(مفاهيم في البرمجة المهيكلة)

السيدة هيفاء جرجيس جولاخ
المركز القومي للحاسبات الالكترونية

1 ـ مقدمـــــــــة :ـ

ان المقالة تضع بين ايدي المبرمجين الاسلوب الاكفأ والابسط في كتابة برامج واضحة تسهل كتابتها ، قراءتها ، تصحيحها وبالتالي اختبارها واداؤها . والسؤال هنا ، هل يمكن تصميم برنامج يكون ترميزه وتصحيحه وتنفيذه من 5 الى 10 مرات اسرع من المعتاد ؟ هل يمكن كتابة برنامج والحصول على نتائج صحيحة بتنفيذه لاول مرة ؟ هل يمكن التقليل من ذاكرة ووقت تنفيذ البرنامج باستخدام افعال او اشكال بناء معينة ؟ .. اضافة الى هذا ، هل يمكان كتابة برامج يمكن الاعتماد عليها وصيانتها وزيادة مرونتها الى اقصى حد ممكن ؟

وسنحاول من خلال صفحات هذه المقالة ان نجيب على كل الاسئلة الواردة اعلاه : فلنحاول ان نبسط هذا الاسلوب بتعريفه اولا . ليس هناك تعريف ثابت للبرمجة المهيكلة لحد الان ، وعلى اية حال فمن الممكن التعبير عنها : على انها طريقة لتنظيم الافكار والبرامج للحصول على برامج سهلة وصحيحة وقابلة للتحوير . وعموما نستطيع القول بأنها "تبسيط لمسار التحكم للبرنامج" . ان نفقات صيانة البرامج والانظمة تزداد اهميتها بزيادة عدد البرامج العاملة وهي

تحتل اكثر من الميزانية السنوية وباستخدام اسلوب البرمجة المهيكلة يمكن ان تقل كلفـــة صيـــانة البرامـــج .

2 ـ ان مفتاح حل مشكلة في حقل البرمجة هو تحديد نقطة الابتداء ، اذ يتطلب البرنامج الناجح تعريفا للمشكلة وتحليلا لعناصرها . وتحليل المشكلة يدور حول ثلاث نقاط رئيسية :ـ

1.2 الناتج المطلوب الذي يقدمه حل المشكلة .
2.2 عناصر البيانات الداخلة في حل المشكلة .
3.2 خطوات المعالجة على مدخلات المشكلة لاستخراج مخرجاتها .

ومن الطبيعي أن اعراض المشكلة هي العناصر التي يمكن ملاحظتها ، الا أن تلك الاعراض لاتظهر دائما المشكلة ، لذلك فمن المستحيل القيام بتحليل دقيـق للحالة التي تثيرها اعراضها دون التعريف المسبق ، بطريقة ما ، للنتائج المطلوبة .

وبعد ان يكون المبرمج قد عرف وحدد متطلبات المعلومات يتخذ الخطوة التالية ليأخذ في اعتباره الاجراءات المنطقية التنفيذية لمعالجة بيانـــات المشكلة ، فالتصميم التكامل لمواصفات خطوات معالجة البيانات يعود على المبرمج بمنفعتين رئيستين :

* تعطي هذه العملية للمبرمج فرصة مراجعة منطقية في معالجة البيانات ومـــن الواضح ان المعالجة غير المنطقية ستقود ختما الـــى نتائج غـــير صحيحة حتى للبيانـــات الدقيقـــة .
* يمثل ناتج التصميم الدقيق لخطوات المعالجة مرشدا في اعداد قيود اللغة التـي يكتبهـــا المبرمـــج .

3 ـ تصميـــم البرنامـــج :

ان البرنامج الجيد هو الذي يؤدي الغرض الذي صمم لاجله ويجب ان يكون :
ـ سهل الكتابة (من قبـل المبرمـج)
ـ سهل القراءة والفهم (من قبل الاخرين)
ـ سهل الاختبار (فبواسطة تقسيم البرنامج الى روتينات Modules) نستطيع ان نفحص كل روتين على حدة مرة واحدة ـ بأعطاء بيانات للفحص

بينما البرنامج الذي يكتب بالطريقة الاعتيادية يتطلب الرجوع الى نفس الروتين عددا من المرات (لاجراء الاختبار)

ـ سهل التغيير (اضافة او حذف جزء معين من البرنامج) .

فالكلفة العالية للبرنامج ترجع الى زيادة تعقيده مما يؤدي الى مصاحبته لكثير من الاخطاء اذا اريد تطوير البرنامج ، مما يؤدي الى ضياع الوقت واعادة كتابة البرنامج . كما ان التعقيد في البرامج يسبب مشاكل اثناء صيانتها ، فعند تعديل برنامج معين نتيجة خطأ ما فأن تعقيداته تجعل من الصعب صيانته حتى لو تم ذلك من قبل كاتب البرنامج نفسه ، وهنا نقول بأنه من الضروري ان يكون البرنامج واضحا لان وضوح البرنامج يساعد على تطويره وصيانته .

ويقصد بالوضوح ، هو امكانية فهمه وسهولة قراءته حتى من قبل الاشخاص الذين يجهلون وظيفة البرنامج الاصلية . يمكن التقليل من تعقيد البرنامج اذا فكرنا بأزالة بعض الاشياء من البرنامج ، كالمسار المعقد والرموز الغامضة لبعض الروتينات التي تصف البرنامج اذ يجب ان تكون طبيعية وقريبة من الواقع (واقع عمل الروتين) وبدون غموض كالاستعاضة عن الرمز R ـ T الذي يمثل الروتين الذي يقرأ ملف الاستقطاع بالرمز Read-Trans كما ان زيادة ايضاح البرنامج تتم اذا قسمنا مخطط البرنامج الى مستويات levels توضح خطوات العمل وجعلنا سلسلة الايعازات والفقرات بصيغة او باسلوب قريب من طبيعة المشكلة .

اما المخرجات لمثل هذه البرامج فيجب ان تكون بعناوين واضحة ، ومرتبة بحيث تكون سهلة الاستعمال وحسب الحاجة وتفي بالغرض المطلوب ، اى انها لا تحتوي على تفاصيل اضافية لا نحتاجها للتنفيذ بصورة رئيسة او فعلية .

4 ـ بناء البرنامج :

ان اهم عناصر بناء البرنامج هي :

1.4 ـ توثيق تصميم البرنامج

2.4 ـ المخرجات المطلوبة

3.4 ـ مصادر المدخلات المتوفرة

ان التوصل الى طريقة جيدة لبناء البرنامج يكون في وضع خطة عامة قبل البدء بوضع خطة منطقية مفصلة وذلك يسهل عملية بناء البرنامج ويؤدي الى امكانية

تقسيمه الى اقسام صغيرة يسمى كل منها (Module) مما يوفر قاعدة جيدة للمنطق تسهل عملية البناء وهو بالتالي اساس لعملية التجزئة (Modularity) .

4.4 ـ يتلخص اسلوب البرمجة المهيكلة في ترتيب وكتابة البرامج بطريقة يجعلها سهلة الفهم والتعديل ، فمثلا عند تعديل جزء معين من البرنامج فان اوامر جديدة قد تضاف لان بعض الاوامر لاتنفذ وظيفة معينة او قد تضاف اجزاء جديدة للبرنامج .

5 ـ الخصائص الاساسية :

هناك ثلاث صفات مميزة يصمم ويبنى على اساسها البرنامج الجيد وهي :

1.5 ـ العمومية Cenerality

2.5 ـ الهرمية Hierarchy

3.5 ـ قابلية التجزئة Modularity

1.5 ـ فالعمومية تعني ان يكون البرنامج عاما يمكن تطبيقه لاى عدد من البرامج بسهولة تغيير أي جزء من اجزائه .

2.5 ـ والهرمية تعني اتباع اسلوب Top-Down في بناء البرنامج وتنفيذه وهي تعريف للوظائف الاساسية للبرنامج وتقسيمها الى وحدات وظيفية متسلسلة حتى الوصول الى أوطأ مستوى للبرنامج ، ويطلق على كل مستوى Level ابتداء من Level - O للمستوى الرئيس . ان استخدام المستويات ـ levels ـ يسهل عملية كتابة البرنامج (سنوضح ذلك بأستعمال لغة كوبول في هذه المقالة) .

قاعدة : يمكن تقسيم اي مجموعة (Set) من المعلومات الى مجموعات فرعية (Subsets) من أعلى مستوى بأستخدام اسلوب التقسيم التجزيئي وهذه الطريقة تؤدي الى الحل الامثل .

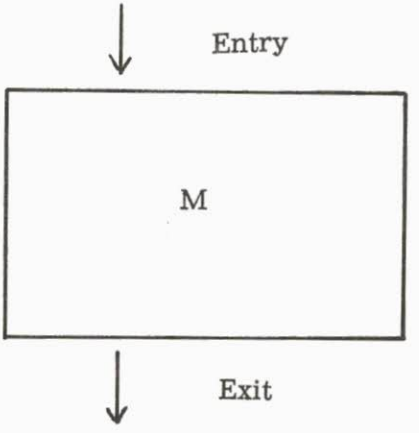
هنالك عدة اساليب للبرمجة المهيكلة منها طريقة Wernier وطريقة الاشكال الشجيرية (Tree) التي تعتمد اساسا على التمثيل الهرمي لمخطط البرنامج ، وسنوضح ذلك بعد تعريف الخاصية الثالثة .

3.5 ـ وقابلية التجزئة تعني تعريف وظائف Functions عامة تستخدم من عدة

أماكن في البرنامج . يطلق اسم Module على كل جزء يكون كوحدة اساسية في نظام التجزئة (Modularity) يمكن تصور البرنامج كمجموعة من Modules او روتينات ترتبط ببعضها منطقيا ويكون اسلوب التنفيذ لها من اليسار الى اليمين ومن الاعلى الى الاسفل .

1.3.5 هناك مجموعة قواعد تحكم الـ Module منها :

أنه يحتوي على مدخل ومخرج واحد ولايحتوي على تفرعات الى اجزاء اخرى داخل البرنامج .

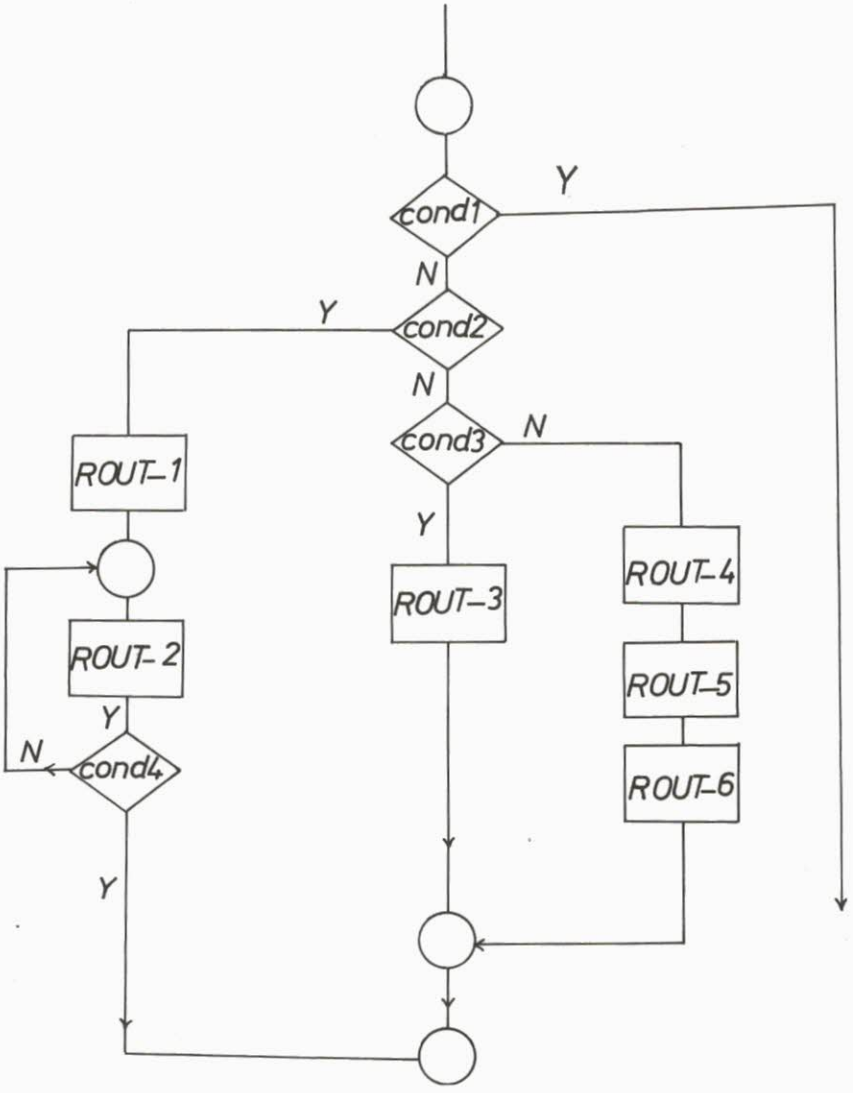


كما ان الـ (Module) الواحد يمكن ان يتكون من جملة او ايعاز واحد او مجموعة ايعازات . في شكل ـ 2 ـ نلاحظ كيفية تقسيم المخطط الانسيابي الاعتيادي (شكل ـ 1 ـ) الى مجموعة من Modules وفي شكل ـ 3 ـ كيفية وضع المخطط بعد التقسيم وحسب قاعدة الهرمية واسلوب الاشكال الشجيرية .

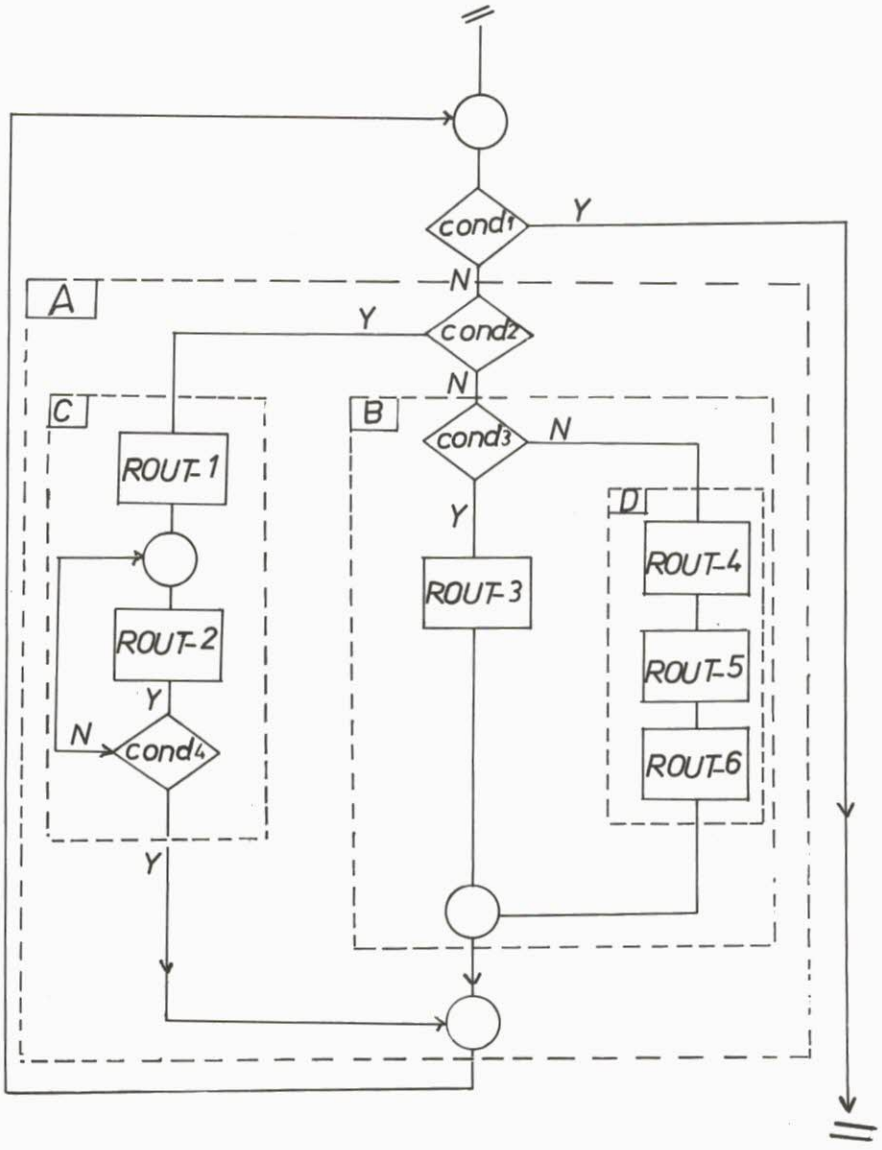
2.3.5 يجب ان يكون حجم الـ (Module) مناسبا او محددا حتى يمكن التحكم فيه ، والمقصود بالحجم هو عدد الايعازات المكونة له ، فقد تحدد 50 أيعازا لتمثل ورقة طباعة واحدة من جهاز الطبع (Lin Printer) وهنا تقل الحاجة الى قلب الصفحة او الاحتفاظ بأكثر من ورقة للروتين (Module) الواحد .

3.3.5 بأستطاعة اى Module استدعاء أى Module اخر ولكن الاخير يعود الى الـ (Module) المستدعي الاول ، نلاحظ هذه الصفة في شكل ـ 4 ـ

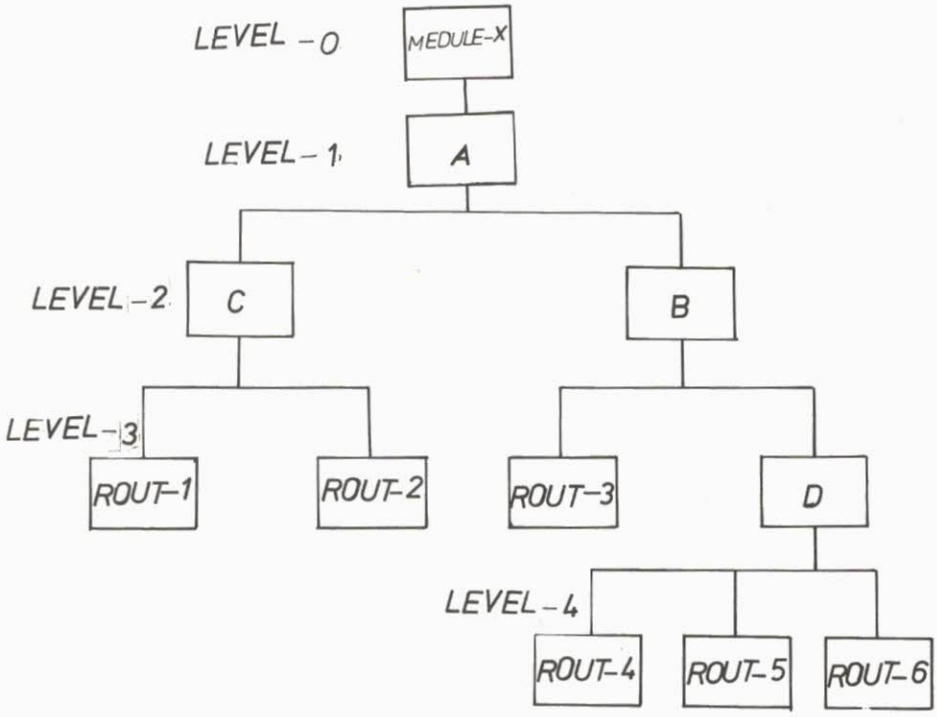
شکل — 1 —


cond1
Y
N
cond2
Y
N
cond3
N
Y
ROUT–1
ROUT–2
Y
N
cond4
Y
ROUT–3
ROUT–4
ROUT–5
ROUT–6

شکل — 2 —


cond1
Y
N
A
Y
cond2
N
C
B
ROUT-1
cond3
N
Y
D
ROUT-4
ROUT-3
ROUT-2
ROUT-5
Y
N
cond4
ROUT-6
Y

شكل — 3 —



6 ـ الهياكـــل الاســاســـية : Basic Structures

ان أية وظيفة في اى برنامج يمكن ان تنفذ بأستخدام واحـــدة من الهياكـــل الاســاســـية الثلاثـــة التاليـــة :

1·6 التتابـــــــع Sequential

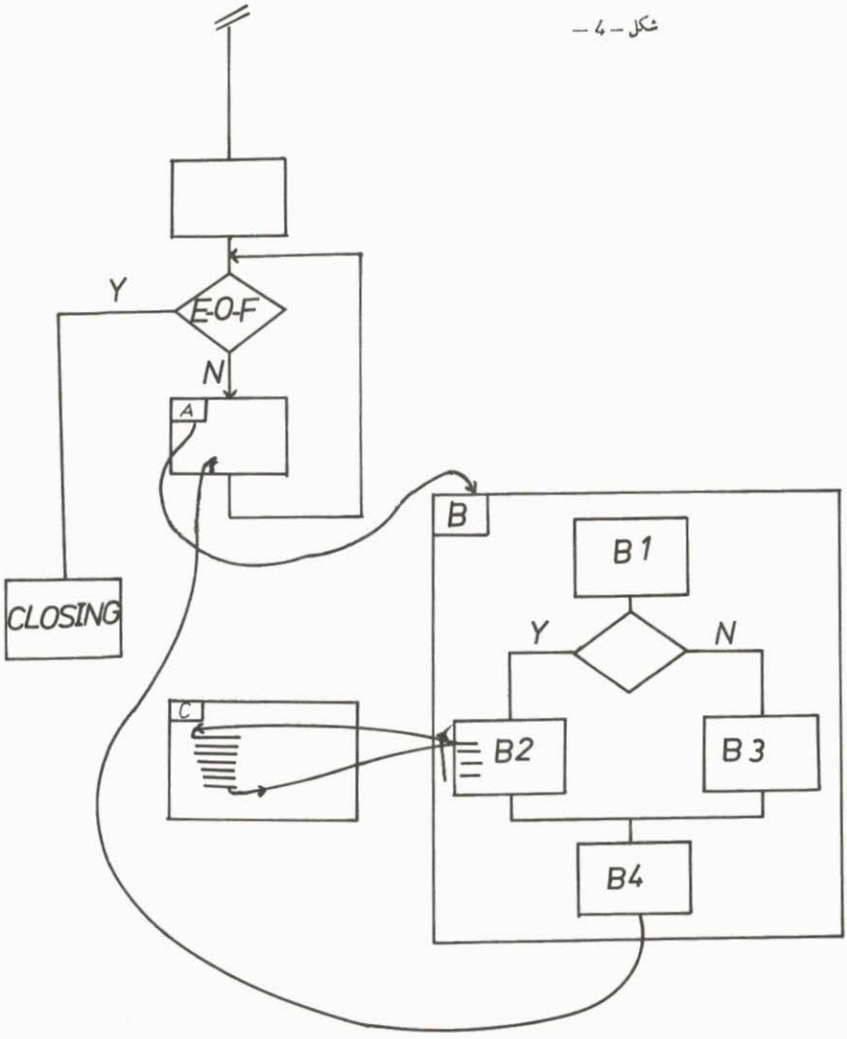
2·6 البدائـــــــل Alternatives

3·6 التكـــــــرار Repetitives

حيـــث أن :

1·6 يعني التسلسل المنطقي لتنفيذ الروتينات (Modules) او الايعازات في الروتين الواحد اى يتم تنفيذ الروتين او الايعاز A ثـم B

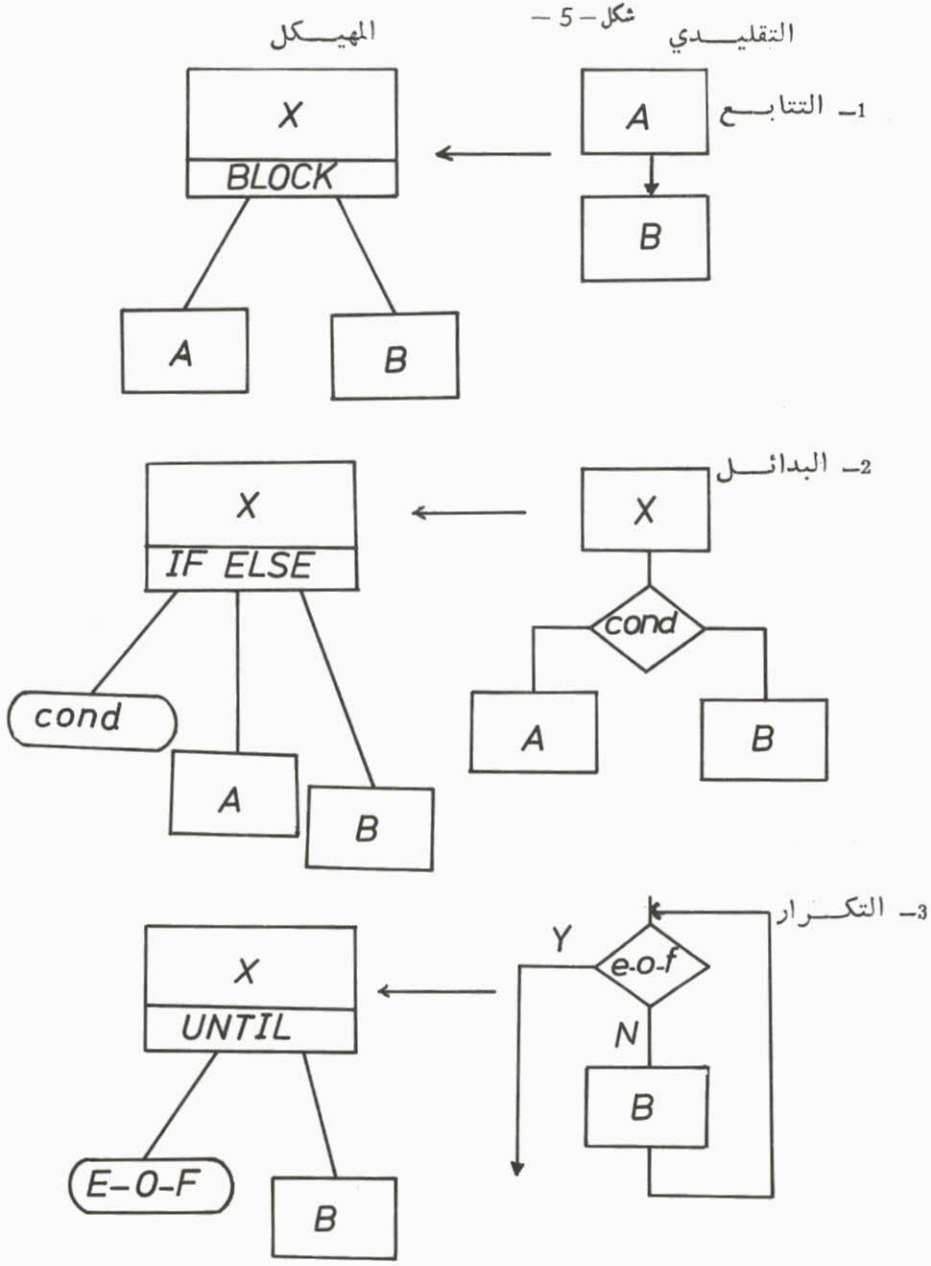
شكل – 4 –



2.6 هو ان يتحدد مسار التنفيذ على ضوء استفسار او شرط معين يؤدي الى تنفيذ A او B ، او تنفيذ A فقط وعكسه العودة الى الروتين المرسل للمعلومات .

و 3.6 هو تكرار تنفيذ عملية معينة (A) عددا من المرات حتى يتحقق شرط معين يؤدي الى التوقف عن تنفيذ A لينفذ B . وبأستخدام اسلوب الاشكال

شكل – 5 –


التقليـــدي
المهيـــكل
1– التتابـــع
A
B
X
BLOCK
A
B
2– البدائـــل
X
cond
A
B
X
IF ELSE
cond
A
B
3– التكـــرار
Y
e-o-f
N
B
X
UNTIL
E-O-F
B

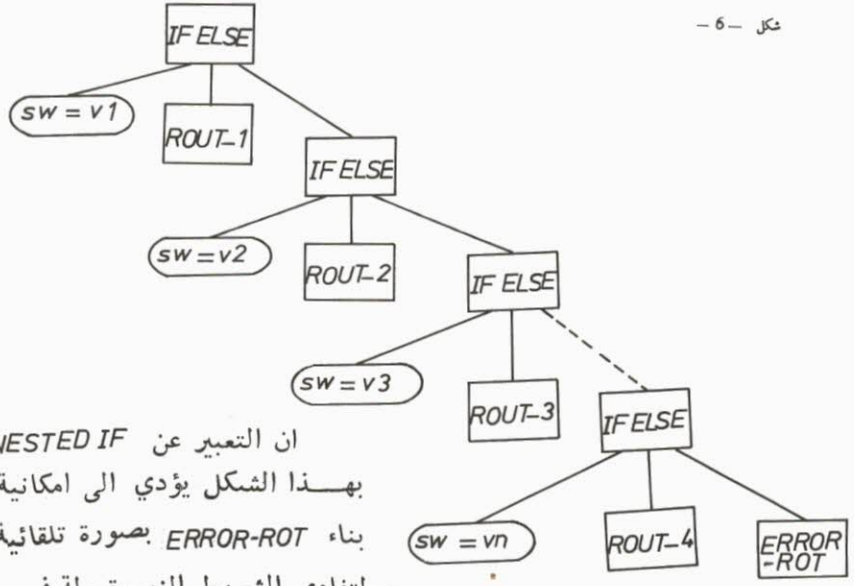
الشجيرية في مخطط البرنامج نستعمل كلمة (Block) للعملية المتسلسلة (1.6) ، و if-Eles او if (2.6) و Until لعمليات التكرار (3.6) . كما في شكل –5– .

كما ويمكن التعبير عن مجموعة البدائل (في حالة وجود عدد من الشروط يلزم تحقيقها عدد من الاوامر بـ Switch وفي البرمجة بلغة Cobol بـ Nested if كما في شـــكل – 6 –

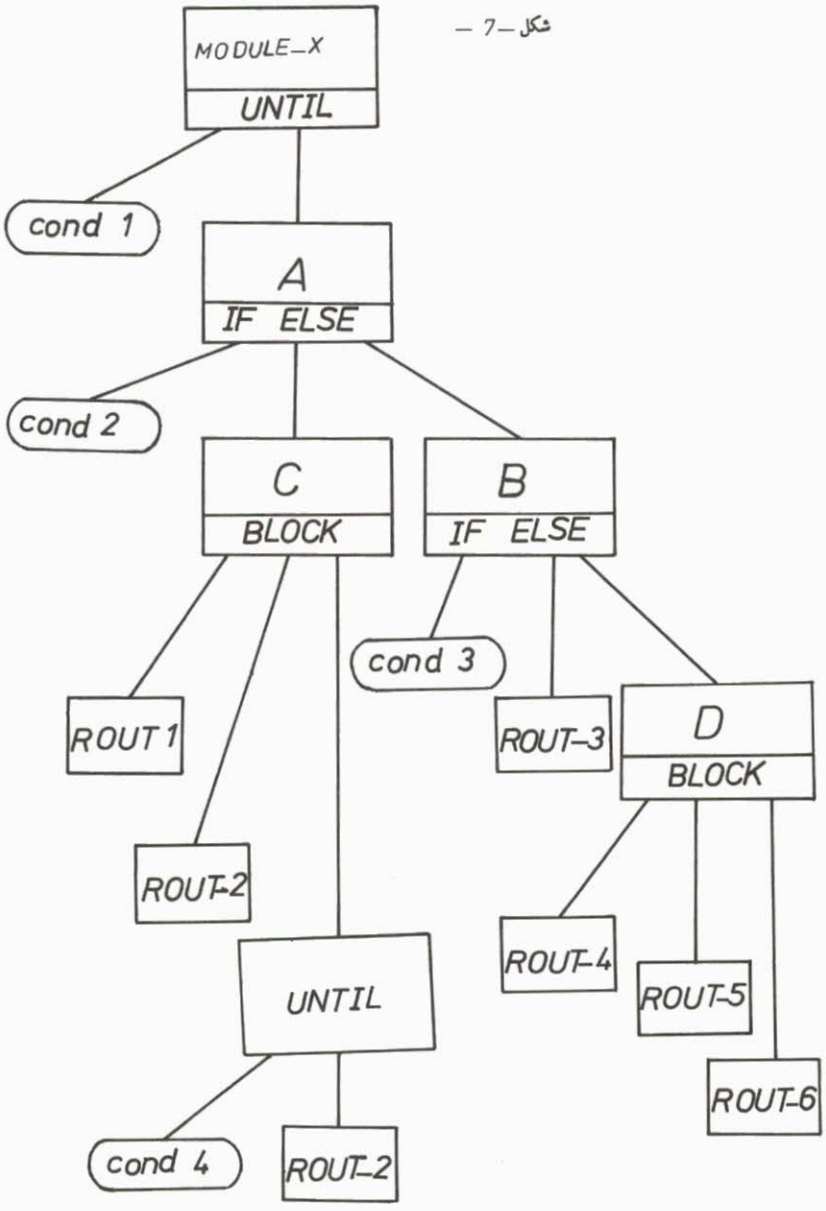
هذه الاشكال يمكن ضمها مع بعضها لتشكل برنامجا سهل المتابعة من أعلى الى اسفل او من نقطة الابتداء الى نقطة النهاية .

في شكل – 7 – استخدمنا الهياكل الاساسية لتحويل المخطط الهرمي المقسم الى Modules (شكل – 3 –) الى الشكل الابسط لكتابة البرنامج بالطريقة المهيكلة .

شكل –6–



ان التعبير عن NESTED IF بهذا الشكل يؤدي الى امكانية بناء ERROR-ROT بصورة تلقائية لتفادي الشروط الغير مقبولة في البرنامج .


MODULE_X
UNTIL
cond 1
A
IF ELSE
cond 2
C
BLOCK
B
IF ELSE
cond 3
ROUT 1
ROUT-2
UNTIL
cond 4
ROUT-2
ROUT-3
D
BLOCK
ROUT-4
ROUT-5
ROUT-6

شکل – 7 –

٧ ـ كتابـــة البرامـــج المهيكلـــة :

يمكن كتابة الهياكل الاساسية بأية لغة من لغات البرمجة وعليه يمكن تتبعها بواسطة اى شخص ، ومعظم لغـــات البرمجة ذات المستوى العالي فيها خصائص تناسب هذه الهياكل ، فمثلا لغة PL/I فيها الامـران If - Then - Else و Do اللذان يقومان بوظيفتى البدائل والتكرار على التوالي ، كما ان لغـــــة Fortran تحتوى على If المبسطة و Do . اما لغة Cobol فهـي تحتوى على If - Then - Else والامر Perform وهما يقومان بوظيفة تكرار بعض الاوامر والبدائل على التوالي وعموما فان معظم اللغات ذات المستوى العالـي تحتوى على ما يقابل او ما يقوم بهذه الوظائف . ان زيادة ايضاح البرنامج تتـم بأدخالنا اشياء معينة اليه مثل عناوين توضح نفسها (كما ذكرنا) وملاحظات جيدة للقاريء تبين غرض البرنامج . وترك مسافات بين الجمل او البدء من امكنة معينة في الكتابة بحيث يمكن القارىء من متابعة البرنامج بسهولة ، كما ان ذلك يؤدي الـى زيادة الوضوح في البرنامج وبذلك يصبح تسلسل الاوامـــر في البرنامج اســـهل وبالتالي سهولة تصحيح اى خطأ بمجرد القراءة العادية .

ان الاتجاه نحو كتابة البرامج المهيكلة يدعونا الى التقليل مـن استخدام الـ Go To ، وذلك لصعوبة تتبعه في البرنامج وسهولة تداخله بـــين الروتينـات بالاضافة الى ان البرنامج تسهل قراءته وتصحيحه عندما لايحتوي علـى اي Go To اطـــــــلاقا .

وبالاستعاضة عن الامر Go To نستخدم الامر Perform لروتين معين او Perform Until الذي يؤدي الى تنفيذ الروتين عددامن المرات حتى يتحقق شرط معين . اذن فحسب القواعد المذكورة سيصبح من السهل ترميز المخطـط في شـــكل ـ ٧ ـ كمـــا يلـــي :ـ

```
Module — X.
        Perform a until cond 1.
A.
        If Cond 2.
        Then perform C.
        Eles perform B.
```

C.

Perform Rout 1.

Perform Rout 2.

Perform Rout 2 Until Cond 4.

B.

If Cond 3.

Then Perform Rout 3.

Else Perform D.

D.

Perform Rout 4.

Perform Rout 5.

Perform Rout 6.

8 ـ الخلاصـــــــــــة :

1.8 هناك هيكلة في نص البرنامج ، في نمط الاحداث عند وقوعها وفي طريقة خلقها وابداعها ، وفي الوصف لمجمل النتائج ، كذلك في تنفيذ البيانات وفي الاغراض التي تخدم ذلك .

وان من المبادىء والاهداف الرئيسة هو الاحتفاظ بالسيطرة على الامور المعقدة بالنسبة الى كافة هذه المجالات بتوطيد اواصر الثقه بأستخلاص النتائج للخطوات القصيرة بمنتهى الوضوح وبالاستقلالية الممكنة .

2.8 تتضح اهمية استخدام اسلوب البرمجة المهيكلة بمقارنتها بمخططات انسياب المنطق الاعتيادي (Flow Charts) حيث تتركز الاهمية على ترتيب الاحداث بشكل يضمن ضبط الانسياب من عنصر الى اخر يحمل مغزى معينا . فأستخدام الهيكلة في تصميم خطوات المعالجة هو تصميم قياسي سهل رسمه وهي في نفس الوقت سهلة المراجعة والصيانة . كما تتميز بالعرض المحكم للحالة قيد البحث

وهي سهلة البناء والتعديل والقراءة وقد ساهمت في معالجة دراسة وتصميم انظمة كانت معالجتها تتم بصعوبة بأساليب اخرى وهي سهلة الفهم في التطبيقات العملية والتجارية كما نها سهل للفهم من مخططات انسياب المنطق . اضافة الى ان توثيق التطبيقات المشتملة على التفاعل المعقد للمتغيرات اكثر وضوحا من مخططات انسياب المنطق .

3.8 ليس اسلوب البرمجة المهيكلة دواء عاما فقد لايكون الحل الامثل لكل مشاكل البرمجة وعموما فأن قسما من الاساليب التي ذكرناها مثل اسلوب Top - Down والرموز او العناوين الواضحة في البرنامج والهياكل الاساسية يساعد في التوصل الى الحل الامثل .

4.8 وخلاصة القول : ان اسلوب البرمجة المهيكلة ليس الدواء الشافي وانما هو تحسين او تطوير نتيجة جهود عملية وخبرات من استخدمها من المبرمجين .

9. ويستفاد من البرنامج الحقيقي التالي كنموذج لتوضيح مبادىء افكار البرمجة المهيكلة .

يقوم البرنامج بطبع تقارير شهرية بأرصدة المقترضين المتوفين في تاريخ الوفاة .

1.9 أن ملف الادخال Death-Fl متسلسل (Sorted) اعتمادا على .

أ ـ رقم الفرع Iend-Br-Cod

بـ رقم القرض Lon-No

2.9 يكتب العنوان (Header) في بداية كل صفحة جديدة .

ابدأ بتسلسل صفحات جديدة عندما يتبدل رقم الفرع .

كون جدولا يحتوي على ارقام الفروع واسمائها فعند مقارنة الرقم الموجود في الجدول مع الرقم الجديد ، أطبع أسم الفرع الذي يقابل الرقم في التقرير مع ملاحظة المكان المناسب له .

الجدول :

، الكرخ	01
، الرصافة	02
، الكرادة الشرقية	03
الكاظمية	04

التقرير المطلوب تكوينه

المصرف العقاري ـ قسم الحاسبة الالكترونية
نظام الاستقطاع الشهري
اسم الفرع ———— صفحة zz 9

تقرير بأرصدة المقترضين المتوفين التاريخ 99/ z9/ z 9

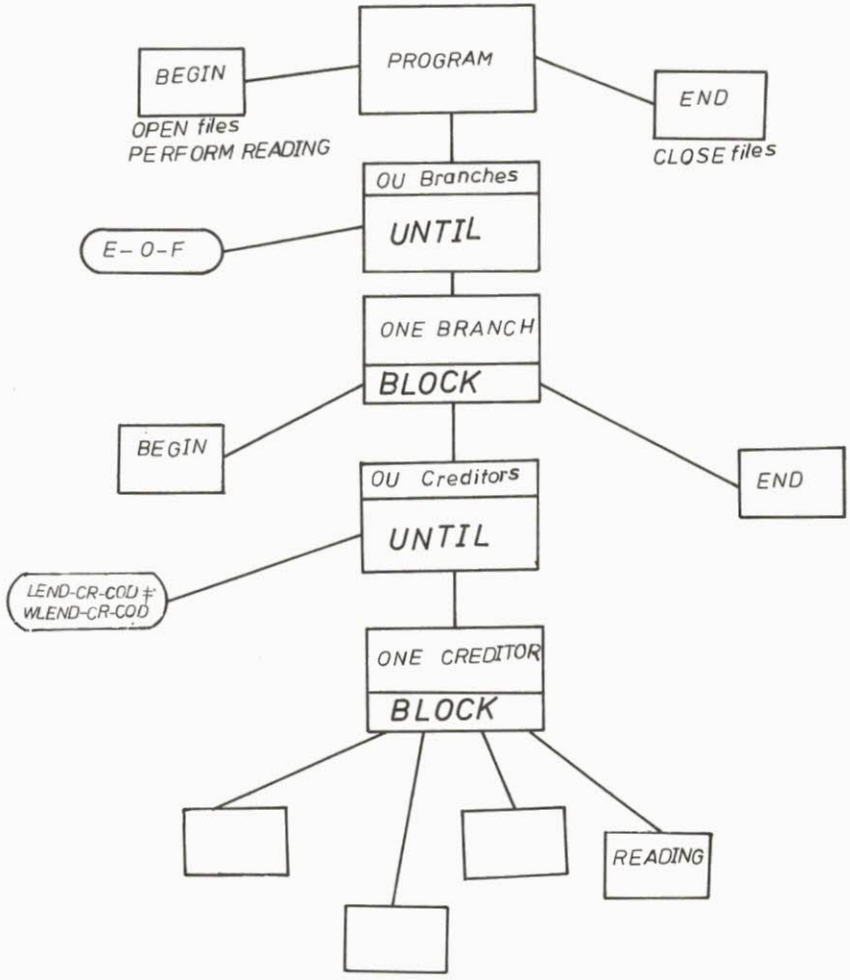
رقم القرض	اسم المقترض	تاريخ الوفاة	رصيد القرض المتبقي
999999999	X———X	99/ z9/ z9	zzz 9/ 999

اما قيد الادخال فهو كما يلي :

Lended branch	Loan Number	Creditor	Date of Death			Balance of
Code	9(9)	Name				Remaining Loan
(99)		X(30)	Year 99	Month 99	Day 99	9(4)V99

لنلاحظ المخطط الهرمي في شكل ـ 8 ـ
والبرنامج في الملحق ـ 9 ـ .

شکل - 8 -


PROGRAM
BEGIN
OPEN files
PERFORM READING
END
CLOSE files
OU Branches
UNTIL
E-O-F
ONE BRANCH
BLOCK
BEGIN
END
OU Creditors
UNTIL
LEND-CR-COD ≠
WLEND-CR-COD
ONE CREDITOR
BLOCK
READING

```
00098       PROCEDURE DIVISION.
00099       START.
00100                   PERFORM   BEGIN-PROG.
00101                   PERFORM   ONE-BRANCH UNTIL E-O-F.
00102                   PERFORM   END-PROG.
00103       ONE-BRANCH.

  104                   PERFORM  BEGIN-BR.
  105                   PERFORM ONE-CREDIT UNTIL
  106                   LEND-BR-COD NOT EQUAL W-LEND-BR-COD.
  107                   PERFORM  END-BR.
  108       BEGIN-PROG.

   09               OPEN INPUT DEATH-FL OUTPUT OUT-REP.

   10                   PERFORM   READING.
00111                   ACCEPT  WDATE  FROM  TIM.
00112       BEGIN-BR.

00113               MOVE LEND-BR-COD  TO W-LEND-BR-COD.
00114                   PERFORM  TOP1.
00115               MOVE  BR-NAME (LEND-BR-COD) TO P-BR-NAME.


00116                   PERFORM   TOP2.
00117       ONE-CREDIT.

00118               MOVE CRDTR-NAM       TO   P-CREDITOR-NAME.
00119               MOVE  LON-NO   TO  P-LON-NO.
00120               MOVE   YR  TO   PYR.
00121               MOVE   MON TO   PMON.
00122               MOVE   DAY TO   PDAY.
00123               MOVE   REMAIN-LON-BAL-D  TO  P-REMAIN-LON-BAL-D.
00124               MOVE   REMAIN-LON-BAL-F TO P-REMAIN-LON-BAL-F.
00125               WRITE  P-REC  FROM  DETL1 AFTER 2.
00126               MOVE   SPACES  TO  DETL1.
00127               ADD   1  TO  COUNTER.
00128               IF  COUNTER  > 25   MOVE  ZERO  TO  COUNTER
00129                   PERFORM   TOP1
00130                   PERFORM   TOP2.
00131                   PERFORM   READING.
00132       O-C-END.

00133       EXIT.
00134       END-BR.

00135               MOVE  ZERO   TO  W-PAGE.
00136               MOVE  ZERO   TO  COUNTER.
00137       READING.

00138               READ DEATH-FL AT END MOVE 1 TO END-OF-FI.

00139       TOP1.

00140               MOVE  SPACES TO  P-REC.
00141               WRITE  P-REC   FROM   HED1   AFTER  ADVANCING TOP
00142               MOVE  SPACES  TO  P-REC.
00143               WRITE  P-REC  FROM   HED2   AFTER 2.
00144       TOP2.

00145               ADD  1 TO W-PAGE  MOVE W-PAGE  TO  P-PAGE.
00146               WRITE  P-REC   FROM  HED3  AFTER 2.
00147               MOVE  SPACES  TO  P-REC.
00148               MOVE   W-DD TO P-DD.
```

```
00149                   MOVE    W-YR TO P-YR.
00150               MOVE    W-MM TO P-MM.
00151               WRITE P-REC   FROM    HED4   AFTER 2.
00152               MOVE  SPACES    TO    P-REC.
00153               WRITE   P-REC    FROM    HED5   AFTER 1.
00154               WRITE   P-REC    FROM    HED7   AFTER 1.
00155       END-PROG.

00156           CLOSE DEATH-FL OUT-REP.
00157               STOP RUN.
```

المصـــــادر

1 — Jean. D.W., "Logical Construction of Programs", Published Under the Auspices of IBC - I 1974.

2 — Andereqs . S.P., "Structured Cobol", 1977.

3 — "Softwear World Series", Volume 8 Number 3, Published Quarterly 1977.

4 — Infotech International Limited, "Structured Programming", England 1976.

5 — Central Agency for Public Mobilisation and Statistics. Electronic Data Processing", Published Bimonthly, Cairo, A.R.E. Volume 11 June 1976.

احمد مطاوع ، « اربعة مداخل للحاسبات الالكترونية » ، مطابع دار الشعب ، عمـان ـ الاردن ، 1978 .