المرشد
في لغة البيسيك

إعداد

لواء أ ح أحمد زكي عبد العظيم

ماجستير في علوم الادارة
درجة الزمالة في علوم الحاسب
مدير مركز CSL للحاسب الآلي

المركز العربي الحديث

١٠٢ ش الإمام على ميدان الإسماعيلية - مصر الجديدة
تلفون ٨٣٤٠٤٧٤ ٦/١١ ٦٦ القاهرة

يطلب من

دار الشام للطباعة والنشر والتوزيع

مكتبة لبنان

[illegible]

مقدمة

ـ لغة البيسيك هي مجموعة قواعد وضعت لتمكن المبرمج من التعبير عن مسألة ما في شكل يمكن لكمبيوتر فهمه وحل هذه المسألة.

أي إنها لغة برمجة

ومعنى بيسيك (BASIC) هو تعليمات رمزية لجميع الأغراض المستهدفة للمبتدئين (Beginness All punpose Symbolic Instruction Code).

ـ وهذه اللغة بسيطة في بنائها ـ ولها مفردات لغوية قليلة ـ وهي سهلة التعلم حتى للمبتدئين في علوم الكمبيوتر ومع ذلك فهي لغة قوية تستطيع أن تلبي جميع المتطلبات تقريباً ـ ويمكن استخدامها في جميع أنواع الكمبيوتر سواء الشخصي (PC) أو الصغير (MICRO) أو المتوسط (MINI) أو الكبير (MINI - FRAME).

ـ تستخدم قواعد لغة البيسيك الرياضيات العادية وأحياناً الحديثة وهذا يجعل منها لغة مناسبة لحل جميع المسائل والمعادلات الرياضية كما أنها تتداول الحروف بطريقة سهلة وهذا يعني أنها تناسب أيضاً تشغيل البيانات الموضوعية

ـ هناك تطبيقات عديدة للغة البيسيك مع اختلافات كثيرة بين هذه التطبيقات. ولكننا سنشرح في هذا الكتاب لغة البيسيك المصغر (Minimal - Basic) والمبني على أساس المستوى ECMA - 55.

ومعنى ذلك أنه يجب أن تتوقع أن ما تجده في هذا المرجع ليس كل شيء في لغة البيسيك ـ ولكننا نؤكد أن هناك القليل الذي لم يذكر في هذا المرجع

ـ ولغة البيسيك تعرف بلغة المبتدئين وهي مكونة من جمل (Statements) وهي جمل أمره مثل اقرأ (Read)، دع (Let)، اطبع (Print)، اذهب إلى (Got To)، إنهي (END) وغالباً ما تستخدم لغة البيسيك من خلال نظام المشاركة الزمنية (Time Sharing) والتي تحقق إمكانية استخدام كمبيوتر معين بواسطة عدد من شاشات المستخدمين في نفس الوقت وفي أغراض مختلفة ـ ويستطيع تلبية طلباتهم بسرعة كافية بحيث لا يشعر أحد منهم بأن غيره يشاركه في العمل على نفس الكمبيوتر (نظام حجز تذاكر السفر في شركات الطيران مثلاً).

ـ وتعرف لغة البيسيك بأنها لغة الاتصال الساخن مع الكمبيوتر حيث إنها بعكس جميع اللغات لا تحتاج برامجه إلى (compilation) وما تتطلبه من وقت وجهد ولكن توجد به إمكانية الترجمة الفورية (Interpreratin) بمعنى أنه يمكنك أن تكتب أمراً واحداً على لوحة الأزرار فيأتيك الرد فوراً حتى بدون برمجه ـ وعلى ذلك نجد أنه أثناء كتابتك لأي برنامج فإن كل سطر تكتبه تتم عليه مراجعة فورية بحيث أنه عند وصولك لنهاية البرنامج ستكون متأكداً من عدم وجود أي أخطاء لغوية به.

ـ ونحن نفترض مقدماً أن القارئ ليست لديه فكرة عن علوم الكبيوتر ولذا فإننا سنعطي أمثلة حية يتعرف منها القارئ على فكرة بناء اللغة قبل مناقشة تفاصيل البناء ـ كما أضفنا رسوم توضيحية لتسلسل الخطوات (Flowchart) لتظهر بوضوح تسلسل الخطوات التي تقود إلى الحل الصحيح وليرسخ في ذهن القارئ أهمية تحضير وتحليل المشلكة للحل الكمبيوتري.

ـ سنحدد بعد ذلك الأجزاء المكونة لبرنامج البيسيك ونعطي معاني لتوصيف تعليمات لغة البيسيك ـ ونتوقع من القارئ أن لا يفهم هذا الجزء تماماً من القراءة الأولى ـ ولكنه بالتدريج سيعتاد على تحديد أسلوب بناء اللغة عملياً أثناء تقدمه في قراءة باقي أجزاء المرشد

ـ بعد ذلك سنقدم الكلمات الحاكمة في تعليمات البيسيك والتي تكون مفردات اللغة وذلك من خلال عدة برامج.

ـ روعي تقديم البرامج بالأسلوب الآتي (LET, PRINT and END) بمعنى إترك/ اجعل، اطبع وانتهى، وذلك حتى يمكن كتاب برنامج بسيط بواسطة القارئ المبتدئ ـ حتى يتمتع برؤية هذا البرنامج وهو يعطي نتائج فورية.

وسنعطي المثال التالي لنوضح أسلوب التفكير لحل مسألة كمبيوترية.

ـ سنفترض أنه في إحدى المدارس أردنا معرفة عدد التلاميذ الناجحين في أحد الامتحانات وكذا عدد الراسبين ـ علماً بأن المعلومات المتيسرة لدينا هي كشوف بأسماء التلاميذ والدرجات الحاصلين عليها في الامتحان.

ـ يجب أولاً أن نجزأ المسألة إلي أجزاء متسلسلة بشكل منطقي وصحيح حتى نصل إلى حل المسألة ـ أي إننا يجب أن نفكر في تسلسل الحل كأحداث متتالية

ـ للمعاونة في توضيح هذه الخطوة فإننا نستخدم الرسوم الإنسيابي (Flowchart)

البرنامج المطلوب

```
010 REM * * * * * * * * * * * * * * *
020 REM * EXAM ANALYSIS PROGRAM
030 REM * FIND NUMBES OF PASSING
040 REM * PUPILS AND FAILURES
050 REM * P - NUMBER OF PASSES
060 REM * F - NUMBER OF FAILURES
070 REM * S - PASS MARK
080 REM * M - PUPIL MARK
090 REM * * * * * * * * * * * * *
100 PRINT "EXAM ANALYSIS PROGRAM"
110 INITLALIZE PASS AND FAILURE COUNTS
120 LET P = Ø
130 LET F = Ø
140 REM READ IN PASS MARK
150 READ S
160 REM READ IN NEXT MARK
170 READ M
180 REM EXIT FROM LOOP IF NO MORE MARKS
190 IF M < Ø TWEN 270
200 REM TEST MARK AGINST PASS MARK
210 IF M < S THEN 240
220 P = P + 1
230 GO TO 170
240 F = F + 1
```

```
250 GO TO 170
260 REM END OF LOOP
270 PRINT P; "PASSES AND" ; F; "FAILURES"
280 DATA 45
290 DATA, 56, 57, 14, 80, 67, 30
300 DATA, 17, 48, 58, 83, 24
310 DATA - 1
320 END
```

OUTPUT

```
EXAM ANALYSIS PROGRAM
7 PASSES AND 4 FAILURES
```

عناصر اللغة

سيتم تحديد مكونات أجزاء اللغة في هذه المرحلة لإعطاء معنى أكبر لتوصيفات أوامر البيسيك المتعددة والتي تأتي في سياق البرنامج وكذا لتفادي التكرار الغير ضروري في التوصيفات.

صمم البيسيك ليتعامل مع البيانات الرقمية وغير الرقمية ـ فالأرقام تسمى بالثوابت الرقمية (Numeric constants) ـ ومجموعات الحروف التي تكون جمل يشار إليها بأنها ثوابت مسلسلة (string constants).

الثوابت الرقمية Numesic Constants:

يمكن أن تكون إما رقم صحيح (Integer) وهو رقم يكتب بلا أي علامات عشرية (Decimal Piont) ـ أو قد يتكون من رقم صحيح متبوع بعلامة عشرية ثم كسر ـ أو قد يكون كسر فقط ومعه العلامة العشرية ـ وتذكر

جيداً أنه غير مسموح بالعلامة العشرية المعروفة في اللغة العربية أن توضع خلال رقم ولكن تستخدم النقطة كعلامة عشرية ـ وإذا كان الرقم سلبياً توضع علامة ناقص على يساره (-).

وعند استخراج النتائج نجد أن لغة البيسيك تطبع الأرقام في أشكال مماثلة والرقم السالب تسبقه علامه ناقص (-3.7) ـ والرقم الموجب تسبقه مسافة خالية.

والبيسيك يطبع آلياً الأرقام فوق وتحت الحجم العادي عند اللزوم، وفي حالة الأرقام الكبيرة فإنه يستخدم الحرف E لتقليل الأرقام المكتوبة؛ بفرض أن E تمثل الأساس ١٠ وكمثال:

3500000000	فإنها تصبح	3.50000E + 9
.000009	فإنها تصبح	9.00000E - 6
.0000192	فإنها تصبح	1.92000E - 5

ولتسهيل الفكرة على القارئ فإنك في المثال الأول تحرك العلامة العشرية لجهة اليمين لمسافة ٩ وتستكملها بأصفار ـ وفي المثال الثاني (-٦) تحرك العلامة العشرية لجهة اليسار لمسافة ٦ مع مليء المسافات الخالية بأصفار.

وقد تكون الأرقام ضمن المدخلات باستخدام فكرة الحرف E.

إن القيم العليا والدنيا التي يمكن تداولها ـ ودقة الحروف المخزنة تعتمد على التطبيق السليم ـ فالبيسيك المصغر يحدد المدى من 38 - 1E وهو كسر ضئيل للغاية لو طبقت القاعدة السابقة ـ حتى 38 + 1E وهو رقم كبير للغاية ـ وليس أقل من ٦ أرقام عشرية ـ وهذا طبعاً مدى واسع لاستخدام الأرقام

الحرفيات الثابتة String Constants:

الحرفية في لغة البيسيك هي حروف متتالية موجودة داخل أقواس صغيرة مزدوجة (" ـ ") وأي حرف قابل للطباعة يمكن أن يظهر في سلسلة مع استثناء الأقواس المزدوجة حيث أنها آلياً تقفل/ تنهي **الحرفية** وبعض التطبيقات تسمح باستخدام قوس مفرد ضمن **الحرفية** لاحظ أن المسافة الخيالية هي رقم اعتباري

وكمثال **لحرفية** ثوابت "OCTOBER 2ND, 1987"

والمسافة الخالية بين الأقواس يطلق عليها **حرفية** فارغة "" (Null string) ولاحظ أن **الحرفية** يمكن أن تحتوي على حروف رقمية ولكن لا يمكن إجراء أية عمليات رياضية/ حسابية على البيانات المخزنة في **حرفية** ثوابت

ولغة البيسيك تسمح بتحديد **حرفية** بدون استخدام الأقواس المزدوجة وذلك خلال حالة بيانات (Data statement) وكذلك كإجابة ذات مفتاح لطلب مدخل (Input Request) لبيانات حرفية

المتغيرات الرقمية Numeric Vasiables:

المتغيرات الرقمية هي أسماء أو تعريفات تستخدم للإشارة إلى مناطق التخزين في الكمبيوتر ـ والقيمة يمكن أن تشير إلى متغير رقمي ثم تتغير كلما رغبنا في ذلك

فالمتغير الرقمي البسيط يتم تعريفه بحرف أبجدي مفرد أو بحرف أبجدي ورقم مثل Q7 S2 T P

ولاحظ أن في بعض التطبيقات فإن المتغيرات الرقمية يتم تصفيرها (أي عودتها إلى الصفر) قبل تنفيذ البرنامج بحيث أن المتغير الذي ليس له أي قيمة قبل التنفيذ فإنه يستخدم كأنه مساوي آلياً لصفر. وحيث أن هذه البداية ليس عمومية لجميع النظم ـ فإنه من المطلوب بشدة أن المتغيرات الرقمية يجب أن تعطي قيمة قبل استخدامها ولاحظ أيضاً أن بعض تطبيقات البيسيك تسمح باستخدام حروف مزدوجة.

الحرفيات المتغيرة String Variables:

تستخدم لتعريف مناطق التخزين التي تحتوي على مسلسلات حروف ولتعريف حرفية متغيرة بسيطة فإننا نستخدم حرفين: حرف عادي وعلامة الدولار A$ Q$ ـ وعدد الحروف يتوقف على نوع لغة البيسيك فبعض اللغات يسمح بثمانية عشر حرفاً ـ وتذكر دائماً أن الحروف لها قيمة رقمية داخل الكمبيوتر (كود رقمي) فالحرف A هو أقل الحروف قيمة بينما Z أكبرهم قيمة ومقارنة قيم الحروف الموجودة في الحرفيات يمكن معرفة قيم الحرفية.

تشكيل المتغيرات.

إن برامج الحاسب الآلي مطلوبة دائمة للتعامل مع تجميعات عناصر بيانات متقاربة لذا فإنه من المناسب أن تكون قادرة على استخدام اسم لتعريف تجميع بيانات بدلاً من أسماء منفصلة لعناصر البيانات إذا أردنا الإشارة إلى التجميع. فاسم التشكيل المتغير هـ تجميع لمتغيرات مع استخدام نفس الاسم وعندما نريد تعريف عنصر ما مشابه في الحجم والمحتويات بعناصر أخرى من نفس التجميع البياني فإننا نستخدم حرفاً ورقماً . والحرف مشترك بين جميع هذه العناصر أما الرقم فيختلف طبقاً لموضع العنصر في التجميع

مثال

A(0) A(1) A(2) A(3) ... A(n)

لاحظ أنه في بعض تطبيقات البيسيك لا يستخدم (0) بل يبدأ بواحد صحيح ويتم كتابة التشكيل المتغير كالآتي

A (3) D (S) BS (2) M$ (K - 2)

ويمكن استخدام الموصف A(n) للدلالة على A (1) ، A (2) إلخ

تذكر نفس الاسم لا يمكن أن يستخدم لتعريف متغير بسيط ومتتالية متغيرة في نفس البرنامج

ولكن في الجداول الثنائية البعد ـ فإن عنصر عادي مستخدم في البعدين يمكن تعريفه بموصف مزدوج R$ (20, 16) ، E (3, 4)

حيث الرقم الأول من اليسار إلى البعد الأفقي والثاني إلى العمود أي البعد الرأسي وواضح أن العنصر E له ١٢ عنصر منها ثلاثة أفقي وأربعة رأسي في جدول.

التعبيرات الرقمية:

+ ADDITION (الجمع) - SUBTRACTION (الطرح)

× MULTIPLICATION (الضرب) / DIVISION (القسمة)

وطبقاً للنظام المستخدم ** أو ^ أو ↑ exponentiation (الرفع للقوة) ومعنى ذلك أن نتعامل أولاً مع الرفع يليه الضرب ثم القسمة وذلك طبعاً من اليسار لليمين ـ ثم نتعامل أخيراً مع الجمع والطرح ويمكن طبعاً إستخدام

الأقواس بنفس الأساليب الرياضية المعروفة فإذا أردنا أن نعبر عن المعادلة الرياضية A + B * C / D ** E فيمكن أن نفصلها كالآتي

العملية الأولى ونتيجتها هى D ** E : R1

العملية الثانية ونتيجتها هي B * C : R2

العملية الثالثة ونتيجتها هي R2 / R1 : R3

العملية الرابعة ونتيجتها هي A + R3 : R4

ولاحظ أنه لا يمكنك أن تضع تعبيرين رقميين متتاليين مثل A / -B وصحتها A / (-B) ـ كما أنك لا تستطيع افتراض فهم الكمبيوتر لعملية مثل A (B + 1) كما تعودنا في الرياضة ولكنها يجب أن تكتب A * (B + 1) ويتم تنفيذ الحاسب الإلكتروني للأوامر الرياضية المبرمجة بلغة البيسيك بالأسبقية التالية:

(١) الأقواس ( ).　　(٢) الأس * *

(٣) الضرب والقسمة *، /.　　(٤) الجمع والطرح +، -

اصطلاحات المقارنة:

= مساوٍ لـ　　< > ليس مساوٍ لـ

< أقل من　　= > يساوي أو أقل من

> أكبر من　　= < يساوي أو أكبر من

وتستخدم مع IF ... THEN. فقط في البيسيك المصغر ولكن في بعض تطبيقات البيسيك الأخرى قد تستخدم للربط في AND, OR, NOT

الثوابت Constants

عددية Numeric

صحيحه INTEGER

حقيقية REAL

حرفية String

تكوين البرنامج:

يتكون برنامج البيسيك من تعليمات متتالية تكتب كمتتاليات من حالات البيسيك وكل حالة أو تعليمه تكتب على سطر منفرد وكل سطر له رقم - ورقم السطر هو عنوانه ويستخدم لترتيب وتجميع الحالات للتنفيذ. وقد يتم سحب الحالة من المخزون للتعديل أو التحريك باستخدام رقم السطر وتنفذ حالات البرنامج بترتيب السطور حتى يتغير الترتيب المثالي بأمر والذي من خلال البرنامج والتنفيذ يستمر حتى END أو STOP وحالة END هي دائماً آخر سطر في البرنامج ولها أعلى رقم سطر

- وتدخل الحالة كما قلنا في سطر له رقم ولا يصح أن تزيد عن ٧٢ حرف ورقم السطر يتراوح بين ١، ٩٩٩٩.

كقاعدة عامة لا تترك فراغات في بداية الحالة / السطر أو فراغات تتحلل الأرقام أو الكلمات المفاتيح.

- وعند ترقيم الأسطر على المبرمج أن يترك احتياطيات في الترقيم حتى يمكن إضافة تعديلات بدون تغيير الترقيم كله لأنه من الصعب أن تكتب برنامجاً من أول وهله بطريقة صحيحه ولابد من التعديل والإضافة لذا ننصح بأن يكون الترقيم بالعشرات (١٠ ، ٢٠ ، ٣٠ إلخ)

الكلمات المحجوزة RESERVED WORDS

هناك كلمات معينة لا يجوز استخدامها كأسماء لمتغيرات أو بيانات فكل مفردات لغة البيسيك تعتبر كلمات محجوزة لا يجوز استخدامها كمتغييرات وإذا اضطررت لاستخدامها فيمكن إضافة حرف لها لتغيير شكلها.

المسافات الخالية:

وتتفق جميع لغات البيسيك في أنه لا يجوز أن يتضمن اسم البيان مسافات خالية.

أوامر البيسيك:

سنقدم الآن شرحاً لكل أمر/ تعليمه بيسيك ـ مع توصيف القواعد اللغوية التي تحكم استخدام كل أمر ـ وقد أضفنا أمثلة لبرامج لتوضيح الأوامر واستخدام الحالات.

ويتكون كل مثال لبرنامج من تسجيل لكل حالات البيسيك والتي تحمل البرنامج في سياق منطقي للأوامر ـ وأرفقنا مع كل مثال برنامج شكل المستخرج بشكل يطابق ما سيخرجه الكمبيوتر تماماً وتدخل البيانات في حروف مصغرة (Small letters).

ويجب أن تكون معتاداً على استخدام لوحة المفاتيح في الكمبيوتر لأنها هي أيضاً لها أوامرها الخاصة كمثال (Return) المكتوبة على مفتاح تستخدم في نهاية كل سطر كأمر لبداية السطر التالي ولكنها أيضاً تستخدم لتحويل السطر المدخل إلى الكمبيوتر.

١- PRINT / END

Program: For Intsoducing

```
10 PRINT "MY 1ST LINE"
20 PRINT 6 + 7
30 PRINT "6 + 7"
40 PRINT "4 - 7/2
50 END
```

the output will be:

```
MY 1ST LINE
 13
6 + 7
-1.5
```

لاحظ أن PRINT قد استخدمت أيضاً لإعطاء سلطة إجراء حسابات رياضية.

٢- LET

Program

```
10 LET A = 5
20 LET B = 4.2
30 LET C = A * B
40 LET D$ = "ADEL ISAC"
50 PRINT D$
60 PRINT C
70 PRINT "END OF OUTPUT"
80 END
```

Output

```
ADEL ISAC
 21
END OF OUTPUT
```

ملاحظات عند استخدام LET

LET تعطي قيم للمتغيرات أثناء تنفيذ البرنامج

LET A = 5 وتعطي القيمة ٥ للمتغير الرقمي A

وتعطي الأحرف المسلسلة بين الأقواس مقابل هو مسلسل متغير

LET B$ = "ALI"

LET D = P/Q + 0.5

LET F$(2) = "MONDAY"

LET S = T5

لإضافة واحد للقيمة القديمة كقيمة جديدة

٣- أمر الطباعة PRINT statement:

تم تصميم حاله الطباعة لنقل المعلومات من الكمبيوتر إلى أجهزة خارجية كشاشة ومطبعة، أو لقدرة الإظهار علي الشاشة ـ والصيغة المبسطة هي:

ـ حيث X هو عنصر للطباعة في شكل رقمي أو عبارة مسلسلة n PRINT x

ـ تطبع القيم المخزنه في المنطقة المرافقة للمتغير B PRINT B

ـ تطبع قيمة التعبير المستخدم للقيمة المتواجدة لأغراض

قييم C PRINT C * B/ 3 + 32

ـ لطبع سلسلة حروف بين أقواس مزدوجه PRINT "CAT MAT"

ـ لطبع السلسلة المخزنة بين المنطقة المرافقة للمتغير المذكور PRINT D$

ـ لطبع الرقم الثابت ٤٢٣ PRINT 423

ويتسبب طلب طبع متغير في سحب محتويات المنطقة الرافقة لهذا المتغير وإخراجها وعندما يخصص ثابت فإن الحروف تتجمع للخروج في نفس الترتيب المستقرة فيه ـ وعندما يكون عنصر طباعة في شكل تعبير رقمي فإن هذا التعبير يتم تقيمه للحصول على قيمة واحدة تجمع للإخراج

وكل حالة طباعة تخلق خط جديد من الإخراج إلا إذا كانت حالة الطباعة السابقة منتهية بفاصلة أو فاصلة منقوطة (, ;)

كما يمكن أن تأخذ حالة الطباعة الشكل التالي

n PRINT S1, S2, SI حيث S عنصر طباعة، I عدد عناصر الطباعة

وسطر الطباعة مقسم إلى عدد من مناطق الطباعة (Print zones) كما هو محدد بالتطبيق المعمول به ـ وفي معظم الأحوال فهناك خمس مناطق بعرض متساوي ـ مع احتمال اختلاف المنطقة الخامسة

ويتم طباعة العناصر المنفصلة بفاصلة (,) في مناطق مختلفة أما العناصر المنفصلة بفاصلة منقوطة (;) فيتم طباعتها متجاورة

لأن الفاصلة تسبب إيقاف الطباعة في نفس المنطقة وتنقل الطباعة إلى المنطقة التالية ـ أما الفاصلة المنقوطة فتأمر باستمرار الطباعة من نفس المنطقة أي إنها تقول: «انسى المناطق واستمر في الطباعة من المكان الذي توقفت فيه».

وسلسلة الخرج من تعبير رقمي تتكون من مسافة خالية إذا كان الرقم موجب أو بعلامة ناقص (-) إذا كان الرقم سلبي ثم يتبع ذلك تقديم الرقم سواء صحيح أو عشري أو أي شكل رياضي ثم يتلو ذلك مسافة خالية ـ وهكذا حتى لو استخدمت العلامة (;) كفاصل فإن النتائج الرقمية في

الحقيقة ستكون منفصلة بفاصل على الأقل والحرفية المستخرجة من تعبير متسلسل في سلسلة حروف يمكن أن تكون أطول من منطقة طباعة واحدة - فإذا كانت أطول فإنها تطبع ما يزيد في المنطقة التالية - واستخدام الفاصلة (,) كفاصل هو أمر لعنصر الطباعة التالي ليبدأ في المنطقة التالية.

وعند ما نضع في الأمر عناصر أكثر مما يمكن إخراجه على سطر واحد فإن الطباعة تستمر بعده طبيعية إلى السطر التالي

ويمكن استخدام الفاصلة (,) في آخر حالة الطباعة لتأكيد أن أمر الطباعة التالي سيخرج مستخرجاته في منطقة الطباعة التالية (وليس بالضرورة على السطر التالي).

وبالمثل فإن معنى أن نضع فاصلة منقوطة في آخر أمر الطباعة بسبب أن سلسلة الطباعة التالية ستستمر على نفس السطر إذا تركنا مسافة والحالة الوحيدة التي يترك فيها سطر خالي من الطباعة ويطبع على السطر التالي له أنه تصدر أمر طباعة هيكلي بدون ذكر عناصر طباعة وتستخدم ذلك في البرامج لنترك سطراً خالياً بدون طباعة

والبيسيك لا يسمح ببدء الطباعة من عمود معين في سطر الطباعة ولكن يمكن التغلب على ذلك باستخدام TAP Function وسيأتي ذكره فيما بعد

والنقاط المختلفة المتعلقة باستخدام (; ,) لفصل المخرجات سيوضحها المثال التالي

Program:

```
010 LET A = 6
020 LET B = 4
030 LET C = 20
040 LET D = A * B/C
050 PRINT A, B, C, D, B - C
060 PRINT A; B; C; D
070 PRINT B * C, "PRICE OF TOTAL SALES",
080 PRINT (B * C) * * A
090 END
```

OUT PUT

```
6          4          20          1.2          -16
6   4   20   1.2
80   Price of toral Sales   2.62144E + 11
```

٤- الأمر ـ انتهى : END Statement

تعنى انتهاء برنامج البيسيك ـ وهي تنهي تتالي الحالات التي تكون البرنامج وتوقف التنفيذ عندما تنفذ وهي تأخذ الشكل حيث n هي أعلى رقم سطر استخدم في البرنامج n END.

٥- الأمر الغير تنفيذي: ملحوظة (Remark) REM Statement

هي أمر غير تنفيذي صمم ليسمح ببيان البرنامج ولكن الكمبيوتر يتجاهله في وقت التنفيذ ـ بسيطرة البرنامج فإنه يتم الانتقال إلى الأمر التنفيذي التالي له ـ وهي تمكن المبرمج أن يضع تعليقاته وملاحظاته التوضيحية

خلال إطار البرنامج ليجعله قابلاً للقراءة والفهم ـ وهي تأخذ الشكل الآتي وطول التعليق يحدد بطول سطر واحد n REM comment

أما التعليقات الأطول فإنه يستحسن تكرار الأمر ـ وكمثال لاستخراج (REM) يمكنك الرجوع إلى برنامج نتائج الامتحان ـ وكمثال آخر مع استخدامDATA , READ

Program:

```
10  REM READ AN ITEM OF DATA AND PRINT IT
20  REM OUT TO KNOW HOW IT IS READ AND
30  REM WRITTEN IN A CORRECT WAY
40  REM N
50  DATA 65
60  PRINT "N =" 3N
70  END
```

OUT PUT

N = 65

٦- الأمر READ اقرأ — READ Statement

تستخدم لتعيين قيم المتغيرات من بيانات متتالية مخلقة بواسطة واحدة أو أكثر من حالة DATA ـ وتأخذ الشكل الآتي

n READ V1, V2, . . . Vi

حيث كل V هي متغير رقمي أو مسلسل ـ بسيط أو موصف وعند تنفيذ Read فإن المتغيرات المسجلة تحدد لها قيم تبعاً لحالة DATA

والعنصر الأول من البيانات يخصص له المتغير الأول والعنصر البياني الثاني يعين له المتغير التالي وهكذا

وتتابع البيانات يتم الحفاظ عليه بواسطة مؤشر يشير إلى العنصر الأول في بداية تنفيذ البرنامج وبعد قراءة العنصر وتحديد قيمته فإن المؤشر يتقدم إلى العنصر التالي

ويستخدم أمر RESTORE ليعيد المؤشر إلى بداية كتل البيانات ولذا فإنه في الإمكان قراءة البيانات المتتالية أكثر من مرة

لاحظ أن العنصر الثابت المحدد له قيمة متغيرة يجب أن يكون من نفس نوعها. فالحرفية المتغيرة تحدد لها فقط سلسلة من الحروف - والرقم المتغير تحدد له فقط ثابت رقمي - وهذه فائدة أساسية يجب اتباعها دائماً لأنه من السهل ارتكاب مثل هذا الخطأ في Read - وأي محاولة في تنفيذ البرنامج لتعيين بيانات من نوعيه مختلفة ستظهر كخطأ كبير (Fatal Error)

Program:

```
10 READ A, B, C, d
20 LET S = A + B + C + D
30 PRINT A, B, C, D, S
40 DATA 6, 7, 3
50 DATA 142,91
60 END
```

٧ الأمر الغير تنفيذي بيانات DATA Statemen

تستخدم لإضافة بيانات إلى ما نعتقد أنه كتلة بيانات داخلية في البرنامج ـ ثم تصبح البيانات بعدها متيسره للتعيين من الكتلة خلال تنفيذ البرنامج إلى متغيرات تحدد بحالات Read وهي تأخذ الشكل

n DATA d1, d2, ..., di

حيث كل d هي عنصر بيانات منفصل في شكل رقمي أو ثابت مسلسل وعناصر البيانات المستقلة يتم الفصل بينها بواسطة شوله (,) وتوضع عناصر البيانات في كتلة أثناء وقت الترجمة Compilation وخلال تنفيذ البرنامج يتم تجاهل أمر البيانات مثل أي أوامر أخرى غير تنفيذية (مثل REM) وتمر السيطرة فوراً إلى الأمر التنفيذي التالي وحيث أن هذا الأمر لا يلعب دوراً خلال التنفيذ فإنه يمكن وضعه بأي مكان في البرنامج ـ وعموماً فإنه يجب ملاحظة أن عناصر البيانات يتم ترتيبهم بالتتالي في كتلة البيانات بالشكل الذي يتم ظهورهم فيه في البرنامج وإننا لنوصي أن يوضع كل بيان (DATA) في البرنامج بالتتالي سوياً وحتى نهاية البرنامج ـ ويظهروا للمبرمج ككتلة واحدة

Program:

```
010 REM THIS PROGRAM READS SOME DATA
020 REM DOES A LITTLE ARITHMATIC
030 REM AND PRINTS OUT THE RESULT
040 READ X, AS, MS
050 PRINT MS; "TOTAL IS "; X; " INCHES
```

```
060 LET C = C + X
070 PRINT "CUMULATIVE";A$/"TOTAL IS"; C;"INCHES
080 DATA 1.96 "RAINFALL","MONTHLY"
090 DATA 10.07
100 END
```

Out Put for Read and DATA Exampls

```
6        7        3        142        158
MONTHLY TOTAL IS  1.96 INCHES
CUMULATIVE RAINFALL TOTAL IS 12.03 INCHES
```

RESTORE Statement — ٨- الأمر: أعد تخزين

تستخدم لإعادة المؤشر إلى بداية كتلة البيانات ـ وهذا يمكن للقراءة التالية في أن تعيد تحديد البيانات من بداية المتتالية ـ وتكون بالشكل الآتي RESTORE n

Program:

```
010 READ A
020 READ B,X
030 RESTORE
040 READ M1, M2
---------- --------
-------------------
-------------------
200 DATA 21
210 DATA 0.5, 6.3, 4
220 DATA 12.9,6.6
```

٩- الأمر ادخل INTPUT Statement

يمكن به تعيين قيم البيانات للمتغيرات بواسطة مستخدم البرنامج خلال التنفيذ الفعلي له ـ وهو أمر فريد في لغة البيسيك يتطلب تفاعل داخلي في مرحلة التنفيذ ـ ويكون بالشكل الآتي

n INPUT V1, V2, ... Vi

فإذا تخصص أكثر من متغير واحد بواسطة INPUT فإننا نفصل المتغيرات وعند تنفيذ هذه الحالة فإن البرنامج في حالته الفعالة يقف لبرهة وينتظر تعيين القيم للمتغيرات المخصصة ـ ولغة البيسيك تزود المستخدم بصيغة إدخال مطابقة لشكل السؤال على الشاشة. لاحظ أن البرنامج لا يمكنه الاستمرار في العمل حتى تزوده بالبيان المطلوب وعند إدخال البيانات كاجابه على هذا الطلب فإننا نفصل كل عنصر بشوله ـ وبعض تصريفات اللغة تسمح بعمل فاصل مساحة فارغة بين الأرقام أما البيانات **الحرفية** فيمكن إدخالها بدون أقواس مزدوجة وبشرط أن لا يسبقها فاصل أو (+) أو (-) أو(.) أو (و)

وأي بيانات مدخلة يجب أن يكون نوعها من نفس نوع المتغير المعينة له، ومن السهل على برنامج كبير أو برنامج لم تستخدمه لفترة أن ينسى إنك من المفروض أن تدخل بيان ـ ولذا فإننا نوصي أن يظهر سؤال بواسطة البرنامج قبل صيغة الإدخال التي يزودنا بها النظام للانتباه إلى صيغة الإدخال ـ حيث أنه ليس هناك شيء يسبب الارتباك أكثر من ظهور علامة استفهام عندما لا تتذكر ما هو السؤال

مثال للأمر: PRINT عندما نستخدمه مع الأمر INPUT

Program:

```
010 REM Program To Print Out The Date
020 REM And Maximum Temperature Recorded
030 PRINT "Day OF Week"
040 INPUT D$
050 PRINT "Date"
060 INPUT M$
070 PRINT "Max Temp"
080 INPUT T
090 PRINT
110 PRINT
110 PRINT "Max Temp On";D$;M$;
120 PRINT "Was";T;"Degees Centigrade"
130 END
```

OUT PUT

```
DAY OF WEEK          للرد على السؤال 040
? Tuesday
DATE                 للرد على السؤال 060
? Oct 4th
```

MAX TEMP? 13 للرد على السؤال 080

Max Temp On Tuesday Oct 7th Was 13 Degree Centigrade

١٠ – اذهب إلى ... ثم GO To Statement

هذا الأمر يمارس عملية انتقال غير مشروطة إلى رقم سطر معين وهكذا يحدث تبديل في تتالي التعليمات تحدده أرقام الجمل الأمره وتكون بالشكل التالي n Go To Line no

ويمكن الانتقال به إلى رقم سطر سواء أعلى أو أدنى من السطر الموجود به هذا الأمر .. بمعنى للأمام أو للخلف.

ومطلوب الحرص الشديد في إجراء الانتقالات الغير مشروطة ـ فإنه من السهل أن تدخل في دوامة (LOOP) بلا نهاية وفي بعض حالات الـ Loop فإنه يمكنك الهروب منها وتنفيذ تعليمات البرنامج ولكن حالة الانتقال الغير مشروط تلغي تنفيذ البرنامج عند النقطة التي بدأ منها الانتقال إذا لم يكن محكوماً تماماً

Program (Go To And IF .... THEN)

```
010 REM SUM THE FIRST 100 INTEGERS
020 REM N : NEXT NUMBER S: COMULATIVE SUM
030 S = 0
040 N = 0
050 N = N + 1
060 S = S + N
070 IF N = 100 THEN 90
080 GO TO 50
```

```
090 PRINT "SUM OF FIRST 100 INTEGERS"
100 PRINT "IS";S
110 END
```

في المثال السابق فإن السيطرة تتحرك من الجملة (٧٠) إلى الجملة (٩٠) فقط عندما تساوي (N) القيمة ١٠٠ ـ وخلاف ذلك فإن السيطرة تسير في الاتجاه العادي إلى الجملة (٨٠).

وهذا كان IF ... THEN تستخدم للهروب من الدوامة (LOOP) الذي تسببها (GO TO).

لاحظ أنه يمكن اختصار الجملتين (٧٠، ٨٠) في جملة واحد كالآتي:

```
070 IF N < 100 THEN 50
```

١١- الأمر IF ... THEN

هذا الأمر يمارس عمليه نقل السيطرة تحت شروط معينة في الجملة الآمرة ويكون بالشكل الآتي:

n IF (تعبير ارتباطي) THEN Line N

في حالة صحة التعبير الارتباطي فإن السيطرة تنتقل إلى رقم السطر بعد (إذا) أما التعبير الارتباطي فيتكون من تعبيرين رقميين أو تعبيرين حرفيين مرتبطين بعامل صلة/ ارتباط ـ ويتم التنفيذ إذا تمت المقارنة بينهما سابقاً لعامل الارتباط ـ فإذا كانت صحيحة تنتقل السيطرة إلى السطر المحدد وإذا لم تكن المقارنة صحيحة فإن تنفيذ البرنامج يستمر مباشرة من الجملة الآمرة التالية للأمر.

ويكون عامل الارتباط في إحدى الصور التالية

= Equal to	مساو لـ	< > Not equal to	
< Less than	أقل من	< = Less than or equal to	
> Greater than	أكبر من	> = Greates than or equal to	

وعند تقدير الصلة الرقمية في التعبير ـ فإن التعبيرين الرقميين قد يلزمهم هم أنفسهم تقدير قبل إجراء المقارنة مثل:

IF (x + 1) * * 2 > = Y * 3 THEN 50

فإذا كانت X تساوي ١ ، Y تساوي 4 فليس معقول أن ٤ أكبر من ١٢ فيكون خطأ أما إذا كانت X تساوي ٤، Y تساوي ١ فيكون التعبير صحيح حيث أن ٢٥ أكبر من ٣

وإذا كنت ملماً ببسط قواعد الجبر فيمكنك تطبيق المعادلة الآتية على المثل السابق إذا كانت (س + ١)٢ أكبر من أو تساوي ص × ٣ فانتقل إلى السطر (٥٠) فقط ضع (س) بدلاً من (X) ، ص بدلاً من (Y)

أما في حالة تقييم تعبير حرفي (غير رقمي) فإن التعبيرين الحرفيين يتم مقارنتهم حرفاً بحرف من الأول للآخر

وتذكر دائماً أن كل حرف يمثل بلغة الكمبيوتر بكود رقمي موحد فمثلاً A تعطي أقل قيمة، Z تعطي أعلى قيمة

فلكي يتساوى التعبيرين فإنهما يجب أن يكونا بنفس الطول ويكونا محتويين على نفس الحروف وبنفس الترتيب ـ أما إذا اختلف حرف من كل منها فيتم تقييمهما وفقاً لقيمة هذا الحرف مثلاً

"AHMID" is greater than "AHMED"

"JACKSON" is greater than "JACK"

"HUGH" is greater than "HUBERT"

وعند وجود مسافة خالية في أحد التعبيرين فإنها تعامل كحرف ولها قيمة أكبر من أي حرف أبجدي مثال:

"JACK " is greater than "JACK"

وبعض لغات البيسيك تسمح باستخدام المعاملات المنطقية التي تمكن أكثر من تعبير ارتباطي بأن يتوحدوا في تعبير واحد ـ وهذه المعاملات المنطقية هي: (AND, OR and NOT).

١٢- الأمر إلى ... التالي FOR .... NEXT Statements

(NEXT, For) هما أمران يستخدمان لتسكين والسيطرة على (Loops) فنجد أن (FOR) تسبب الـ (Loop) وتحدد عدد مرات تنفيذه. كما نجد أن (NEXT) تحدد نهاية الدوامة وترجع السيطرة إلى الجملة التي تلي (FOR) أي أن حالات الدوامة ستجدها بين (NEXT, For) . وتكون (FOR) بالشكل التالي: n FOR nv = ne 1 TO ne 2 STEP ne 3

حيث (nv) هو متغير رقمي بسيط يسمى متغير السيطرة على الـ Loop أو عداد الدوامة (Loop counter).

أما (ne3, ne2, ne1) فهي تعبيرات رقمي تمثل تماماً القيمة المبدئية لمتغير السيطرة سواء الحد الأقصى أو الأدنى ـ وهي القيمة التي سيزيدها متغير السيطرة (nv) في كل دورة خلال الدوامة.

لاحظ أن الحاجة إلى تعيين خطوة الزيادة اختيارية ـ فعند غيابها فإن الزيادة تقرأ على أنها (STEP1) الخطوة الأولى.

وفي التنفيذ فإن التعبيرات عند الضرورة تقيم لتجنب الحد المبدئي وقيم الخطوات وتحدد القيمة المبدئية لمتغير السيطرة وتقارن مع الحد المقرر سواء

كانت ضمن المدى المخصص أم لا ـ مثلاً تكون أقل أو مساوية للحد عندما تكون الخطوة الإيجابية ـ أو تكون أكبر من أو مساوية للحد عندما تكون الخطوة سلبية

فإذا كان متغير السيطرة داخل المدى ـ فإنه يتم تنفيذ جميع حالات الدوامة بالتوالي حتى يصل إلى الأمر التالي (NEXT) ويكون شكلها كالآتي: n NEXT nv

وقد سبق شرح معنى nv (متغير السيطرة على الدوامة/ العداد) وعند تنفيذ (NEXT) فإنه يزيد بقيمة الخطوة (STEP) ويقارن بالحد ـ فإن كان ضمن المدى فإن السيطرة تمر إلى أول حالة تنفيذية بعد (FOR) ـ وتستمر عملية الدوامة طالما ظلت قيمة ضمن المدى ـ فإذا اخرجته الزيادة من المدى فإن السيطرة تنتقل إلى الأمر التنفيذي التالي للأمر (NEXT) لاحظ أن الخطوة السلبية مسموح بها في حالة ما إذا زادت القيمة المبدئية عن قيمة الحد.

وأيضاً لاحظ أن قيم القيمة المبدئية، الحد، الخطوة يمكن أن تكون أرقام حقيقية كما يمكن أن تكون حروف ـ ولذا فإن الآتي بعد شكل صحيح

FOR J = 0.1 TO 10 STEP 0.1

وهذه الحالة تنفذ الـ Loop مائة مرة قبل أن تكتفي

FOR I = 6.5 TO- 3.5 STEP - 0.5

وتنفذ الأمر السابق للدوامة (Loop) ١٤ مرة قبل أن تكتفي ويمكن وضع (FOR Loop) بداخل (FOR Loop) أخرى ولكن بشرط عدم تداخلهم وتسمى هذه التوافقية شبكية الدوامة (Nesting Loop) والعمق الذي تتشابه فيه هذه الـ Loops يعتمد على حسن تطبيق البيسيك. وكمثال:

```
S = 0
FOR J = 1 TO N
For K = 1 TO N
S = S + (J, K)
NEXT K
NEXT J
```

وباستخدام عدادات اللوب كموصفات لمكونات كل منطقة (n) بواسطة عدد من المصفوفات (n) ـ فإنها تسحب وتضاف سوياً لتكون كم مركب من كل القيم المخزنة في الصف.

بمعنى توصيف كامل للمتغيرات المصفوفة.

فعند استخدام (FOR ..... NEXT) يجب حماية البرنامج من القفز داخل القامة (Loop) من الخارج (مثلاً عن طريق GO TO) حيث أن حالة (FOR) لن تنفذ إذا قفزنا فوقها بحيث أن متغير السيطرة لها لن يكون له أي قيمة وأيضاً لاحظ تجنب تغيير قيمة متغير السيطرة داخل الدورة (Loop) . وأخيراً لا تستخدم نفس متغير السيطرة لشبكية (Loop) إلا في (Loop) الذي يقع فيه.

Program:

```
010 REM POWERS OF THREE
020 FOR N = 1 TO 12
030 PRINT "3 TO THE POWER OF"; N;" = "; 3 * * N
040 NEXT N
050 END
```

Output

3 TO THE POWER OF 1 = 3

3 TO THE POWER OF 2 = 9

3 TO THE POWER OF 3 = 27

3 TO THE POWER OF 4 = 81

3 TO THE POWER OF 5 = 243

3 TO THE POWER OF 6 = 729

3 TO THE POWER OF 7 = 2187

3 TO THE POWER OF 8 = 6561

3 TO THE POWER OF 9 = 19683

3 TO THE POWER OF 10 = 59049

3 TO THE POWER OF 11 = 177147

3 TO THE POWER OF 12 = 531441

١٢- الأمر «قف» STOP Statement

يستخدم لأنهاء البرنامج خلال التنفيذ ـ وهو مطلوب فقط عندما تكون هنا حاجة لإيقاف التنفيذ في نقطة ما قبل الوصول للنهاية END ويأخذ الشكل STOP n

وعند التنفيذ فإن البرنامج ينتهي عند هذه النقطة ـ وقد تظهر في أكثر من مكان في البرنامج ولكن لا يحتاج اظهورها إطلاقاً إذا كان البرنامج سيصل بصورة طبيعية إلى النهاية دون الحاجة إلى إيقاف إضافي

Program:

```
010 PRINT "KEY IN YOUR AGE"
020 INPUT A
030 IF A > 30 THEN 60
040 PRINT "TIME IS STILL YOURS"
050 STOP
060 IF A > 55 THEN 90
070 PRINT "YOU MISS THE TRAIN"
080 STOP
090 PRINT "YOU OUGHT TO RETIRE"
100 END
```

Output:

```
KEY IN YOUR AGE
? 45
YOU MISS THE TRAIN
```

Program:

```
010 REM CONVERT LBS TO KILOS
020 PRINT "LBS (ENTER A NEGATIVE VALVE TO
    STOP EXCUTION):";
030 INPUT P
040 IF P < 0 THEN 100
050 REM 1 KILO = 2.2 LBS
060 LET K = P/ 2.2
```

070 PRINT K;"KILOGRAMS"

080 PRINT

090 GO TO 20

100 END

OUTPUT

LBS (Enter a negative value to stop Exevtion) : ? 11

5 KILOGRAMS

LBS (Enter a negative value to stop Exevtion) : ? 152

96.0909 KILOGRAMS

LBS (Enter a nagtive value to stop Exevtion) : ? -1

١٤- الأمر: عندما...إذهب إلى ON....GO TO Statement

تزودنا هذه الحالة بإمكانية النقل الشرطي والتي تسمح للسيطرة أن تتفرع إلى سطر من عدة سطور ـ وتأخذ الشكل الآتي

n ON ne GO TO Ln1, Ln2, ..., Lni

حيث ne تعبير رقمي، Ln رقم سطر تقفز إلى السيطرة طبقاً لقيمة التعبير الرقمي.

وعند التنفيذ فإنه يجري تقييم للتعبير الرقمي ـ فإذا كانت النتيجة عشرة فإن القيمة تجبر لتحصل على رقم صحيح ـ وعندما تكون بالقيمة واحد فإن السيطرة تنتقل إلي السطر الأول، وكذا عندما تكون القيمة ثلاثة فإن السيطرة تنتقل إلى السطر الثالث وهكذا ـ وكمثال

ON X GO TO 300, 300, 180, 400

فإن السيطرة تنتقل إلى السطر ٢٠٠ عندما تساوي X واحد صحيح

وكذا فإنها تنتقل إلى السطر ٤٠٠ عندما تساوي X أربعة.

١٥- الأمر: إنسخ LIST Statement

يستخدم لعرض البرنامج على الشاشة بالكامل - وفي حالة الرغبة في عرض جزء من البرنامج ولنفترض السطور من ٤٠٠ إلى ٧٠٠ مثلاً يستخدم كالآتي: LIST 400 - 700

ولعرض السطور من ٣٠٠ للنهاية LIST 300 -

والسطور من البداية إلى ٦٠٠ LIST -600

وللطبع على ماكينة الطباعة يضاف L ليصبح L LIST

١٦ - الأمر: إحفظ SAVE Statement

هذا الأمر يؤدي إلى عمل نسخة من البرنامج الموجود بالذاكرة وحفظها على الإسطوانة وتختلف مرفقات الأمر تبعاً لنوع وسيلة التخزين كالاتي

لحفظ البرنامج (١) على قرص مرن SAVE DSK1 PRG 1

لحفظ البرنامج (١) على كاسيت SAVE "P1"

لحفظ البرنامج (١) على كارتريدج SAVE "m":1:"P1"

١٧- الأمر: نفذ/ إجري RUN Statement

وتستخدم لتسيير برنامج بعد إدخاله وذلك بترجمته فتكتب كالآتي بعد نهاية البرنامج بدون رقم سطر RUN

وقد سبق أن أوضحنا أسلوب عملية الترجمة في المقدمة

المناهج الفرعية SUBroutines

من الشائع عند كتابة البرامج أنك غالبا ما تطلب من البرنامج ممارسة حالات متتالية أكثر من مرة في مراحل مختلفة من تنفيذ البرنامج ولتجنب تكرار الأكواد في كل نقطة من البرنامج ـ فإن الحالات يمكن ترتيبها كمنهج فرعي لبرنامج فرعي ـ ثم يمكن استخدامها عندما يتطلب الأمر ذلك ـ فإن استخدام هذا المنهج الفرعي يعني أنه عليك فقط كتابة الجزء الأكثر استخداماً في البرنامج مرة واحدة.

وأن لغة البيسيك تستخدم (GO SUB) كأمر يمكننا من إجراء نقل السيطرة من الجزء الرئيسي في البرنامج إلى البرنامج الفرعي ثم نستخدم الأمر RETURN للعودة ثانية إلي البرنامج الأصلي.

١٨- الأمر: إذهب إلى الفرع GO SUB Statement

وهو أمر يمكن من نقل السيطرة إلى منهج فرعي محدد بواسطة المستخدم ويكون بالشكل الآتي n GO SUB In

حيث (In) هي رقم السطر للأمر الأول في المنهج الفرعي وعند تنفيذه كان السيطرة تقفز إلى الأمر الأول في المنهج الفرعي ويستمر التنفيذ حتى تصل إلى الأمر (PRINT)

PROGRAM:

```
010 REM This Program Demonstates Some Statements
020 REM AS a Method of Control Transfer
030 PRINT "Subroutine Demonstration
040 LET X = 5
```

```
050 GOSUB 120
060 PRINT X, Y
070 GOSUB 120
080 PRINT X,Y
090 STOP
100 X = X * * 2
110 Y = X/10
120 RETURN
130 END
```

OUTPUT

```
Subroutine Demonstration
 25                    2.5
 625                   62.5
```

المنهج الفرعي في المثال عالية يتكون من الجملتين (١٠٠، ١١٠) وتم تنفيذه مرتين المرة الأولى تمت العودة (Return) إلى الأمر (٦٠) ـ والمرة الثانية إلى الأمر (٨٠) لاحظ أنه يمكن عمل أكثر من إشارة في البرنامج الرئيسي إلى نفس المنهج الفرعي ـ فإن السيطرة دائماً تعود إلى الحالة في البرنامج الرئيسي إلى النقطة (GO SUB) التي قفز منها إلى المنهج الفرعي وكذا يمكن استحضار منهج فرعي من منهج فرعي أخر.

كما أنه يمكن وضع (GO SUB...RETUN) خلال (FOR ... NEXT) دون أن يؤثر ذلك على تنفيذ

الأمر الراجع / عد RETURN Statement

يمكن من إعادة السيطرة إلى البرنامج الرئيسي أو المنهج الفرعي الذي بدأت منه (GO SUB) ـ عند تنفيذه فإن السيطرة يتم إرشادها إلى الأمر التنفيذي الأول التالي مباشرة للأمر (GO SUB) ـ لاحظ أنه لا حاجة لإعلان رقم السطر بعده

وإذا تطلب الأمر فإن المنهج الفرعي يمكن أن يحتوي على أكثر من (Return) واحده ـ والقيد الوحيد هو أن الأمر الأخير يجب أن تكون (Return)

١٩- الأبعاد: الأمر DIM Arrays : DIM Statement

يستخدم لتحديد حجم المصفوفات المتغيرة حتى يتم حجز مناطق التخزين لاستخدامهم خلال تنفيذ البرنامج ـ وأن لغة البيسيك مصممة لتتعامل مع الأبعاد/ الجداول ثنائية الأبعاد وطبعاً أحادية الأبعاد، وهي اختصار لكلمة (Demensial) ـ وتستخدم بالشكل التالي

n DIM V1 (S), V2 (S), . . . Vi (S)

حيث (V) متغير رقمي أو مسلسل يستخدم كمتغير مصفوف بالجمع داخل كشف (DIM) ـ (S) ثابت رقمي داخل قوسين يحدد حجم الصف وفي حالة المصفوفة ثنائية البعد فإنه من الضروري تحديد موصفين (بينهما فاصلة) مثل V(S1, S2) حيث (S1) عدد الصفوف، (S2) عدد الأعمدة ـ وكمثال DIM A (25), B$ (100), X (15, 20)

لاحظ أنه عندما يذكر أكثر من متغير صفي واحد بعد (DIM) فإنه يتم فصل المتغيرات بفاصلة .

ويمكن استخدام متغير صفري في البرنامج دون الحاجة لاستخدام (DIM) ولكن بحد أقصى (١١) موصف (صفر - ١٠) - أما عدد الموصفات التي يمكن استخدامها مع حالة (DIM) فتتراوح بين صفر وأي عدد يحدد في سطر (DIM) - ويكون البرنامج كالآتي

Program:

```
010 REM Program To Determine The Minimum Value
020 REM OF An Array Containing 20 Values
030    REM    ..... ..... ..... ..... .....
040 REM Declare Array Size
050 DIM N(20)
060 REM Set Up Loop To Fill Array With Values
070 FOR K = 1 To 20
080 READ N(K)
090 NEXT K
100 REM END OF LOOP
110 REM RETRIEVE Value Stored In First Location
120 REM And Hold Temporarily As Minimum Value
130 X = N(1)
140 REM Loop To Test Successive Values For A New
150 REM Minimum Value And Replace When Found
160 FOR I = 2 To 20
170 IF X < N(I) Then 190
```

```
180 LET X = N(I)
190 NEXT I
200 REM End OF Loop
210 PRINT "Minimum Value OF Array Is"; X
220 DATA 115,221,77,55,186,30,15,11,75
230 DATA 127,90,276,138,93,22,115,84
240 END
```

OUTPUT

```
Minimum Value OF Array IS 11
```

والبيسيك يزودنا بخيار لتغيير موصف الحد الأدني للصفوف حتى (1) ويمكن ذلك باستخدام الأمر (OPTION BASE) والذي يأخذ الشكل الآتي: n OPTION BASE i

لأن البرنامج الذي لا يشتمل على (OPTION BASE) فإن الحد الأدنى لموصفات الصفوف يكون دائماً صفر.

وعندما يتطلب الأمر أن يكون هذا الحد واحد ـ فإننا نستعمل هذه القاعدة قبل حالة (DIM) التي تعلن عن الأبعاد ـ أو قبل أول استخدام للأبعاد في حالة عدم استخدام DIM.

وفي بعض التطبيقات نجد أن الحد الأدنى هو واحد حيث أنه توجب حاجة لإمكانية إظهار حد الصفر خلال البرنامج.

وعموماً فإن الحاجة لاستخدام (OPTION BASE) هي حالة نادرة والآتي يعد مثال استخدم فيه الصفر كحد أدنى.

```
10 DIM X (20)
20 FOR K = 1 To 20
30 INPUT S (K)
40 NEXT K
```

ففي هذا المثلا نجد اختلاف السيطرة في (For Loop) تحدد أن قيمة الإدخال الأول تعين بواسطة X(1) ـ والثانية X(2) وهكذا مع ترك X(0) وعدم استخدامها ـ وهي من وجهة نظر تشغيل البرنامج فإنها غير موجودة على الإطلاق.

وهناك تطبيقات مختلفة تتعامل مع قاعدة الأبعاد ـ ففي بعض هذه التطبيقات فإن الحد الأدنى (1) ـ وقد يكون هناك اختلافات في شكل وطريقة استخدام هذا الخيار.

وسنورد هنا جدول أبعاد ذو بعد ثنائي لتوضيح أسلوب توصيف كل جزء في هذا الجدول

		K				
	1,1	1,2	1,3	1,4	1,5	1,6
	2,1	2,2	2,3	2,4	2,5	2,6
J	3,1	3,2	3,3	3,4	3,5	3,6
	4,1	4,2	4,3	4,4	4,5	4,6

شكل البرنامج:

```
070 FOR J = 1 To 4
080 FOR K = 1 To6
090 READ A (J,K)
100 NEXT K
110 NEXT J
```

فإذا اعتبرنا أن (J) هي (LOOP) خارجي، (K) هي (LOOP) داخلي فإن K تنفذ ٦ مرات في كل مرور خلال (J)

بمعنى أن كل (٦) عناصر من البيانات تتم قرائتهم في سطر واحد من الصفحة ويستمر البرنامج كالآتي.

```
300  DATA 1,2,3,4,5,6,7,8,9,10
310  DATA 11,12,13,14,15,16,17,18,19,20
```

مثال عن برنامج فرز أبجدي يوضح استخدام

IF...THEN,FOR - NEXT,DIME

```
010 REM ALPHABETICAL SORT PROGRAM FOR UP
    TO 100 NAMES
020   REM ..................
030 REM DECLARE ARRAY SIZE AND SET COUNT TO
    ZERO
040 DIM N$ (101)
050 LET K = O
060 PRINT "List Names To Be Sorted And. Key IN"
070 PRINT "No More At The End OF The List"
080 PRINT
090 REM SET UP LOOP TO INPUT DATA INTO ARRAY
095 REM Locations
100 FOR M = 1 TO 101
110 INPUT N$ (M)
120 REM TEST FOR END OF LIST
130 IF N $ (M) = "NO MORE" THEN 200
```

```
140 REM ADD 1 TO NAME COUNT
150 K = K + 1
160 NEXT M
170 END OF LOOP
180  REM ... ... ... ... ... ... ... ...
190 REM SET UP NESTED LOOPS TO CARRY OUT
    THE SORT
200 FOR M = 1 TO K - 1
210 FOR J = M+1 TO K
220 REM Compare Names And Swap Positions When
230 REM A Name of Lower Value Is Found
240 IF N$(M) < N$(J) Then 280
250 LET X$ = N$(M)
260 LET N$(M) = N$(J)
270 LET N$(J) = X$
280 NEXT J
290 REM END OF INNER LOOP
300 NEXT M
310 REM END OF OUTER LOOP
320 PRINT
330 PRINT
340 PRINT "THERE IS THE SARTED LIST"
350 PRINT
360 REM Set Up Loop To Print Out The Contents
370 REM Of The Array Now In Alphabetical Order
```

```
380 FOR J = 1 TO K
390 PRINT N$(J)
400 NEXT J
410 REM END OF LOOP
420 END
```

OUTPUT:

```
  List Names To Be Sorted And Key In
  No More At The End Of The List
? Peter
? Sarah
? Vicky
? Jane
? Simon
? Helen
? Richard
? No More
  Here Is The Sorted List
HELEN
JANE
PETER
RICHARD
SARAH
SIMON
VICKY
```

ملاحظات:

لاحظ أن عدد (١٠١) موضع لاسم قد تم تحديدهم حتى يمكن إضافة (No More) في نهاية الكشف ـ افحص بدقة الأوامر التي تتداول الأسماء بين مناطق التخزين بدون فقد أي اسم وهي المحددة بالقوس }

وتبعاً لتكملة دوامة For - Next (١٠٠ - ١٦٠) فإن الصف (N$) سيحتوي الأسماء من N$ (1) حتى N$ (8) (الأسماء + No More)

أما المتتالية الأولى فقد نفذت أولاً عندما كانت (M) تساوي (١)، (J) تساوي (٤) (حيث ا واحد، اثنين، ثلاثة).

وقد تناسينا المتوالية لأن الجملة (٢٤٠) هي:

```
240 IF N$(M) < N$(J) Then 280
250 LET X$ = N$(M)
260 LET N$(M) = N$(J)
270 LET N$(J) = X$
280 NEXT J
```

ومحتويات (X$) بعد تنفيذ ٢٥٠ عندما تساوي (M) واحد وتساوي (J) أربعة.

ويصبح الصف (N$) بعد تنفيذ (٢٦٠) عبارة عن خانات من N$(1) إلى N$(3) كالآتي مع ملاحظة تكرار اسم JANE

N$(1)	N$(2)	N$(3)	N$(4)	N$(5)	N$(6)	N$(7)	N$(8	X$
Jane	Sarah	Vicky	Jane	Simon	Helen	Richard	No More	Peter

كما يصبح الصف بعد تنفيذ ٢٧٠ كالآتي

Jane Sarah Vicky Peter Simon Helen Richard No More Peter

نلاحظ أن Jane ، Peter أتموا تبادل المراكز بالاستعانة بالمخزن الهيكلي.

التعليمات: Functions

هي عبارة عن تشغيل محدد يمكن استخدامه بتكرار من خلال برنامج وذلك بمجرد ذكره بالاسم.

ولغة البيسيك تزودك بمكتبة عامة للتعليمات سبق اعدادها لتحديد عدد من التعليمات الرقمية شائعة الطلب مثل الجذر التربيعي لرقم ما ـ وتسمح أيضاً للمترجمين بتحديد التعليمات الخاصة بهم باستخدام حالة DEF .

وعندما نشير إلى Function تنفذ فإن قيمة مفردة تعود إلى البرنامج والإشارة إلى Function يمكن عملها في أي مكان يسمح فيه بتعبير رقمي ويمكن أن يعمل مستقلاً أو كجزء من تعبير رقمي

المكتبة التقليدية للتعليمات Standard Library Functions

كل تعليمة يتم تشغيلها بذكر اسمها متبوعاً بما نسميه العنصر المتغير في التعليمة بين قوسين ـ وهذا التوضيح قد يكون في شكل أي تعبير رقمي ـ وعند التقدير فهي تزودنا بقيمة مفردة ممكن استخدامها عندئذ كبيانات بواسطة نفس التعليمة ـ والتعبير الرقمي الذي يشكل هذا التغيير يحتوي على إشارات لتعليمات أخرى

ففي المحددات التالية نجد أن (X) هي العنصر المتغير في الدالة ـ ونذكر أنها يمكن أن تكون تعبير رقمي مركب فمثلاً

في حساب المثلثات نجد ATN(X) COX(X) SIN(X) TAN(X)

وفي عمليات الجذور نجد EXP(X) - LOG(X) - SOR(X)

وفي العمليات الرياضية نجد ABS(X) - Int(X) - SGN(X)

٢٠ – التعليمة RND RND Function

وتعمل على الاختيار التالي والمسمى الرقم الاختياري المحتمل من أرقام اختيارية متتالية سبق تحديدها طبقاً للتطبيق

والأرقام تقع بين صفر، ١ ـ وتبدأ بالتوالي من نقطة بداية لا يمكن التنبؤ بها ـ وذلك يمكن إجراؤه باستخدام (RANDOMISE) قبل (RND) .

لاحظ أن غير مطلوب أي إضافات/ زيادة توضيح للتعليمة RND ولكن فقط للسيطرة على نقطة البداية للمتوالية ـ مثال (تمثيل رمي الزهر):

```
010 PRINT "Simulation of Dice Throwing
020 PRINT
030 REM SET UP LOOP TO SIMULATE 20 THROWS
040 FOR K = 1 TO 20
050 LET A = INT (RND*6)+1
060 PRINT
070 NEXT K
080 END
OUTPUT
Simulation of Dice Throwing
2 3 6 1 4 4 3 4 6 3 2 4 2 2 4 6 6 6 4
```

عندما ننظر السطر (٥٠) ـ نتذكر أن الرقم الذي يزودنا به رقم المتتالي الاختياري بواسطة (RND) دائما يكون بين صفر ، واحد ـ وبضرب هذا الرقم في ستة وتقريب النتيجة الخروج برقم صحيح بواسطة (INT) فسيصبح الرقم عن صفر إلى خمسة بإضافة واحد سيكون لدينا رقم من ١ إلى ٦ أي الوجوه الستة للزهر والأرقام الاختيارية يمكن استخدامها كبيانات اختبارية ـ كما يمكن استخدامه لتكرار نفس ترتيب الأرقام لمعرفة ما يحدث بعد تعديل البرنامج.

وإذا كررت البرنامج السابق مرة أخرى فستحصل على نفس النتائج.

٢١- أمر الاختيار RANDOMIZE Statement

وهو أمر يغير من نقطة البداية للرقم الاختيار المتتالي الذي تم تشغيله بواسطة (RND) ـ ويتم ذلك بطريقة يصعب التنبؤ بها بحيث أن البرنامج الذي يحتوي على هذه الحالة وعلى RND يختار أرقام مختلفة في كل مرة ينفذ فيها البرنامج ـ وتأخذ الشكل الآتي: RANDOMIZE n

وفي التنفيذ ، تولد نقطة بداية جديدة للاسم الاختياري المتوالي وبإضافة RANDOMIZE للمثال السابق سيكون كالتالي:

```
010 PRINT Simulation If Dice Throwing
020 PRINT
030 RANDOMIZE
040 FOR K = 1 TO 20
050 LET A = Int (RND * 6) + 1
060 PRINT A;
070 NEXT K
080 END
```

OUTPUT:

2 1 6 4 5 4 5 4 2 5 3 6 4 1 3 1 4 5 5 3

لاحظ أن الأمر (Randomize) لا يوجد في كل تطبيقات البيسيك.

٢٢- تعليمه: TAB TAB Function

للسيطرة على أوضاع الطباعة ـ وتستخدم فقط مع (PRINT) فنجد أن TAB(X) تحرك مكان الطباعة إلى رقم العمود المتحصل عليه من تقدير (X) مقربة إلى أقرب رقم صحيح ـ وقيمتها لا يصح أن تكون سلبية وفي العادة يوجد (٧٢) مكان للطباعة مرقمة من الصفر إلى (٧١)، فنجد أن TAB(8) تحرك نقطة الطباعة إلى العمود (٨) الذي هو في الحقيقة الوضع التاسع من اليسار كما يوضح ذلك المثل التالي:

```
010 PRINT "ABCDEFGH"
020 PRINT "01234567"
030 LET X = 8
040 PRINT TAB(X);"H"; TAB(6);"Z"
050 END
```

OUTPUT:

```
A B C D E F G H
0 1 2 3 4 5 6 7
      G       Z
```

تعليمات المستخدم المحددة User - defined Functions

يسمح البيسيك بخلق تعليمات مستخدم محددة ذات بعد واحد وذلك من خلال استخدام DEF ـ حيث يتم حساب قيمتها حيثما ظهر اسم التعليمة في البرنامج.

٢٣- الأمر حدد (Define) واختصارها DEF Statement

ويستخدم إما لإعطاء اسم تعليمة أو كتحديد تعليمه متكامل.

n DEF FNa (S1, S2, ...Si) =e

حيث (F Na) هذا اسم التعليمه S1 S2 ... Si هي أسلوب لتحديد التعليمه في شكل متغيرات، بصيغة رقمية بسيطة ومتنوعة، ن هو الناتج الحسابي للتعليمه في شكل تعبير رقمي.

وأول حرفين من اسم التعليمه يجب أن يكونا (FN) والحرف الثالث والذي يميز اسم عن الآخر يمكن أن يكون أي حرف أبجدي.

والمتغيرات الموضوعة في كشف المتغيرات (P) هي تلك التي تضبر في (e) وذلك يتطلب إعطاءهم قيم عند تنفيذ البرنامج عندما يمكن تقييم التعليمه وقتما يتم الرجوع إليها.

وتعتبر تلك توضيحات صورية لأنه عندما تقيم التعليمه Function، فإن القيم لتلك العناصر تحدد خلال التعليمه نفسها وليس خلال التحديد

والتغيير الذي يتم في قيمة الدالة يزودنا بوسائل للتعبير والقيم - وكمثال

```
010 REM AN EXAMPLE OF HOW TO CREATE A
    FUNCTION
020 REM AND HOW THE ARGUMEMENT
    STRUCTURE WORK
030 REM
040 REM FUNCTION TO CONVERT DOLLARS TO
    POUNDS
```

```
050 DEF FND(X) = X/2.40065
060 REM FUNCTION TO CONVERT YEN TO POUNDS
070 DEF FNY(X) = X/524
080 REM TEST OUT FUNCTIONS
090 LET C = 100
100 LET D = FND(C)
110 LET Y = FNY(C)
120 PRINT C; DOLLARS = ";D;"POUNDS"
130 PRINT
140 PRINT C;"YEN =";Y;"POUNDS"
150 END
```

OUTPUT:

```
100 DOLLARS = 41.5541 POUNDS
100 YEN = .19089 POUNDS
```

ملحوظات:

أن التعليمه التي استخدمناها لتغيير الدولار إلى جنيه محدده في الحالة (٥٠) ـ ولتغيير الين إلى جنيه في حالة (٧٠) ثم استخدمنا تعليمات في (١٠٠، ١١٠) ومعها قيمة (C) كمثال ١٠٠ تمر إلى كل التعليمات خلال التوضيح البديل (X) في كل من المحددات.

ولذا فإن التقييم كل تعليمه X = C = ١٠٠

ويجب تحديد التعليمه في البرنامج مثل أي إشارة إلي التعليمة ـ فمثلاً رقم الحالة (DEF) يجب أن يكون أقل من رقم الحالة التي روجعت فيها في المرة الأولى والتعبير الرقمي الذي يشكل التحديد الحسابي قد يحتوي على

متغيرات علاوة على ما هو مدون به من إيضاحات واضحة ـ ولنوضح هذا فإننا سنعدل البرنامج ليسمح للمستخدم بوضع قيمة العملة يومياً حسب أسعار البورصة ـ ولقد استبدلت بعملية تحويل الدولار (R1) ـ وتحويل الين (R2) في المثال التالي:

```
050 DEF FND(X) = X/R1
060 FUNCTION TO CONVERT YEN TO POUNDS
070 DEF FNY(X) = X/R2
080 REM TEST OUT FUNCTIONS
090 PRINT "DOLLAR AND YEN RATES"
100 INPUT R1,R2
120 PRINT
130 LET C = 100
140 LET D = FND(C)
150 LET Y = FNY(C)
160 PRINT C; "DOLLARS = "Y;"POUNDS"
170 PRINT
180 PRINT C; "YEN = "Y;"POUNDS"
190 END
```

ففي هذا المثال استطعنا إضافة إدخال أسعار العملة صطبقاً لآخر سعر أثناء تنفيذ البرنامج ورمزنا لها بـ (R2, R1) خلا جملة الإدخال الموجودة في السطر (١٠٠)

لاحظ أن المتغير (R1) يحدث في تحديد تعليمه FND في السطر (٥٠) وبالنسبة للمتغير (R2) فإنه يحدث في السطر (٧٠).

وعندما نطلب من تعليمه (FNI) في السطر (١٤٠) فإن قيمة (R1) ستكون متيسرة للتقييم بالرغم من أن (R1) ليس طرفاً في المناقشة والمثل. فإن قيمة (R2) ستكون متيسرة عندما تبدأ (FNY) في السطر (١٥٠) وتحديد التعليمه قد يحتوي على مراجعة لتعليمات محددة للمستخدم أخرى أو لأي تعليمات مكتب تقليدية. وكمثال

```
010 REM A FUNCTION TO ROUND VALUES
020 REM TO TWO DECIMAL PLACES
030 DEF FNA(Y) = INT(100*Y+0.5)/100
040 FOR K = 1 TO 3
050 PRINT "N=";
060 INPUT N
070 PRINT "ROUNDED VALUE IS";FNA(N)
080 NEXT K
090 END
```

OUTPUT:

```
N = ? 17.32715
ROUNDED VALUE IS 17.33
N = ? 99.0821
ROUNDED VALUE IS 99.08
```

يلاحظ أن بعض التطبيقات تسمح بأكثر من سطر فيه التعليمه DEF لتحديد اسم التعليمه، (FNEND) لإيضاح نهايه تحديد التعليمة والأسطر

بين (PRINT) تقدم التحديد الحسابي للعليمه وهناك أيضاً بعض التطبيقات التي تسمح بتعليمات مستخدم مسلسله ومحدودة

الملفات FILES

عندما تكون نتائج برنامج يراد بها أن تكون بيانات لبرنامج آخر فإنه من المفيد أن تكون قادراً على أن تترك نسخة من المعلومات في نظام الكمبيوتر جاهزة للإستخدام مرة أخرى عند الطلب. والمعلومات المتحصل عليها بهذه الطريقة تخزن في مخزن متوسط مستديم مثل الأسطوانة الممغنطه ـ وتجميع المعلومات هذه يطلق عليها ملف بيانات.

وإن ميكانكية الحفاظ على الملفات واسترجاعها من مخزن مستديم هي من الصفات الضرورة للنظام الآلي المستخدم ويجب أن تعرف أوامر النظام الخاصة بذلك.

وكما سبق أن قلنا فإن الملف يتم استدعاؤه في النظام باسم الملف والنظام يسترجع كتالوج لكل أسماء الملفات المستديمة وبذا يمكنه استرجاع أي ملف بالإشارة إلى اسمه.

وتطبيقات البيسيك المختلفة عادة ما تحتوي على عدد من التعليمات التي تمكن المستخدم من خلق ملفات خارجيةللبيانات. وبالتالي يمكن استخدامهم من خلال برامج البيسيك

وهذه التطبيقات أيضاً تختلف في طريقة تداول الملفات وتسكينها والسيطرة عليها ـ والأسس التي سيتم شرحها في هذا الفصل من تخدم كل التطبيقات بالرغم من أن التفاصيل الخصوصية قد تختلف فمثلاً تعليمات A READ file \ WRITE file تصنع البيانات على ملف خارجي وتعليمه

تقرأ البيانات على ملف خارجي تمت كتابته بواسطة a write file وتعليمه A PRINT file تضع البيانات على ملف خارجي وتعليمه A INPUT file تقرأ البيانات من ملف خارجي تمت كتابته بواسطة a Print file ، ونجد أن (PRINT , WRITE) أنهما تضعا البيانات باستخدام أشكال مختلفة ـ فالأمر (READ) يستطيع فقط أن يتفهم الشكل الذي تكتب به (WRITE) وكذا (INPUT) بالنسبة لشكل (Print) وقبل إمكانية كتابة ملف أو قراءته ـ فإن حالة الملف مطلوبه لتقريب رقم الملف إلى اسم ملف رقمي أبجدي (First) مثلاً فرقم الملف عادة ما يشار إليه بطريقة عادية ـ وعندما نستخدم ملف للكتابة أو القراءة أو للطباعة أو للإدخال في برنامج فإننا نجعله متصلاً بالملف الخارجي المناسب باستخدام الرقم الأبجدي (الأول/ الثاني...).

لاحظ أن ordinal fo 1 تستخدم كاسم الملف الأول الذي يتم الرجوع إليه، (2) لاسم الملف الثاني وهكذا

وكما سنرى الأمثلة التي تلي "#" (Hash) وستستخدم في كل حالات الملفات في البيسيك.

مثال: يقدم # WRITE # ، FILE #

Program:

```
010 FILE # 1, "TEST.A"
020 FOR K = 1 TO 3
030 INPUT N
040 T = N ^ 2.5
050 WRITE # 1, N, T
```

```
060 NEXT K
070 END
OUTPUT:
? 40
? 2
? 27
```

تعليق

السطر ١٠ يحدد اسم الملف الخارجي الذي يلزم تواجده للاستخدام طوال تنفيذ البرنامج ـ وأيضاً الرقم الأبجدي الذي سيستخدم كمرجع من خلال البرنامج إلى الملف الخارجي.

"TEST A" هو الاسم الذي يعرفه النظام للملف ـ ويظهر مرة واحدة فقط في البرنامج.

وعلى كل فإن الرقم الأبجدي يظهر حيثما وجدت الحاجة لتشغيل الملف الخارجي وفي هذه الحال نجد السطر (٥٠) لتأكيد أن ارتباط القيم بالمدلولان (I , N) قد تم تسجيله في الملف الخارجي

لاحظ أن ما يظهر في المستخرج سيكون فقط القيم التي تم إدخالها بواسطة المبرمج في التنفيذ المتكرر للسطر (٣٠)

والنتائج الصحيحة لن تكون مرئية حيث أن النتائج قد نسخت إلى الملف ـ ولإثبات ذلك يمكن استعادتهم من خلال البرنامج التالي

```
010 FILE # 1 , "TEST A"
020 READ # 1 , A , B
030 PRINT A , B
```

040 NODATA # 1,60
050 GO TO 20
060 END
OUTPUT:
40 100
2 5
27 67.5

تعليق:

السطر ١٠ يحدد كما قلنا اسم الملف الذي سيستخدمه البرنامج ـ والسطر ٢٠ يقرأ القيم الموجودة على الملف #١.

مثال: "TEST A" التي أوجدناها في المثال السابق وحفظنا محتوياتها في A, B . وهذه القيم يتم طبعها بالسطر (٣٠).

لاحظ استخدام NODATA # لاختبار ما إذا كانت هناك قيم ما زالت موجودة في الملف ـ وبعض تطبيقات البيسيك تستخدم بدلاً منها IF MORE أو IF END #.

وهذه التطبيقات أيضاً تستخدم (RESTORE #) أو (RESET #) لإعادة المؤشر (Pointer) إلى بداية الملف ـ وقد سبق بحث وظيفة المؤشر في فصول سابقة.

يلاحظ أن القواعد التي تطبق على إقرأ، إطبع، إدخل عموماً تنطبق على حالات الملف المماثلة وكمثال المتغيرات المحددة يجب أن تكون من نفس نوع البيانات التي تنتمي إليها/ تشير إليها وملف البيانات يمكن أيضاً أن يجهز كملف خارجي عمومي وفعال لأي برنامج بإدخال بياناته ـ ومطلوب فقط تعليمات من النظام لتسمية الملف وإدخال بياناته ببساطة بعدها مباشرة

وبعض النظم تنظم الملفات بحيث يكون لكل سطر منه رقم سطر يسبقه حيث أنه من المفيد جداً إجراء ترقيم للأسطر

والمثال التالي يبرز أهمية عملية ترقيم الأسطر

```
010 MR. MAHER WAHBA
020 ALEXANDRIA
030 MIAMI
040 SHARIA SALEM
050 GAWHARA BUILDING
060 ADDRESS END
```

والبرنامج التالي لقراءة هذا الملف الخارجي بواسطة استخدام INPUT

```
010 FILE # 1, "ADDRESS"
020 LET L$ = "ADDRESS END"
030 FOR J = 1 TO 8
040 INPUT # 1, N, A$(J)
050 IF A$(J) = L$ THEN 70
060 NEXT J
070 FOR K = 1 TO J - 1
080 PRINT A$(K)
090 NEXT K
100 END
```

OUTPUT:

```
MR. MAHER WAHBA
ALEXANDRIA
MIAMI
```

SHARIA SALEM
GAWHARA BUILDING

تعليق:

السطر (٤٠) يدخل بيانات من الملف الخارجي المصاحب للرقم الأبجدي (١) وفي هذه الحالة فإن الملف المسمى (ADDRESS) يتم تخليقه بإدخال البيانات كما تم في المثال عاليه.

لاحظ أن رقم السطر في ملف البيانات تم تمت قراءته ويتم تخزينه مرة أخرى في (N).

والمتغير (N) لا يستخدم في مكان اخر من البرنامج ـ والغرض منه هو التأكد من أن البيانات الصحيحة في كل سطر قد تم التقاطها وتخزينها في المصفوفة (A$).

لاحظ أيضاً أنه في هذا المثال أن اختبار عدم تواجد بيانات أخرى قد تم مستخدما عنصراً من ملف البيانات (ADDRESS END) وبالإضافة يجب أن نكرر مرة أخرى أن تداول الملف يتوقف على النظام المستخدم وتطبيق البيسيك المستخدم ـ ولاحظ أن تعليمات النظام المحلية للحفاظ على الملفات وترتيبها لم ترد في هذا المرشد

خلاصه اوامر الجدولة.

كثيراً من تطبيقات البيسيك قد أوجدت أسلوباً لتنفيذ الجداول الإدارية ـ والأساليب المتيسرة قد تكون واحدة من تلك التي تم شرحها في هذه الخلاصة.

وهذه التعليمه تنظم ملئ البيانات في أعمدة منظمة MAT READ وتأتي هذه البيانات من تلك المدخلة في الحالة DATA .

وهي تعليمه تمكن من ملىء الأعمدة بالبيانات MAT INPUT المدخلة بنفس ترتيبها بمجرد الإدخال

تطبع البيانات في أعمدة MAT PRINT

Program:

```
010  DIM X (3,3)
020  MAT READ X
030  MAT PRINT X
040  DATA 1, 2, 3, 4, 5
050  DATA 6, 7, 8, 9
```

OUTPUT:

```
1        2        3
4        5        6
7        8        9
```

تعليق السطر (١٠) يكفي لتحديد كل البيانات التسعة وأن الجدول ثنائي البعد أو كما ذكرنا MATRIX X

والسطر (٣٠) يخرج محتويات الجدول (X) ـ لاحظ أنه من الضروري أن تحدد أبعاد الجدول قبل استخدام (MAT) كما هو مكتوب في السطر (١٠) ولاحظ أيضاً أن (MAT PRINT) تخرج الأعمدة كما لو كانت فاصلا لطباعة فاصلة (,) وأن الطابعة تترك سطر فارغ بين السطور المكتوبة

Program:

```
010 DIM A(2,3)
020 PRINT "ORGANIZE DATA ROW BY ROW"
030 MAT INPUT A
040 PRINT
050 MAT PRINT A;
060 END
```

OUTPUT:

```
? 1, 2, 3
? 4, 5, 6
 1    2    3
 4    5    6
```

تعليق:

في السطر (٣٠) يتوقف البرنامج في انتظار إدخال البيانات نجد أن الفاصلة المنقوطة (;) في نهاية السطر (٥٠) تسبب طباعة الجدول في شكل متلاحق بعكس الحالة السابقة

ويمكن إجراء عمليات حسابية بواسطة:

MAT Assignment	MAT A = B
MAT Addition	MAT C = A + B
MAT Subtraction	MAT C = A - B
MAT Scalar Multiplication	MAT C = (n) * A
MAT Multiplication	MAT D = E * F

وفي هذه الأمثلة فإن المدلولات (A , B , C) يجب دائماً أن تكون طرفاً في مصفوفة ذات أبعاد ـ وفي المثال الأخير يجب أن يكون عدد الأعمدة في (B) مساوياً للصفوف في (A).

وكذا رقم الصفوف في (A) مساوي لرقم الصفوف في (B) ورقم الأعمدة (D) مساوي لعدد الأعمدة في (A) وفي المثال الذي يسبقه فإن (n) يمكن أن تكون أي تعبير رقمي وهناك أيضاً عدد من استخدامات (MAT) الأخرى مثل

إنشاء تعريف للجدول يتكون من وحدات على MAT IDN حول المحور الرئيسي، أصفار في أي مكان آخر.

ولعمل جدول يتكون من آحاد فنستخدم MAT CON

وكذا لعمل جدول يتكون من أصفار نستخدم MAT ZER

وإذا أردنا جدول متحرك على طول القطر فإننا نستخدم MAT TRN

أما الجدول الذي يحسب الزيادة الطردية لمربع جدول فنستخدم MAT INV

* * *

أمثلة لبرامج بيسيك

١ مخزن يحتوي على سلع مختلفة وكل سلعه لها سعر بيع وإجمالي تكلفه - أرسم جدولاً لإظهار هذه البيانات ومعرفة الربح الذي يحققه بيع كل سلعة مع توضيح الربح كنسبة مئوية

PROGRAM:

```
010 REM STOCK INVENTORY PROGRAM
020 PRINT TAB(25);"INVENTORY TABLE"
030 PRINT
040 PRINT"ITEM",TAB(12);"SALES PRICE";
050 PRINT TAB(29);"NET COST";TAB(45);"PROFIT";
060 PRINT TAB(60);"PERCENT"
070 READ N$
080 IF N$ = "STOP" THEN 200
090 READ X, Y
100 LET P = X - Y
110 LET C = P/X * 100
120 PRINT N$,X,Y,P,C
130 GO TO 70
140 DATA "RULER", 1.2, 0.88
150 DATA "CARD", 0.35, 0.22
160 DATA "RAZOR", 2.50, 2.05
170 DATA "INK", 0.55, 0.39
180 DATA "PEN", 0.15, 0.09
190 DATA "STOP"
900 END
```

OUTPUT:

INVENTORY TABLE

ITEM	SALES PRICE	NET COST	PROFIT	PERCENT
RULER	1.20	0.88	0.32	35.00
CARD	0.35	0.22	0.13	37.10
RAZOR	2.50	2.05	0.45	18.00
INK	0.55	0.39	0.16	30.09
PEN	0.15	0.09	0.06	40.00

* * *

٢- اكتب برنامج لجدول تحويل جنيه مصري إلى دولار، فرنك، مارك، ين ياباني في تاريخ معين.

PROGRAM:

```
010 REM CURRENCY CONVERSION PROGRAM
020 PRINT "CURRENCY CONVERSION TABLES"
030 PRINT
040 REM INPUT DATE
050 PRINT"DATE";
060 INPUT D$
070 REM  $ : DOLLAR     F : FRANK
080 REM  M : MARK       Y : YEN
090 PRINT "RATES OF EXCHANGE"
100 PRINT "DOLLAR, FRANK, MARK, YEN
110 INPUT S, F, M, Y
120 PRINT " =    "
```

```
130 PRINT "BASED ON EXCHANGE RATES VALID
    ON"; D$
140 PRINT
150 PRINT "L - E","DOLLAR","FRANK"
160 PRINT "MARK","YEN"
170 PRINT
180 REM SET UP PARAMETERS TO PASS TO COVERSION
190 REM ROUTINE
200 REM S : START VALUE  E: END VALUE
210 REM P : STEP VALUE
220 LET S = 1
230 LET E = 9
240 LET P = 1
250 GO SUB 340
260 REM RESET PARAMETERS
270 LET S = 10
280 LET E = 100
290 LET P = 10
300 GO SUB 340
310 GO TO 999
320 REM LOOP TO COMPUTE VALUES AND PRINT THEM
330 REM OUT
340 FOR J = S TO E STEP P
350 LET C1 = J * $
360 LET C2 = J * F
```

```
370 LET C3 = J * M
380 LET C4 = J * Y
390 PRINT J, C1, C2, C3, C4
400 NEXT J
410 RETURN
420 END
```

OUTPUT:

```
CURRENCY CONCERSION TABLES
DATE ? June 7 1987
RATES OF EXCHANGE
DOLLAR, FRANK, MARK, YEN
=========================================
```

والآتي برنامج لتعداد السكان.

نفترض أن إمارة الشارقة مثلاً عدد سكانها مليون نسمة ـ ولنفترض أيضاً أن عدد المواليد في الإمارة نسبته إلى عدد السكان ٦٪ سنوياً ـ بينما عدد الوفيات إلى عدد السكان ٤٪ سنوياً فكم يبلغ عدد سكان الإمارة في الأعوام الخمسة القادمة.

التعداد S ـ نسبة المواليد C ـ نسبة الوفيات D

PROGRAM:

```
010 REM EMARAAT EL SHARQAA POPULATION
020 LET S = 1000000
030 PRINT "YEAR", "POPULATION"
040 FOR Y = 1 TO 5
```

```
050 LET C = .06 * S
060 LET D = .04 * S
070 LET S = S + C - D
080 PRINT Y, S
090 NEXT Y
100 END
RUN
YEAR   POPULATION
1  1020000
2  1040400
3  1061208
4  1082432.2
5  1104080.8
OK
```

المراجع

أولاً: الأجنبية

- Fundamentals of programming in Basic R.C.Neckerson.
- Theory and Problems of programming with Basic
By: B. S. gottfried
- Pitman Prohramming Pocket Guide Roger Hunt
- Basic with Buisiness Applications By: R.W.LOTT

ثانيا: العربية

معجم مصطلحات الحاسبات الالكترونية مركز الأهرام

استخدام التعليمات الرئيسية للغة البيسيك

في بعض أنواع أجهزة الكمبيوتر الشخصي

أولاً الاختصارات التي سنستخدمها

exp	تعبير حسابي
cst	رقم ثابت
Fnm	اسم الملف
Str	جملة
stm	تعليمة
lno	رقم السطر
Var	متغيرة
adr	عنوان
Param	معامل
mode	نوع الملف

ثانياً التعليمات والاوامر المشترك بين أنواع الاجهزة التي سنذكرها

CLOSE # Fnm	اغلق ملفات الاسطوانة
LOAD "Fnm"	انقل برامج من الاسطوانة للذاكرة
LET	للتعريف بقيم ثابتة
INPUT	لتعيين قيم البيانات للمتغيرات
GO TO	لممارسة انتقال غير مشروط
IF ... THEN	لعمل الجملة الشرطية
FOR ...NEXT	لتكرار التعليمة

للنقل الشرطي	ON ... GO To
لتسيير البرنامج	RUN
لتحديد حجم المصفوفة	DIM
للسيطرة على أوضاع الطباعة	TAB
للطباعة	P R I N T
لإدخال بيانات	DATA
لعرض أسطر من البرنامج	LIST
لإعادة المؤشر الى البداية	RESTORE
لنقل البرنامج الى الاسطوانة	SAVE
لانهاء سير البرنامج	END
لكتابة الملاحظات والبيانات	REM
لايقاف البرنامج	STOP
للتحديد الكامل للتعليمة	DEF FN
للانتقال الى روتين فرعي	GOSUB
للعودة الى قبل GOSUB	RETURN

ثالثاً: أنواع الكمبيوترات الشخصية وبعض التعليمات المستخدمة مع كل منها:

١- الكمبيوتر الشخصي أي . بي . إم IBM P. C

افتح ملف في الاسطرانه	OPEN Fnm
اغلق الملف وتعود الى نظام التشغيل	SYSTEM
لتحرير سطر من البرنامج	EDIT lno

لمحو أسطر من البرنامج	DELETE[lno][-lno]
لمعرفة حجم الذاكرة المتبقي	FRE(exp)
لقراءة تسجيله من اسطوانه	GET[*][#nm[,record no]
لقراءة بيانات مدخله من لوحة المفاتيح	INPUT [Prompt] var [,var]
توقيف سير البرنامج لمدة محددة	WAIT port,exp[,exp]
استئناف سير البرنامج	CONT
لمحو قيمة المتغيرات	CLEAR
للخروج من أي خطأ إذا حدث	ON ERROR GOTO lno

٢- الكمبيوتر الشخصي كومودور ٦٤ أو ١٢٨ — COMODOR 64, 128

لفتح ملف في الاسطوانة	OPEN # exp, Fno/mode, "Fnm"
تحرير سطر من البرنامج	[cursor editing]
لمعرفة حجم الذاكرة المتبقي	FRE(x)
لقراءة تسجيله من اسطوانة	GET # Fno,var
لقراءة بيانات مدخلة من لوحة المفاتيح	INPUT[str;]var[,var]
توقيف سير البرنامج لمدة محددة	WAIT adr, exp[,exp]
استئناف سير البرنامج	CONT
لمحو قيمة المتغيرات	CLR

٣- الكمبيوتر الشخصي أبل ٢ APPLE II

لفتح ملف في الاسطوانة	OPEN Fnm,Parameter
لتحرير سطر من البرنامج يتم على الشاشة مستخدما المفتاح	ECS
لمحو أسطر من البرنامج	DEL lno,lno
لمعرفة حجم الذاكرة المتبقي	FRE(exp)
لقراءة تسجيلة من أسطوانة	INPUT var[,var,...]
لقراءة بيانات مدخلة من لوحة المفاتيح	INPUT[str;]var[,var]
توقيف سير البرنامج لمدة محددة	WAIT adr, exp[,exp]
استئناف سير البرنامج	CONT
لمحو قيمة المتغيرات	CLEAR
للخروج من الخطأ إذا حيث	ONERR GOTO lno

٤- الكمبيوتر الشخصي أتاري ATARI

لفتح ملف في الاسطوانة	OPEN # Fnm, var, Fnm
لاغلاق الملفات والعودة الى نظام التشغيل	BYE
لتحرير سطر من البرنامج	[cursor editing]
لمحو أسطر من البرنامج	[cursor editing]
لمعرفة حجم الذاكرة المتبقي	FRE(exp)
لقراءة تسجيلة من اسطوانة	INPUT [str;] var[,var]
لقراءة بيانات مدخلة من لوحة المفاتيح	INPUT [str;] var[,var]
لمحو قيمة المتغيرات	CLR
للخروج من الخطأ إذا حدث	TRAP lno or var or exp

٥ الكمبيوتر الشخصي شارب م ز ـ ٨٠ ك SHARP MZ 80K

لفتح ملف في الاسطوانة للقراءة	RI OPEN
للكتابه	WOPEN
لاغلاق الملفات والعودة إلى النظام التشغيل	BYE
تحرير سطر من البرنامج	[cursor climing]
لمعرفة حجم الذاكرة المتبقي	SIZE
لقراءة تسجيلة من اسطوانة	INPUT # record
لقراءة بيانات مدخلة من لوحة المفاتيح	INPUT [STR] var
لمحو قيمة المتغيرات	[var]
	CLR

٦ـ الكمبيوتر الشخصي سيكتروم SPECTRUM

لفتح ملف في الاسطوانة	OPEN # Fnm
لتحرير سطر من البرنامج	EDIT [lno]
لتوقيف سير البرنامج لمدة محددة	PAUSE
لاستئناف سير البرنامج	CONT
لقراءة بيانات مدخلة من لوحة المفاتيح	INPUT [STR] var
	[var]
لمحو قيمة المتغيرات	CLEAR [var]

للمعاونة في توضيح هذه الخطوة فإننا نستخدم الرسوم الإنسيابية (Flowchart)