الفقرة (5-2) اعلاه) . ولكن في هذه الوظيفة فإننا نأخذ الحروف من أقصى اليمين . **مثال** :

```
450 B$=RIGHT$(A$,5)
```

فإذا كانت (A$) تساوي ("RSTUVXYZ") ، عند ذاك يعني السطر(450) اعلاه إن :

```
B$=("UVXYZ")
```

5-5-(فراغ سلسلة SPACE$) :

الصيغة العامة هي :

SPACE$ (e)

تستخدم هذه الوظيفة لتكوين سلسلة من الفراغات عددها يساوي قيمة العبارة (*e*) . **مثال** :

```
75 A$=B$+SPACE$ (C+5)
```

6-5- (سلسلة حرفية STRING$) :

الصيغة العامة هي :

STRING$(*e*,*s$*)

هذه الوظيفة تفيدنا في تكوين سلسلة من الحروف المكررة (*s$*)، ويكون طولها مساوياً لقيمة العبارة (*e*) . **مثال** :

```
30 PRINT STRING$(A,B$)
```

6-مجموعة(الوظائف المنطقية LOGICAL FUNCTIONS):

مفردات هذه المجموعة تفيدنا في حل المسائل التي تكون عواملها منطقية وليست حسابية (كما مر بنا في مجموعة الوظائف العددية) . وأهم عناصر هذه المجموعة الوظائف التالية :

6-1- ((و)AND) :

الصيغة العامة هي :

e1* AND *e2

القيمة المنطقية للصيغة السابقة تكون (حقيقية) ، اذا كانت العبارتين المنطقيتين (*e1*) و (*e2*) كلاهما حقيقيتين . وعدا ذلك تكون القيمة المنطقية (زائفة) . اما اذا كانت قيم العبارتين (*e1*) و (*e2*)

رقمية (وليست منطقية) ، فعند ذاك يتم الحساب لكل موضع ثنائي (بايت) مع ما يقابله . **مثال** :

```
50 IF X>Y AND C<Z THEN 200
```

6-2- ((أو) OR) :

الصيغة العامة هي :

e1* OR *e2

تكون القيمة المنطقية للصيغة العامة (حقيقية) اذا كانت أي من العبارتين المنطقيتين (*e1*) و (*e2*) كمية منطقية (حقيقية) . عدا ذلك تكون قيمة الصيغة العامة (زائفة) . مثال:

```
100 IF X<6 OR Y>8 THEN PRINT "OK"
```

6-3- ((أو) الاستثنائية XOR) :

الصيغة العامة هي :

e1* XOR *e2

القيمة المنطقية للصيغة اعلاه تكون (حقيقية) ، أذا كانت قيمة العبارة (*e1*) عكس قيمة العبارة (*e2*) ، اي اذا كانت احداهما (حقيقية) والاخرى (زائفة) . **مثال** :

```
70 IF X<6 XOR Y>5 THEN 120
```

4-6- (NOT (لا)) :

الصيغة العامة هي :

```
NOT e
```

تستخدم هذه الوظيفة للحصول على القيمة المعكوسة للعبارة المنطقية (*e*)، وإذا كانت قيمة العبارة (*e*) عددية (وليست منطقية) . فإن ذلك معناه انه ستعطينا المتمم العددي للعبارة (*e*) . حيث سيتم التعامل معها بصيغة عدد ثنائي (بت) وسنحصل في هذه الحالة على المتمم المنطقي لها بصيغة عدد ثنائي ايضاً .

7- مجموعة (سيطرة الصيغة FORMAT CONTROL) :

مفردات هذه المجموعة تستخدم في السيطرة على ادخال/اخراج البيانات ، وسنأخذ مثالين عنها ، وهما :

7-1- (فراغ SPC) :

الصيغة العامة هي :

```
SPC (n)
```

نستخدمه فقط في تعبير (إطبع PRINT) ، وذلك بطبع (*n*) من الفراغات . **مثال** :

```
35 PRINT SPC(15) ;X$
```

2-7- (جَدْوَلَة TAB) :

الصيغة العامة هي :

TAB (*e*)

يستخدم مع تعبير (أطبع PRINT) لتحريك المؤشر على الشاشة الى العمود المطلوب ، الذي تُحدَدْ قيمته بالعبارة (*e*) او ما شابه ذلك من الاستخدامات . علماً بأن العمود الاول (الحرف الاول) موجود في اقصى يسار الشاشة وتكون قيمته (0) . **مثال** :

```
84 PRINT TAB(A); A$ ; TAB (A+15) ;B$
```

8- مجموعة (المتغيرات الخاصة SPECIAL VARIABLS):

مفردات هذه المجموعة تستخدم لأغراض خاصة في إدخال البيانات وسنأخذ مثالين عنها ، هما :

8-1- (مفتاح سلسلة INKEY$) :

الصيغة العامة هي :

INKEY$

يستخدم هذا المتغير لإدخال حرف واحد فقط عن طريق الضغط على اي مفتاح من لوحة المفاتيح ، ودون الحاجة ان نضغط على مفتاح (إ ياب RETURN) (او (ENTER)) . **مثال** :

```
55 X$=INKEY$
```

8-2- (ادخال سلسلة INPUT$) :

الصيغة العامة هي :

INPUT$(*n*)

نستخدم هذا المتغير عند قيامنا بإدخال مجموعة حروف (او سلسلة) عددها (*n*) من لوحة المفاتيح ، ولا نحتاج بعد ادخالها الى الضغط على مفتاح (إيابRETURN) . **مثال** :

```
90 X$=INPUT$(8)
```