تطور أساليب البرمجة

تطور أساليب البرمجة

هناك ثلاثة أساليب للبرمجة :-

1- البرمجة العشوائية (التقليدية):يتم التركيز في هذا الأسلوب على حل المسألة برمجيا للوصول إلى الهدف دون النظر إلى عملية تنظيم البرامج مما يؤدي إلى إنتاج برامج صعبة التطوير أو اكتشاف الأخطاء ولا يظهر مشاكل هذا الأسلوب ألا إذا كان البرامج كبيرا أما في حالة البرامج الصغيرة والبسيطة ربما كان هذا الأسلوب مناسب لها

2- البرمجة المهيكلة:يعتمد هذا الأسلوب على تجزئة البرنامج إلى عدة برامج جزئية أو فرعية حيث يتم الربط بين هذه البرامج الفرعية لتشكيل البرنامج العام ويحتاج هذا الأسلوب إلى تخطيط جيد وتظهر فاعليته في حالة المسائل متوسطة الحجم كما يسهل اكتشاف الأخطاء بهذا الأسلوب وأجراء عمليات التطوير بالإضافة إلى عدم تكرار المقاطع البرمجية .

3- البرمجة الكائنية الموجهة:هذا الأسلوب الشائع حاليا وهو يحاكي الواقع ويعتمد فيه على بناء الكائنات التي تضم البيانات والإجراءات وظهر هذا الأسلوب في بداية السبعينات حيث واجهت البرمجة الهيكلية عدة مشاكل منها عدم حماية البيانات وعدم القدرة على تمثيل الواقع وصعوبة تقسيم البرنامج إلى إجراءات .وقد اعتمد هذا الأسلوب على مفهوم الكائن والذي يمثل نمط جديد وينتج من دمج البيانات (Data)والإجراءات (Procedure). وعند دراسة المشكلة يتم التركيز على الكائنات بدلا من الإجراءات ويؤدي إلى تطابق بين المفهوم البرمجي والواقع الفعلي للمشكلة.

مقدمة لأسلوب البرمجة المهيكلة
structured programming

ان الطرق المتبعة في البرمجة التقليدية تنظر الى صياغة البرامج على أنها عملية إبداع شخصي وكان الاعتقاد السائد عن المبرمج الجيد انه الشخص الذي تكون لديه القدرة على كتابة برامج ذكية إذ غالبا ما تعطي كلمة ذكي معنى مرادفا لكون البرنامج معقد ومبهما ومعتمدا على بعض الحيل البرمجية. ان التعامل مع برامج من هذا النوع يبرز مصاعب تتجلى في كون هذه البرامج مرمزة بصورة مبهمة يصعب فهمها وتتبعها من قبل أي شخص آخر غير كاتب البرنامج وحتى الاخير فانه قد يجد صعوبة في فهم برنامجه بعد مرور اشهر أو سنوات على كتابته. ان واقع الحال العملي يفرض ان تعدل وتغير بعض البرامج بين آونة وأخرى وذلك لأجل ان تواكب هذه البرامج متطلبات استجدت لظروف معينة.

لقد انعكس ذلك على أسلوب البرمجة المتبعة، بحيث أصبحت الحاجة ملحة لصياغة برامج تسهل صيانتها وتطويرها .وقد أعطى هذا الموضوع اهتماما بالغا وكرست الجهود من اجل إيجاد الأساليب المناسبة لذلك. وقد ظهر أسلوب البرمجة المهيكلة الذي أصبح يشير إلى مجموعة الأفكار والمبادئ والممارسات التي استنبطت من الجهود المبذولة لأجل صياغة برامج ذات هيكل واضح متميز مطابق لهيكل المشاكل المطلوب حلها. إذ تمتاز هذه البرامج بسهولة قراءتها وتتبعها ومن ثم صيانتها.

وكما هو الحال مع الموضوعات العلمية الجديدة فان موضوع البرمجة المهيكلة قد رافقته منذ البداية بعض التصورات والمفاهيم الخاطئة نتيجة لقصور في استيعاب إبعاد الموضوع. فقد اعتقد البعض أن البرمجة المهيكلة وبسهولة هي الاقتصار على استبعاد تعبير GO TO من لغات البرمجة في الحقيقة أن أهداف البرمجة المهيكلة أكبر من مجرد إقصاء تعبير GO TO إذ يعد هذا الهدف سلبيا .ومن المهم بالنسبة للمبرمجين أن يتعرفوا على المنافع الايجابية التي في الإمكان جنيها من خلال تطبيق أسلوب البرمجة المهيكلة . و سوف نأتي هنا على تقديم الملامح الدقيقة التي يمكن بواسطتها توضيح معالم الموضوع من دون أن نحدد أنفسنا بإعطاء صيغة معينة لهذا المصطلح إذ أن أي محاولة من هذا النوع ستكون مختصرة جدا ولا تعكس واقع الحال. .

ان الهدف الرئيسي الذي نسعى الى تحقيقه هو كتابة برامج تعمل بصورة صحيحة 100% فضلا عن سهولة قراءتها و تتبعها و من ثم صيانتها و تطويرها. فإذا وضعنا هذه المجموعة من الأهداف نصب أعيننا و نظرنا الى مواصفات المسائل أو المشاكل التي يطلب من المبرمج صياغة برامج لها نجد ان الكثير من هذه المشاكل المطروحة مواصفاتها معقدة بطبعها .وهنا يبرز سؤال لا يمكن إغفاله هو :كيف نزيل التعقيد المتأصل بهذه المشاكل؟ عند الإجابة عن هذا السؤال ، يجب ان لا ننسى حقيقة كون العقل البشري ذا قابلية محدودة ولا يمكنه التفكير بعدة أمور في آن واحد لقد انعكست آثار هذه الحقيقة على أسلوب البرمجة المهيكلة واقترحت سبل عديدة لتجزئة المشاكل المعقدة إلى عدد من المشاكل الصغيرة .

إن أسلوب ((التقسيم و التغلب على المشاكل)) يتركز في وضع الهيكل الصحيح للمشكلة ككل على شكل مستويات هرمية مفصلة .أما تفاصيل ومكونات كل جزء من الأجزاء المنفصلة فأنها تؤجل لحين الانتهاء من أعداد الهيكل . أن اللجوء إلى أسلوب البرمجة المهيكلة يساعد في الإقلال و في السيطرة على التعقيدات الموجودة أصلا في المشاكل، والتي من غير الممكن التخلص منها نهائيا. و فوق ذلك ان التقسيم الى وحدات و استخدام البرمجة المهيكلة يضفي وضوحاً و بساطة على أجزاء و وحدات البرنامج المحددة و المعرفة بصورة جيدة . فعند تقسيم البرنامج الى أجزاء صغيرة منفصلة بحيث يتخصص كل جزء للقيام بعمل ما ؛يسهل حين ذاك حصر ومعالجة التعقيدات التي توافق المشكلة ، وإذا ما سلمنا بأن هذه هي الطريقة المثلى لحل المشاكل فيجب أن نهتم أيضا بالطريقة التي يتم بها ربط هذه الأجزاء فيما بينها لتعطينا من ثم برنامجا بسيطا يعمل على حل المشكلة بطريقة نظامية سليمة .

التراكيب المستخدمة في البرمجة المهيكلة

أن الطريقة الميسرة لتكوين برامج مفهومة وواضحة هي في أن يتم تنظيم كل برنامج على شكل سلسلة متتالية Sequenceمن الإجراءات من دون أي تفرع أو تكرار .فإذا استعرضنا السلسلة الآتية من الإجراءات وتصورنا أنها تؤلف برنامجا .

S1

S2

S3

وجدنا أن الإجراء (S1)سينجز أولا ومن ثم الإجراء (S2) وأخيرا (S3)ولكن كما هو بديهي ، فان البرامج بصورة عامة اعقد بكثير من ذلك ، إذ يرد فيها تكرار دائري LOOPS لبعض الفقرات، أو تفرع BRANCH من فقرة إلى أخرى في البرنامج . و إلى حد ما فإن ما نحاول تفصيله هنا كيفية إخفاء هذه الحقيقة . أو بمعنى أخر كيف يتسنى لنا كتابة برامج تحتوي في الواقع على تكرار دائري أو تفرعات بحيث يمكن قراءتها و فهمها على أنها سلسلة بسيطة من الإجراءات ؟ انه لمن المجدي أن نعد تسلسل انجاز البرنامج يتم كما يأتي ((الأجراء S1أولا ,يتبعه S2, وأخيرا S3))بالرغم من إننا على علم مسبق من أنS2 تشمل عملية تكرار دائري.

عند التعمق في توضيح ما تقدم ، نجد ان هناك نماذج من التراكيب التي من الأفضل معاملتها على أساس كتلة واحدة عند القراءة ،تسهيلا لفهم الأعمال التي يقوم بها البرنامج .

أن النظرية الأساسية للبرمجة المهيكلة ترتكز على مبدأ إمكانية صياغة البرنامج المضبوط –المتميز بمدخل واحد ومخرج واحد – باستخدام ثلاث تراكيب رئيسية هي :

1. تركيب السلسلة(SEQUENCE)
2. تركيب الاختبار(IF-THEN-ELSE)
3. تركيب التكرار(DO-WHILS)

وقد أثبتت هذه النظرية من خلال بحث (1) قدم سنة 1966 أن جوهر البحث ينص على أن كل المخططات الانسيابية للبرامج مهما كانت درجة تعقيدها يجب أن تكون قابلة للاختزال إلى التراكيب الثلاث أعلاه. أو بصيغة أخرى أن أي مخطط انسيابي لا يمكن اختزاله إلى تلك التراكيب هو معقد جدا ويجب أن يعاد تصميمه.

أن هنالك وجهين مهمين لهذا البحث أولهما:إمكانية كتابة أي برنامج من دون اللجوء إلى الاستخدام الواضح لإيعاز GO TO وثانيهما :أن هيكل كل تركيب سيكون له نقطة دخول ENTRY واحدة ومخرجEXIT واحد. وسنأتي هنا على ذكر كل من هذه التراكيب بشي من التفصيل.

1-تركيب السلسلةSEQUENCE

يشير هذا التركيب إلى المجرى المنطقي المتتالي للبرنامج فمثلا ,أذا استخدمنا لغة كوبول COBOL لكتابة الايعازات التالية :

```
ADD TOTAL-1 TO TOTAL-2 GIVING GRAND-TOTAL.
MOVE GRAND-TOTAL TO PRINT- TOTAL.
WRITE PRINT-RECORD.
```

فان انجاز هذه الايعازات سيتم واحدا بعد الأخر. ويمكن تمثيل هذه الايعازات من خلال مخطط السلسلة الموضح في الشكل (4أ-1)

2-تركيب الاختيار IF-THEN-ELSE

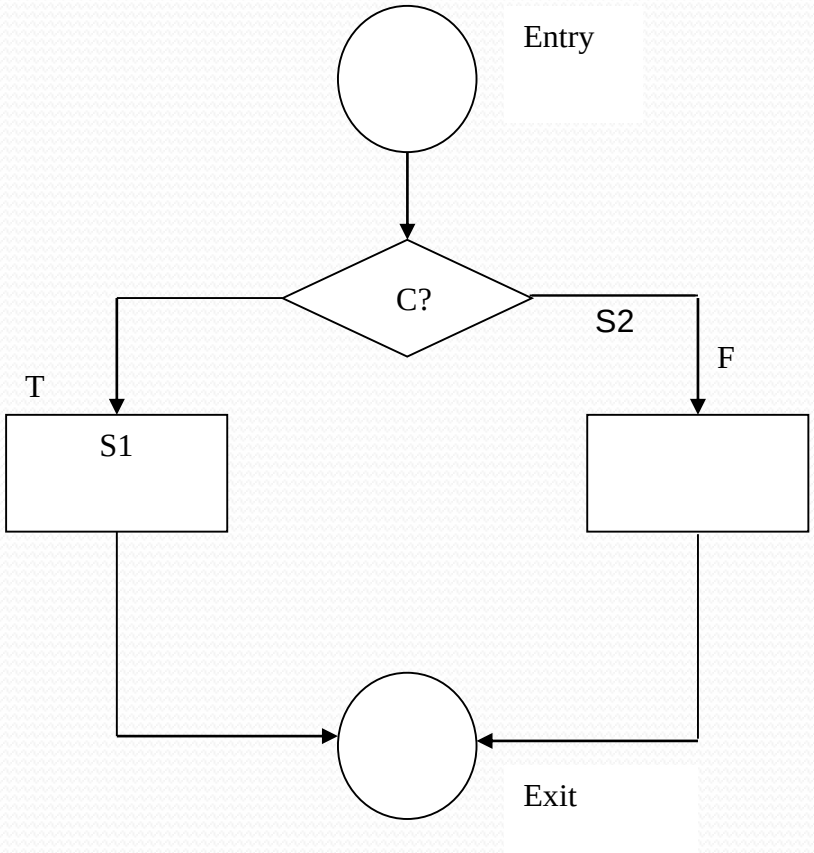
عندما يعترض مجرى البرنامج تركيب شرطي ،فإنه يؤدي إلى تفرع البرنامج إلى فرعين يجب الاختيار بينهما .و يتم اختيار التسلسل المنطقي الملائم تبعا للشرط أما أن يصح الشرط true أو لا يصح false ولتوضيح ذلك نستعين بالمثال التالي:

```
IF   QUANTITY>100
MULTIPLY 0.2  BY PRICE GIVING VALUE
ELSE  MULTIPLY 0.1 BY PRICE GIVING VALUE
```

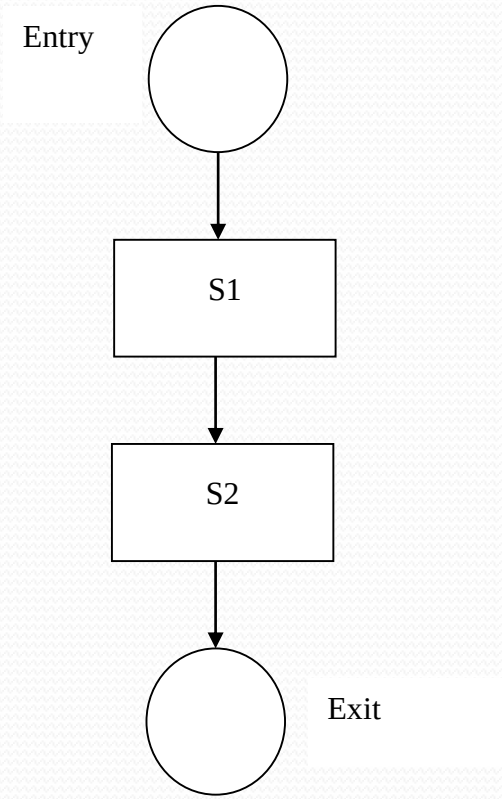
وقد تصادف في بعض الأحيان حالات شرطية أبسط من هذا يترتب عنها تفرع واحد يتبعه مجرى البرنامج في حالة التي يصح فيها الشرط ،وبخلافه فان البرنامج يواصل سيرة لانجاز الإيعاز التالي في السلسلة مثلا :

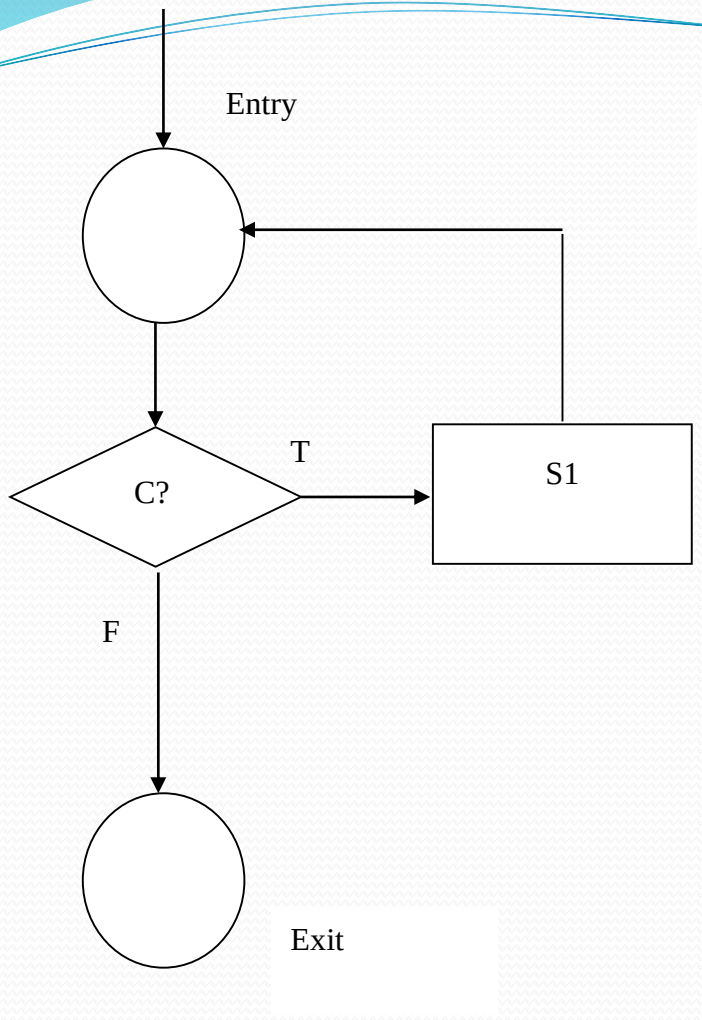
```
IF QUANTITY>100 MULTIPLY 0.2 BY PRICE
```

شكل (4ب-1)

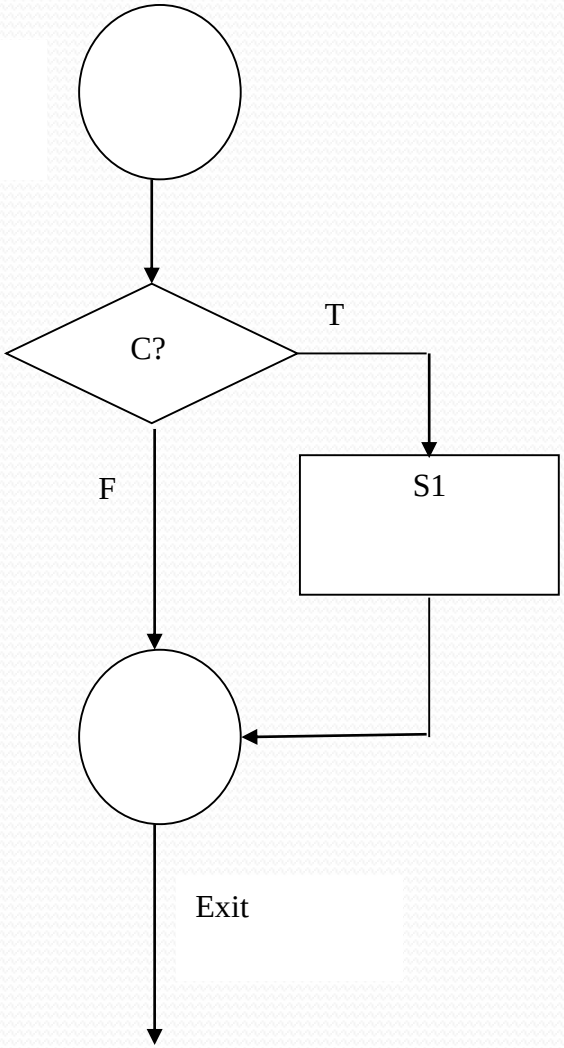


شكل (4أ-1)

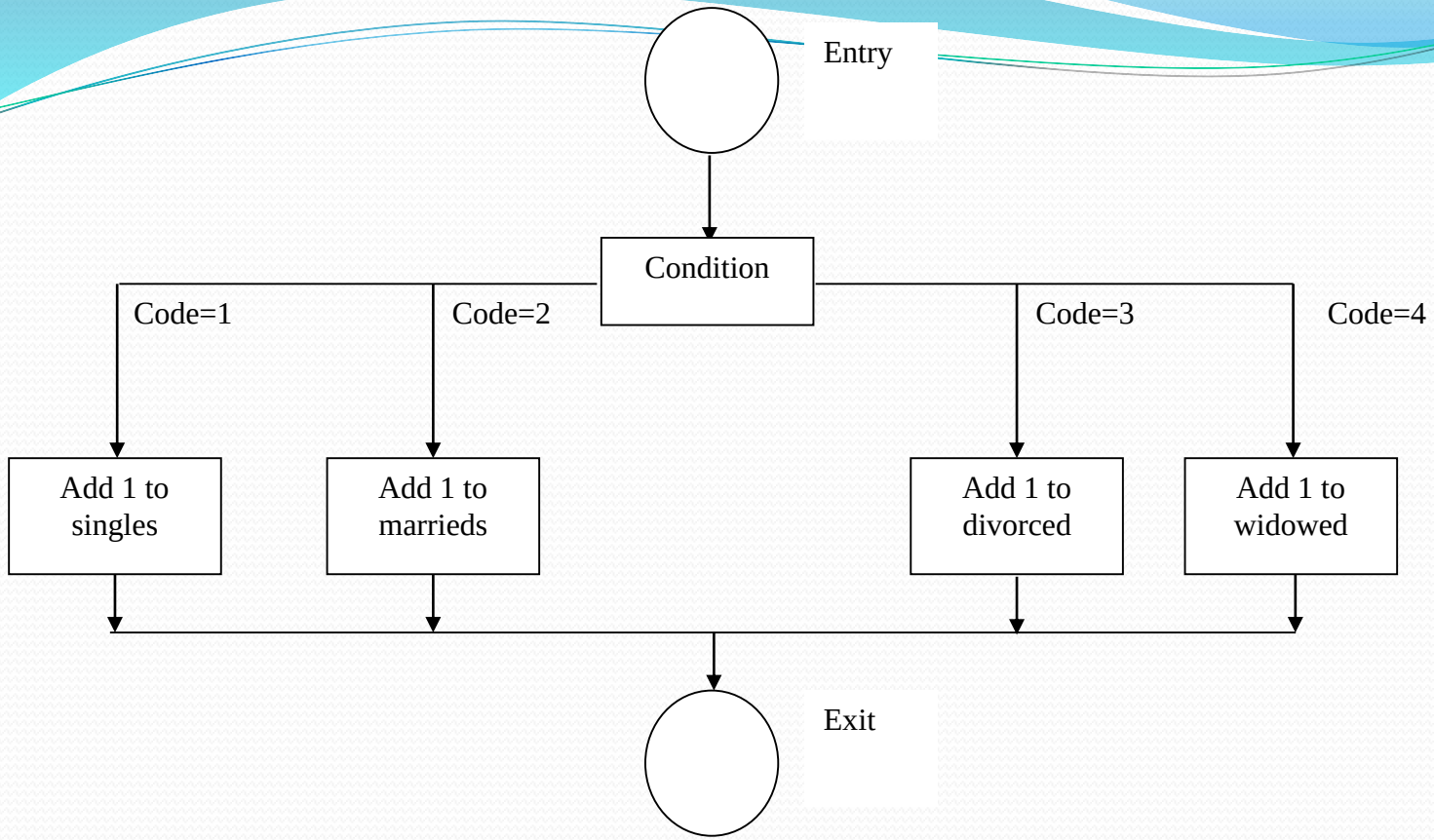




شکل (4د-1)



شکل (4ج-1)



شكل (3هـ-1)

شكل (4هـ-1)

نلاحظ هنا كيف تم إلغاء ELSE من تركيب الاختيار. أن الشكل (4ب-1)يمثل مخطط الاختيار في الحالة الأولى ,أما الشكل (4ج-1)فيمثل الحالة الثانية.

3-تركيب التكرار DO-WHILE

يسمح هذا التركيب بعمليات التكرار الدائري LOOP .أو بمعنى أخر أنه يسمح لمجموعه من الأيعازات أن يتكرر تنفيذها عدة مرات . ولأجل إيقاف التكرار الدائري يوضع شرط في هذا التركيب بحيث يتم اختباره قبل البدء بتنفيذ الدورة في كل مرة وينتهي التكرار الدائري عندما تكون نتيجة أحد الاختبارات للشرط سالبة إذ يتحول مجرى البرنامج إلى تنفيذ الايعازات التالية . مثال على ذلك :

```
K:=0;WHILE k< 1000 do being statements;
K:=k+1
End
```

في هذا المثال نجد أن تكرار التعابير التي تلي DOسيتوقف حالما تصبح (K=1000) ويمكن تمثيل هذا التراكيب كما هو في الشكل (4د-1)

أن تركيب التكرار الدائري يمثل في بعض لغات البرمجة بواسطة (PERFORM-UNTIL) أو بواسطة (REPEAT-UNTIL) .

و بصورة عامة فإن كل من هذه التراكيب يخضع إلى أسس موحدة و هي :أن تركيب التكرار يتألف من جزئين ، الجزء الأول وهو الشرط الذي يحدد إيقاف عملية التكرار ، أما الجزء الثاني فهو الهيكل الذي يراد تكراره.و قد يتكون هذا الهيكل من تعبير واحد أو عدة تعابير .

وفضلا عن التراكيب الثلاثة الأنفة الذكر فان هناك تركيبا أخر يمكن الاستفادة منه ويدعى CASE أو "الاختيار المتعدد" .ويعد هذا التركيب ملائما في حالة وجود عدد كبير (أكثر من اثنين) من البدائل التي يراد تنفيذ واحدة منها ، بعد اختيار الشرط الذي يحدد البديل الملائم لأي حالة كانت . وفي الإمكان الاستعاضة عن هذا التركيب بالاستعانة بمجموعة متداخلة من تراكيب IF-THEN-ELSE

أن التعبير المستخدم بلغة كوبال COBOL والذي يقوم مقام تركيب الاختيار المتعدد CASE هو

GO TO.....DEPENDING ON

مثال على ذلك :لنفرض أن حالة الزوجية للموظفين قد رمزت بشكل إعداد (1 ,2 ,3 ,4)والتي تعني بالتسلسل :أعزب ,متزوج , مطلق ,وأرمل كان المطلوب أعداد إحصائية عن الموظفين كل حسب حالته الزوجية , فبالمكان استخدام التعبير GO TO ..... DEPENDING ON لهذا الغرض كما يلي :

```
GO TO SINGLES, MARRIED, DIVORCED, WIDOWED
   DEPENDING ON CODE.
SINGLES.
    ADD 1 TO SINGLES-COUNT.
    GO TO  NEXT – PARAGRAPH
MARRIED.
   ADD 1 TO MARRIED-COUNT.
   GO TO NEXT –PARAGRAPH
.
.
.
.
```

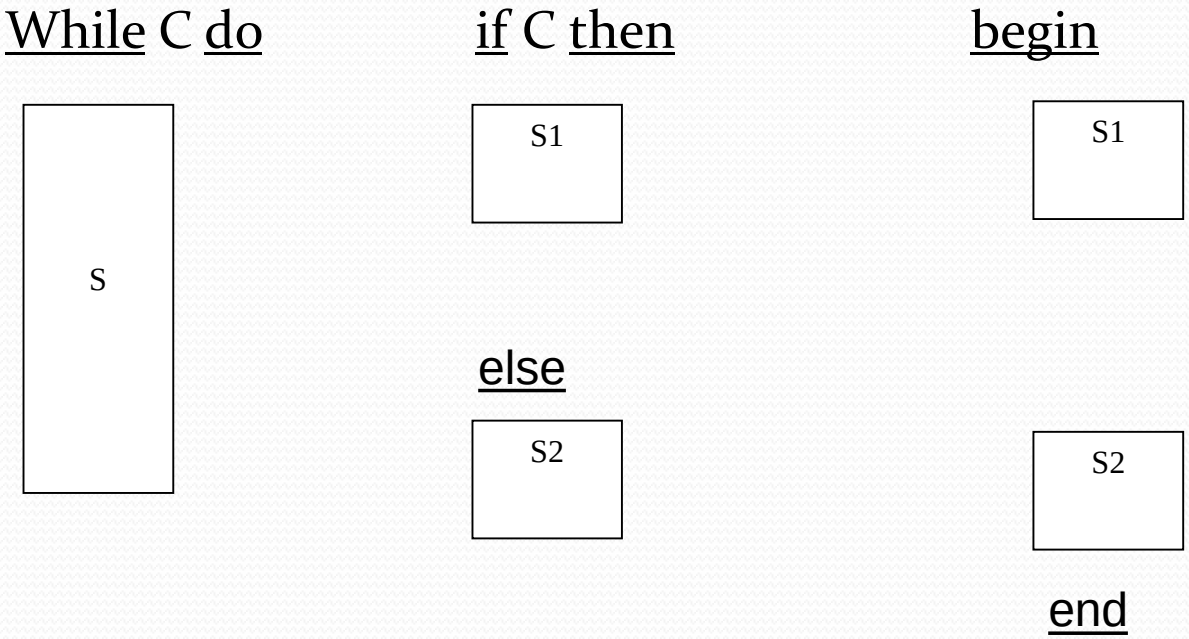
ويمكن تمثيل هذا التركيب كما في الشكل رقم (4هـ-1).

من ملاحظة مجموعة الرسوم التي تمثل التركيب الرئيسية السابقة نجد أنها جميعا تمتاز بوجود مدخل واحد ومخرج واحد. ونتيجة لذلك فأن نجد البرنامج المتكونة من هذا التراكيب المهيكلة مدخلا واحدا ومخرجا واحدا. وعليه فأن البرنامج سينفذ بصورة محكمة من أول تعبير فيه إلى أخر تعبير.

في الإمكان تمثيل كيفية الظهور الإشكال (4أ-1) ,(4ب-1),(4د-1) في قائمة الترميز كما في الإشكال (5أ-1),(5ب-1),

و (5جـ-1) على التوالي:

و(5ج-1) على التوالي



شكل (5أ-1) شكل (5ب-1) شكل (5ج-1)

أن استخدام التراكيب المهيكلة الأساسية في كتابة البرامج، يضمن لنا برامج تغني عن استخدام تعابير GO TO (اعتمادا على لغة البرمجة المستخدمة) في بنائها .ولهذا السبب فان البرمجة المهيكلة التي توصف في بعض الأحيان على أنها أسلوب البرمجة الذي يلجأ إلى استخدام تعبير GO TO اقل ما يمكن "GO TO LESS PROGRAMMING".

وعلى آية حال فان هذا مفهوم ضيق للموضوع فضلا عن أنه قد يصعب تطبيقه في بعض لغات البرمجة. إن النقطة الرئيسية هنا تتجلى في أن إقصاء تعبير GO TO حصل ,لأن بناء البرامج سيتم بالاستعانة بتراكيب البرمجة الرئيسة فقط .

أن استخدام التراكيب المهيكلة الرئيسة في بناء البرامج يتمخض عن نتيجة مثمرة أخرى، وهي أن مجرى سريان منطوق البرنامج سيكون من
أعلى إلى ادني (TOP-DOWN).أي أن تعابير البرنامج المحددة سوف يتم إنجازها بتسلسل الذي تظهر في قائمة البرنامج.

وبطبيعة الحال فأن هناك كتلا من الرموز سوف يتخطاها البرنامج في أثناء الانجاز.ففي حالة انجاز التركيب IF-THEN-ELSE ,فان البرنامج سيمر في إحدى الكتلتين (أما كتلة الرموز التي تلي THEN تاركا كتلة الرموز التي تلي ELSE ، أو يتخطى كتلة الرموز التي تليTHEN مارا بالكتلة التي تلي ELSE) . أما في حالة DO-WHILE فان كتلة الرموز المحصورة بينهما سوف تنجز لعدة مرات. لكن , في كافة الأحوال ,فان أي تفرع في البرنامج سيؤدي إلى نقطة تقع في موقع متقدم من قائمة البرنامج . أن اتجاه منطوق البرنامج نحو الأمام يسهل عملية قراءة وتتبع البرنامج .