بسم الله الرحمن الرحيم

مقرر هندسة البرمجيات
١٤٩٤

الوحدة الرابعة

تصميم البرمجيات
Software Design

إعداد: م. هناء قشطة

الفصل الدراسي الأول

٢٠٢١م-٢٠٢٢م

أهداف اللقاء

- أساسيات تصميم البرمجيات.
- التصميم على أساس انسياب البيانات.
- التصميم الكينوني.
- طرق التصميم المستندة إلى البيانات.
- التصميم الواقع.

٢ . أساسيات تصميم البرمجيات

- تصميم البرمجيات هو عملية ابداع وتطوير واختيار الحلول المناسبة وتقديم وصف دقيق للبرامج من أجل تحقيق متطلبات الجهة المستفيدة ضمن حدود التقنيات والإمكانيات المسموح بها.
- يمكن اعتباره انه وصف تفصيلي لطريقة بناء البرمجيات يوضح المفاهيم الاساسية ويساعد على اختيار طريقة الحل المناسبة.
- عبارة عن ادراك وتخطيط أو تطوير أو بناء خطة العمل .
- تصميم البرمجيات عبارة عن تخطيط يبين الخصائص الرئيسية للبرنامج المقترح.

٢ . أساسيات تصميم البرمجيات

هو تطوير أو بناء خطة عمل تأخذ بعين الاعتبار متطلبات الجهة المستفيدة، لينتج عن ذلك تخطيط يبين خصائص البرمجية المطلوبة من نواحٍ مختلفة.

- أفضل تصميم للبرمجيات هو الذي يعتمد الخيارات التي:
  - ✓ تحقق متطلبات و اهداف الجهة المستفيدة.
  - ✓ يوازن بين الإمكانيات المتوفرة، و القيود الموضوعة (مثل: التقنيات المتوفرة، طرق التخزين، مقدار الزمن، الإمكانيات الاقتصادية).

٢ . أساسيات تصميم البرمجيات

✓ **مواصفات جودة النظام:**

- كفاءة النظام.
- إمكانية الاستخدام المتكرر للبرمجيات (إمكانية إعادة استخدام العناصر البرمجية المكونة للنظام في أكثر من موقع و في أنظمة أخرى).
- قابلية البرمجيات للتعديل و التطوير.
- سهولة الاستخدام و التعلم.
- سهولة الصيانة.

٢ . أساسيات تصميم البرمجيات

- من الجدير بالذكر أن الجهة المستفيدة قد لا تكون البشر أنفسهم، و إنما قد يتطلب استخدام برمجيات من قبل برمجيات أخرى أو حتى مكونات مادية.
- يستلزم من المصمم عند تصميم البرمجيات الجيدة معرفة عدة نقاط:
  - أهداف النظام البرمجي المطلوب تصميمه.
  - وظيفة الجهة المستفيدة.
  - التغيرات المستقبلية المحتملة.
  - أجهزة الحاسوب المتوفرة.
  - تكلفة التصميم من حيث الأموال، الوقت، الكفاءة، الصيانة، وسهولة الاستخدام.

٢. أساسيات تصميم البرمجيات

- مخرجات مرحلة التصميم هي مدخلات المرحلة التالية، و تتضمن **المخرجات**:
  - تصميمًا هيكليًا يبين العلاقات بين أجزاء البرمجيات، بما في ذلك البيانات الداخلة و الخارجة لكل جزء من أجزاء البرمجيات.
  - مواصفات كل جزء من أجزاء البرمجيات المراد بناؤها.
  - تصميم للبيانات الجديدة التي يمكن أن يستفيد منها النظام البرمجي.

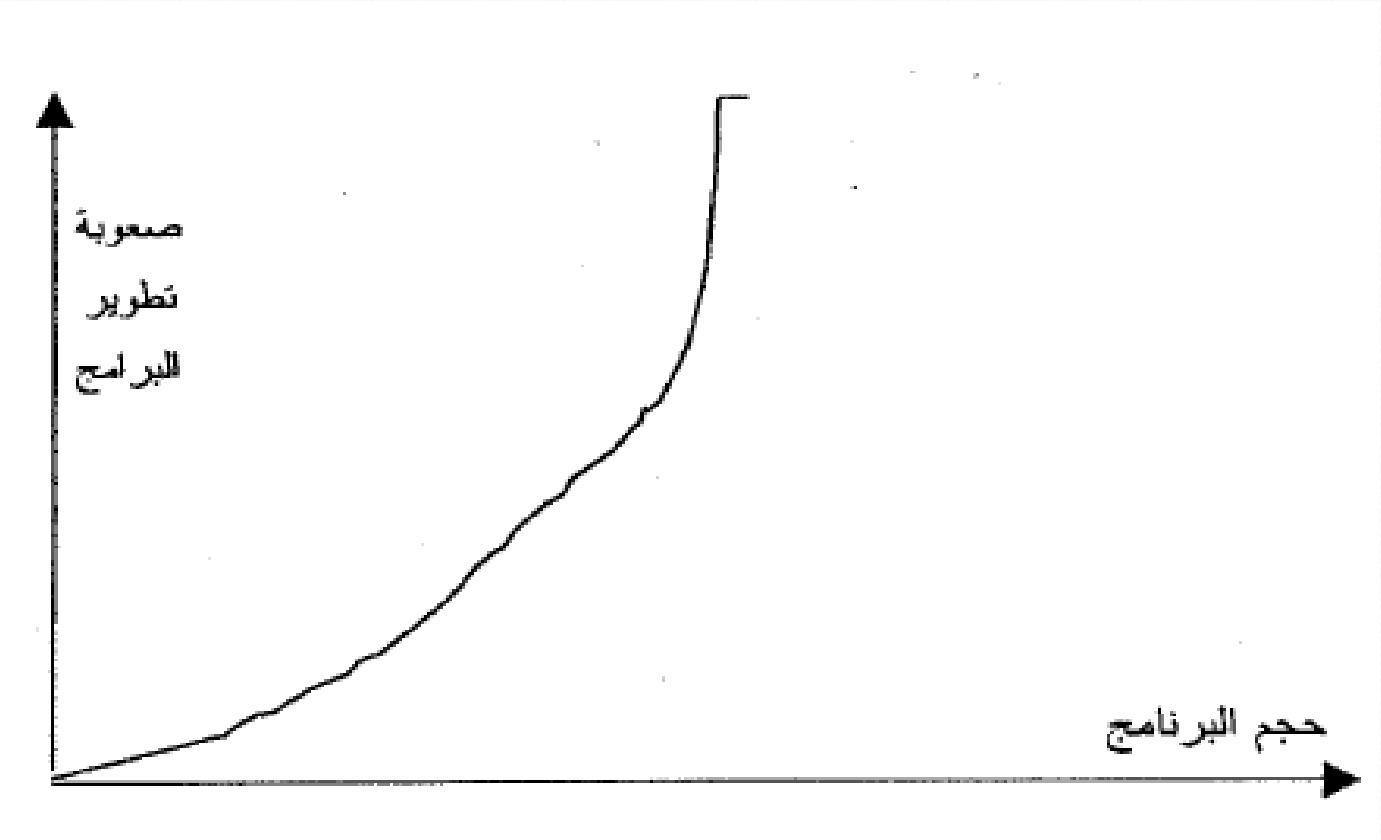
٢ . أساسيات تصميم البرمجيات

العوامل التي تؤثر في عملية إنتاج البرمجيات:

❖ **حجم البرنامج و صعوبة التطوير:**

- حجم البرنامج يعتبر من العوامل التي تؤثر على عناصر إنتاج البرمجية، فكلما زاد حجم البرنامج، فإن صعوبة فهم البرنامج، و تصحيح أخطائه، و تعديله، تتناسب أسيًا مع الحجم (ص ٩٥ الشكل ١).
- بالتالي كلما كان حجم البرنامج كبيرًا كانت الصعوبات أكثر، و يمكن تقليل الصعوبات من خلال تقسيم المسألة إلى مقاطع يتم إيجاد الحلول لكل مقطع بصورة منفردة و منفصلة، فكلما كانت المقاطع المسألة مستقلة عن بعضها البعض، نستطيع تذليل الصعوبات.
- إن تداخل مهام البرنامج أو المقاطع يؤدي إلى صعوبة تعديل مهمة أو برنامج دون تعديل المهمة الثانية المتربطة بها.

٢ . أساسيات تصميم البرمجيات

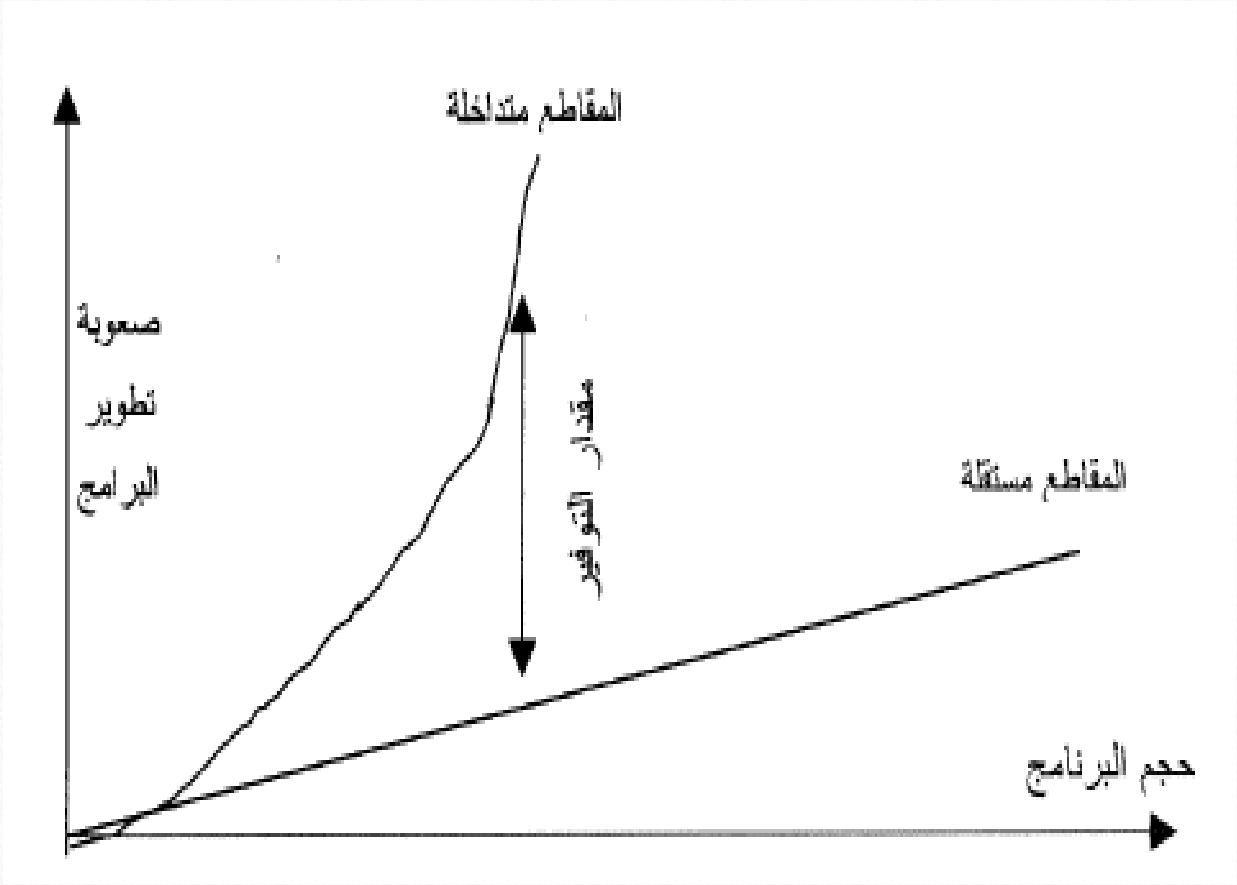


الشكل (١): العلاقة بين حجم البرنامج وصعوبة التطوير

٢ . أساسيات تصميم البرمجيات

- تقسيم البرنامج إلى مقاطع مستقلة يساعد في أداء مجموعة من المهام، منها:
  - ✓ سهولة فهم جمل المقطع البرمجي و تعديله.
  - ✓ سهولة اختباره و تصحيحه.
  - ✓ سرعة تطوير البرمجيات و بساطتها.
  - ✓ إمكانية إعادة الاستخدام للمقاطع.
  - ✓ زيادة الإنتاجية و رفع الجودة.
- تستمر عملية التقسيم حتى نصل إلى مستوى تكون فيه المقاطع صغيرة يسهل بناؤها و تتبعها و اختبارها.
- إذا كانت المقاطع مستقلة عن بعضها البعض، فإن الصعوبات تتناسب طرديًا مع حجم البرنامج (ص ٩٦ الشكل ٢).

٢ . أساسيات تصميم البرمجيات

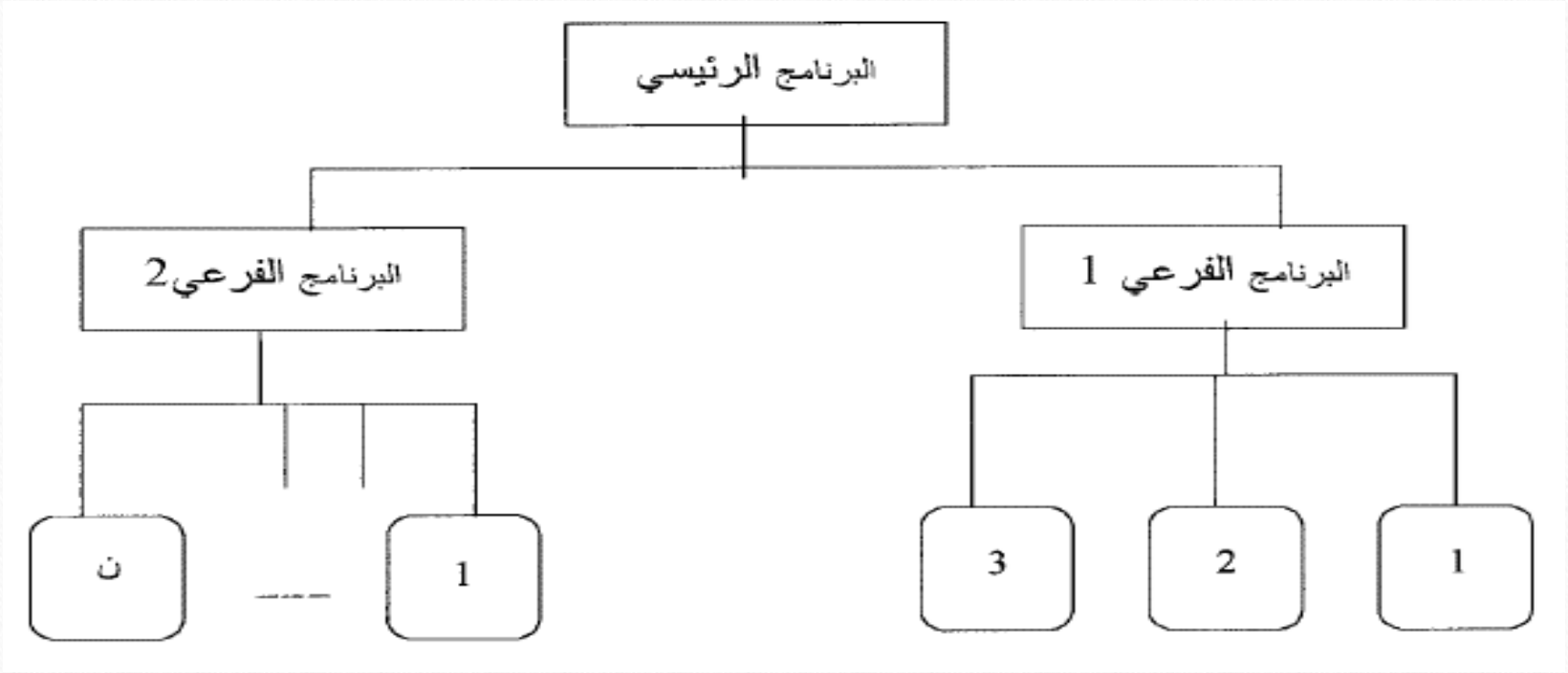


الشكل (٢): أهمية استقلالية مقاطع البرنامج

٢ . أساسيات تصميم البرمجيات

- إن المخططات هي الوسيلة المرئية التي تسهل تتبع تدفق البيانات، و تحديد مساراتها التي تبين العلاقات بين مقاطع البرنامج الواحد.
- تحتاج استقلالية البرامج إلى تصميم هيكلي، من أجل ربط الأجزاء مع بعضها البعض، بحيث يوضع البرنامج الرئيسي في رأس الشكل الهرمي، و نتسلسل أكثر في التقسيمات و البرامج الفرعية كلما انتقلنا إلى المستويات الأدنى حتى نصل إلى مستوى يحوي المقاطع الصغيرة (ص ٩٦ الشكل ٣).
- من محاسن التصميم الهرمي أو البرمجة الهرمية، أننا نعرض العلاقات بين المقاطع المختلفة على مستوى معين، أو على المستويات كافة.

٢ . أساسيات تصميم البرمجيات



الشكل (٣): مخطط هيكلي

٢ . أساسيات تصميم البرمجيات

العوامل التي تؤثر في عملية إنتاج البرمجيات:

❖ خاصية التماسك Cohesion:

هي قابلية المقطع البرمجي لتنفيذ مهمة واحدة، دون الحاجة إلى استدعاء برامج أو مقاطع أخرى.

- كلما كانت خاصية التماسك عالية للبرنامج أو المقطع البرمجي كلما كان أفضل تصميم البرنامج أفضل.
- و يمكن قبول خاصية تماسك من الدرجة الوسطى.
- و يستحسن في تصميم المقاطع تجنب خاصية التماسك المدنية.

٢ . أساسيات تصميم البرمجيات

- خاصية التماسك يوجد لها سبع مستويات:
  ١. التماسك الوظيفي Functional Cohesion.
  ٢. التماسك المعلوماتي Information Cohesion.
  ٣. التماسك الاتصالي Communicational Cohesion.
  ٤. تماسك البرامج الفرعية Procedural Cohesion.
  ٥. التماسك المؤقت Temporal Cohesion.
  ٦. التماسك المنطقي logical Cohesion.
  ٧. التماسك العرضي Coincidental Cohesion.
- هذا التدرج نسبي، حيث يعد التماسك الوظيفي و التماسك المعلوماتي هما الأقوى، بينما التماسك المنطقي و التماسك العرضي هما الأضعف.

٢ . أساسيات تصميم البرمجيات

- **التماسك العرضي:** عندما يقوم البرنامج او المقطع البرمجي بأداء اكثر من مهمة غير مترابطة. يصعب صيانة هذا النوع او الاستفادة منه في برمجيات اخرى. ويمكن تحسينه بتجزئة البرنامج الى اجزاء اخرى، وهذا النوع ايضا يعتبر الاسوأ وغير مقبول على الاطلاق.
- **التماسك المنطقي:** يعد البرنامج متماسكا منطقيا اذا كان يؤدي مجموعة من المهمات المترابطة.
- **التماسك المؤقت:** يكون البرنامج متماسكا بصورة مؤقتة اذا كان يؤدي مجموعة مهام مترابطة من حيث الزمن.

٢ . أساسيات تصميم البرمجيات

- **التماسك الاجرائي:**
  - يعد البرنامج متماسكا فرعيا اذا كان يؤدي مجموعة مهام ذات علاقة بالبرنامج الفرعي نفسه.
  - ويعتبر اقوى من التماسك المؤقت لعدة اسباب منها: ان المهام في البرنامج الفرعي ذات طبيعة واحدة بينما في التماسك المؤقت كان عامل الوقت هو الذي يجمع بين المهام.
- **التماسك الاتصالي:**
  - يشبه تماسك البرامج الفرعية، يؤدي مجموعة مهام ذات علاقة بالبرنامج الفرعي نفسه، بالإضافة الى ان المهام جميعها تتعامل مع البيانات نفسها.
  - ويحدث عندما تتضمن الوحدة الوظيفية مهاما عدة تتبادل البيانات فيما بينها.

٢ . أساسيات تصميم البرمجيات

التماسك المعلوماتي: يقال اذا كان يؤدي عددا من المهام لكل مهمة نقطة بداية خاصة بها، ويصاحبها برنامج مستقل خاص بها. وتستخدم المهام جميعها تركيب بيانات واحدا. وهو تطبيق لأنواع البيانات الموجزة Abstract Data Types.

التماسك الوظيفي:

- عندما ينفذ البرنامج مهمة واحدة فقط، او هدفا واحدا، فان هذا النوع يسمى التماسك الوظيفي. ويمكن استخدامه في برامج أخرى.
- ويعد البرنامج ذو التماسك الوظيفي المصمم بصورة جيدة والذي اجتاز الاختبارات والذي يشمل على الملاحظات والتوثيق الشاملين بمثابة الاستثمار الحسن للمؤسسة.

٢ . أساسيات تصميم البرمجيات

الشكل (٤): مقطع للتماسك المعلوماتي

٣. التصميم على أساس انسياب البيانات Data–Flow Diagram (DFD)

- تعتمد هذه الطريقة على انسياب البيانات خلال المنتج مع العمليات التي تجري على هذه البيانات في اثناء انسيابها، ويستخدم عند تصميم برامج ذات حجم متوسط او كبير.
- يبدأ العمل بطريقة التصميم على اساس انسياب البيانات مع العمليات التي تجري عليها. وأول خطوة هي بناء مخطط انسياب البيانات.
- تعتبر خرائط تدفق البيانات من الوسائل المفيدة في توضيح انسياب البيانات، وخرائط تدفق البيانات عبارة عن وسائل مساعدة لتطوير انظمة البرمجيات.
- تدفق البيانات عبارة عن نموذج يمثل حركة المعلومات من مواضيع معينة عبر مجموعة من المعالجات الى مواضع اخرى.

٣. التصميم على أساس انسياب البيانات Data–Flow Diagram (DFD)

خطوات طريقة التصميم على أساس انسياب البيانات:

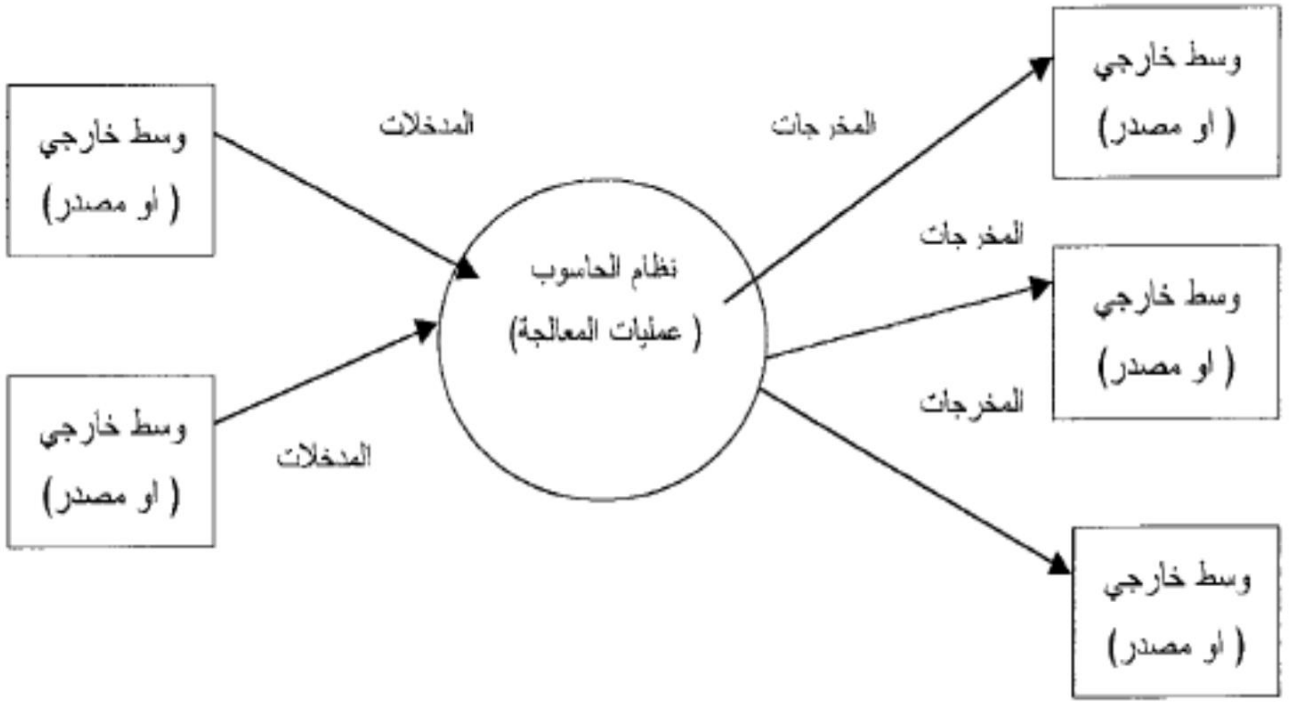
- أول خطوة في طريقة التصميم على أساس انسياب البيانات هي بناء مخطط انسياب البيانات (Data Flow Diagram DFD).
  - مخطط تدفق البيانات أو خرائط تدفق البيانات (Data Flow Diagram DFD) هي نموذج يمثل حركة المعلومات من مواضع معينة في النظام عبر مجموعة من المعالجات و العمليات إلى مواضع أخرى في النظام، فتبين حركة البيانات عند عبورها من وسائل الإدخال و العمليات التي يتم إجراؤها عليها وصولاً إلى عبورها وسائل الإخراج ص ١٠١ الشكل ٥.
  - تسمى مخططات تدفق البيانات أيضًا الخرائط العامة (Bubble Charts).

٣. التصميم على أساس انسياب البيانات Data–Flow Diagram (DFD)

-خرائط تدفق البيانات تفيد في عرض الأنظمة بأي مستوى من التفاصيل.

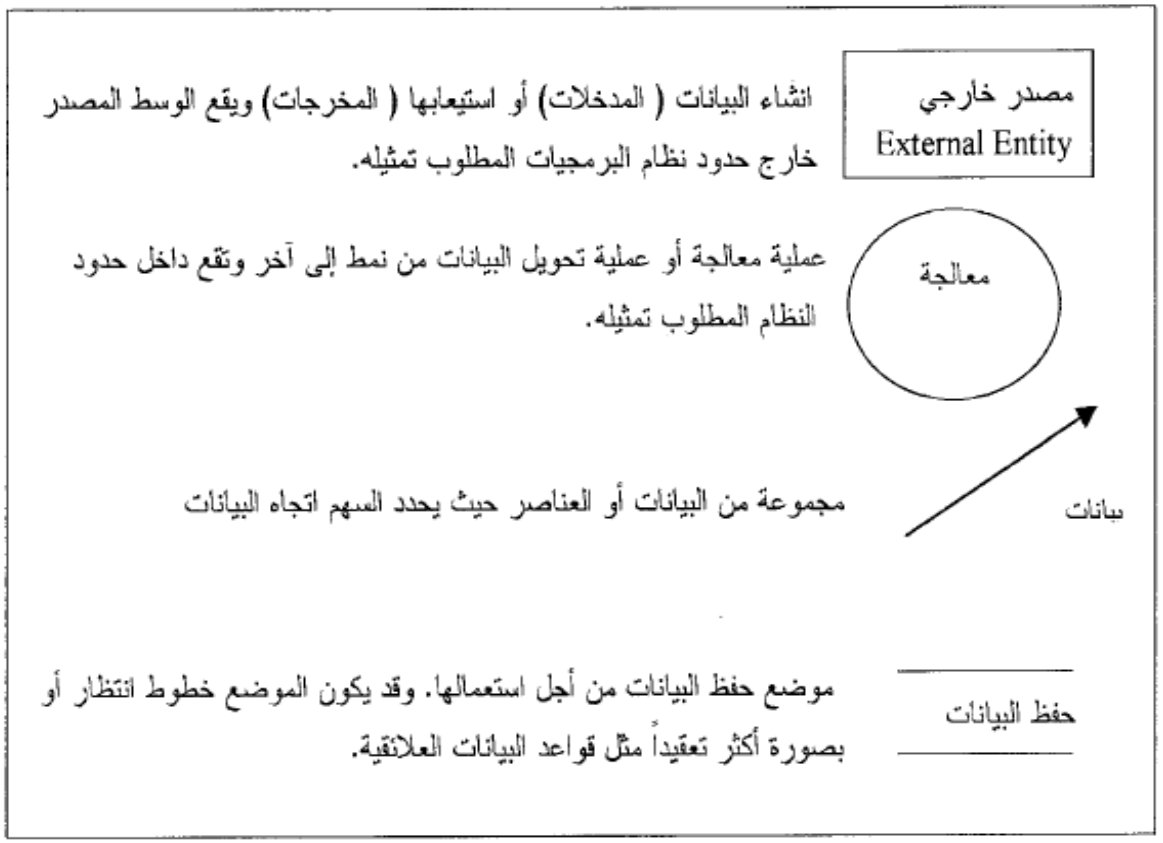
-خرائط تدفق البيانات تفيد في عرض الأنظمة بصورة هيكلية أو هرمية، فالمستوى الأول يسمى النموذج الأساسي أو نموذج السياق العام Context Model، ثم المستوى الثاني الذي يمثل تفاصيل أكثر لحركة البيانات و معالجتها.

٣. التصميم على أساس انسياب البيانات Data–Flow Diagram (DFD)



الشكل (5) : نموذج تدفق البيانات

٣. التصميم على أساس انسياب البيانات Data–Flow Diagram (DFD)



الشكل (6) : الرموز الأساسية المستخدمة في خرائط تدفق البيانات

٣. التصميم على أساس انسياب البيانات Data-Flow Diagram (DFD)

يوجد عدة طرق للتصميم و الوصول إلى أفضل تصميم بهذه الطريقة:

-البدء بدائرة كبيرة واحدة و تقسيمها إلى دوائر (Bubble) أصغر.

-البدء من البيانات الخارجة من البرنامج و تفحصها بالاتجاه المعاكس (Trace backward).

-البدء من البيانات الداخلة إلى البرنامج و تفحصها بالاتجاه نحو الأمام (Trace forward).

- الخطوة التالية في طريقة التصميم على أساس انسياب البيانات بعد الانتهاء من تصميم انسياب البيانات هي تحويل التصميم إلى مخطط هيكلي (Structured Chart)، و الذي يهدف إلى تحديد أهم العمليات التي تجري على البيانات.

٤. التصميم الكينوني Object –Oriented Design

هو أحد أنواع تصميم البرمجيات الذي ينظر إلى البرمجية على أنها مقسمة إلى مجموعات من الكينونات، تتعامل و تتفاعل مع بعضها البعض لتحقيق أهداف البرمجية، ولكل كيان استقلاليته وبياناته الخاصة به.

خطوات بناء برامج الكينونات:

- تحديد تركيب الكينون المطلوب.
- تحديد و تعريف المتغيرات التي يتضمنها كل كينون وتعريف كل متغير.
- تحديد البرامج الفرعية (الوسائل) المطلوبة لتنفيذ عمليات الكينون.
- تحديد أهداف كل برنامج فرعي و أهمية المتغيرات الأساسية، و تحديد الشروط الأولية و النتائج المتوقعة.

٤. التصميم الكينوني Object –Oriented Design

الركائز الأساسية في التصميم الكينوني للبرمجيات:

١. **الكينون**

هو وحدة برمجية لها استقلاليتها، و هو تجسيد لشيء واقعي يؤدي مهمة محددة، يحتوي بياناته الخاصة به التي تتعلق بالشيء الذي يجسده، كما يحتوي على العمليات التي يسمح بتنفيذها على تلك البيانات، و يمتلك القدرة على حفظ أثر إجراء هذه العمليات.

٢. **العمليات**

مجموعة من المهام التي يقوم الكينون بإجرائها على البيانات الخاصة به، لتنفيذ مهام و تحقيق أهداف محددة.

٤. التصميم الكينوني Object –Oriented Design

٣. **الإشعارات**

المهام التي يؤديها الكينون هي خدمات يقدمها للكينونات الأخرى، فعند استدعاء كينون ليطلب منه تنفيذ مهمة تفيد كينونًا آخر، يتم عمل إشعار بتمام المهمة.

٤. **برمجة الكينونات**

من المستلزمات الضرورية في برمجة الكينونات، تحديد كينونات نظام البرمجيات، و خصائصها، و العمليات المصاحبة لها، و يعتمد التحديد على خبرة مصمم البرمجيات و مهارته.

٤. التصميم الكينوني Object –Oriented Design

٥. مزايا برمجة الكينونات في تصميم البرمجيات

تمكننا البرمجة الكينونية من:

- تصميم برمجيات ذات قوة عالية.
- تصميم برمجيات قابلة للاستخدام المتعدد.
- تصميم برمجيات قابلة للتعديلات.
- تقليل التكاليف الكلية لتطوير البرمجيات و إدامتها.
- إمكانية التعديل فقط على مقطع معين، و سينتقل التأثير إلى بقية أجزاء التصميم أو البرنامج.
- إمكانية إضافة مقاطع برمجية جديدة لا يكون لها تأثير مباشر على بقية المقاطع، أو حذف أو تعديل مقطع برمجي دون الحاجة إلى تعديل متعلقاته.

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

توفر عملية التصميم استنادًا إلى البيانات وصفًا إجرائيًا لإنتاج البرمجيات، بالتالي هي ليست مخططات انسياب البيانات (DFDs)، فهي تستخدم الأسلوب المتسلسل أو الهرمي لتمثيل تركيب المعلومات في البرمجيات.

صفات مشتركة بين طرق التصميم استنادًا إلى البيانات و طرق التصميم الكينوني:

- كلاهما يهتم بأنماط و كينونات التطبيقات على أرض الواقع.
- كلاهم يعتمد البيانات أساسًا في للتنفيذ.
- كلاهما يعتمد تمثيل البيانات لبناء تمثيل مناسب للبرمجية.
- كلاهما يمتلك وسيلة محدد للتصميم و تسلسل البيانات.
- كلاهما يطبق فكرة تجريد البيانات، و يستخدم مجموعة من العمليات ذات العلاقة.
- لكنهما يختلفان في تعريف الكينون (Object)، ففي التصميم استنادُا إلى البيانات يعتبر بيانات فقط (Data)، أما في الثانية يعتبر بيانات و معالجة (Data & Process).

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

يوجد نوعين ضمن التصميم استنادًا إلى البيانات:

أولاً: تصميم جاكسون لتطوير الأنظمة (Jackson Development System: JSD)

هي طريقة متكاملة من التحليل، فالتصميم، إلى التنفيذ و التطبيق، و تعد نموذجًا لدورة حياة البرمجيات في التصميم و البناء، و تستند في عملية تحليل المتطلبات و تصميم البرمجيات إلى المفهوم الهندسي العلمي من حيث التركيب و التحديث المستمر، و التدرج في التصميم من العام إلى الخاص.

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

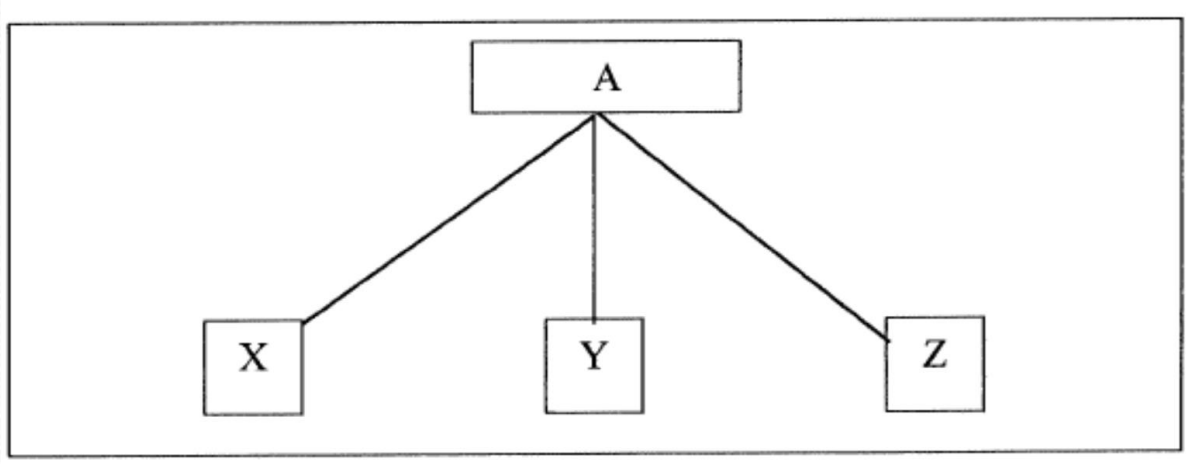
تعتمد طريقة جاكسون على فكرة أن البرمجيات تعتمد على المدخلات و المخرجات اللذان يحددان الشكل العام، فتتكون هذه الطريقة من الخطوات:

- تخطيط الأشكال التركيبية للبيانات، لتبين حركة البيانات إلى النظام كمدخلات، و من النظام كمخرجات.
- إيجاد شكل تركيبي واحد للبرنامج، من خلال مجموع من الأشكال البيانية التركيبية.
- تحديد كافة العمليات الأساسية التي يستخدمها البرنامج في تنفيذ المهمة المطلوبة (تكتب هذه العمليات حسب مفهوم لغة البرمجة المستخدمة).
- ربط العمليات الأساسية بمواقعها المناسبة في الشكل التركيبي للبرنامج.
- تحويل تركيبة البرنامج من الأشكال التخطيطية إلى الشكل الإنشائي (على شكل جمل).

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

الأشكال تركيبية في طريقة جاكسون تنقسم إلى ثلاثة أنواع:

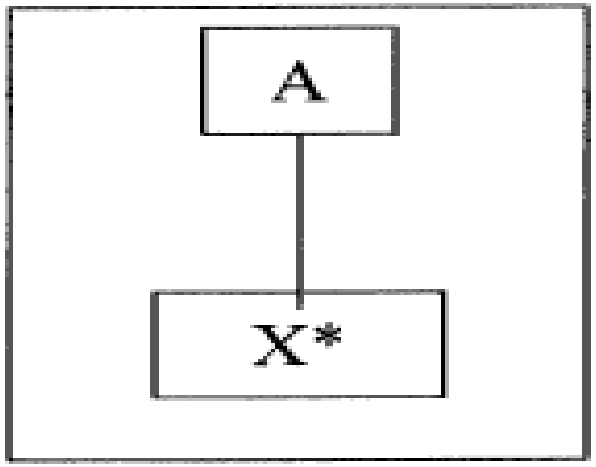
١. التتابع Sequence:



يفيد الشكل أن تركيب البيانات A يتكون من X ويتبعه Yويليه Z. أما معناه في مفهوم البرمجة فهو أن المقطع البرمجي A يتطلب تنفيذ المقطع X،Y والمقطع Z على الترتيب.

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

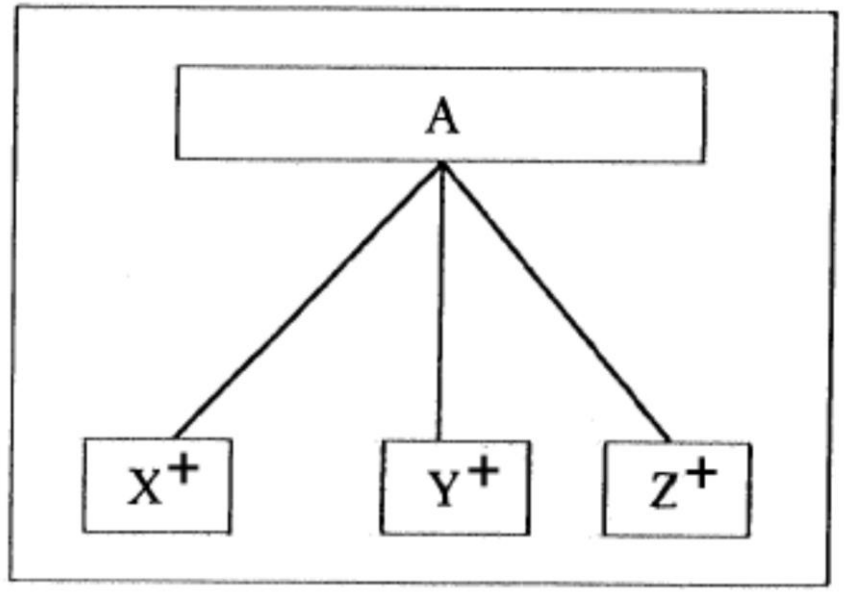
٢. التكرار Iteration :



ويمكن توضيح ذلك بالشكل (8)، حيث يفيد هذا الشكل أن تركيب البيانات A يتكون من تكرار X صفر أو أكثر من المرات .أما معناه في مفهوم البرمجة فهو أن المقطع البرمجي A يتطلب تنفيذ المقطع X صفر مرة أو أكثر.

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

٣. الاختيار Selection:



هام جداً: أنظر المثال في صفحة ١٠٩ – ١١٠

ويمكن توضيح ذلك بالشكل (9)، حيث يفيد هذا الشكل أن تركيب البيانات A يتكون من عنصر واحد من العناصر X، Y، Z. أما معناه في مفهوم البرمجة فهو أن : المقطع البرمجي A يتطلب تنفيذ مقطع واحد من المقاطع X، Y، Z. ويمكن توضيح الأشكال الثلاثة في مخطط هرمي واحد كما في الشكل (10).

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

- من أهم محاسن طريقة جاكسون إنشاء التفرع المنطقي من تعريف البيانات، فباتباع هذا النهج، يكون لكل مقطع بياناته مدخلاته الخاصة و مخرجاته الخاصة، و هذا يؤدي إلى تشعب طرق تراكيب البيانات.
- تعالج طريقة جاكسون هذا التشعب من خلال دمج تراكيب البيانات المتعددة عبر معرفة نقاط التشابه بين التراكيب، و لكن كلما كان حجم البرنامج كبيرًا فإن عدد التراكيب سيكون كبيرًا مما يؤدي إلى صعوبات في دمج التراكيب.
- الرامج المعقدة لا يعتمد بناؤها على شكل البيانات المدخلة و المخرجة، كبرامج السيطرة على آلات التصنيع.
- تعد طريقة جاكسون جيدة عندما يكون المقطع البرمجي صغيرًا، و منطق المسألة يسيرًا.
- طريقة جاكسون ينقصها إمكانية تجزئة النظام (قد يتطلب التصميم معرفة البيانات المشتركة بين المقاطع أكثر من تسلسل الجمل).

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

ثانيًا: التصميم وفق تراكيب البيانات لتطوير الأنظمة (Data Structures Systems Development: DSSD)

هي جزء من طريقة شاملة لتطوير الأنظمة، طورها وورنر عام ١٩٨١م، و تنقسم طريقة التصميم هذه إلى قسمين: التصميم النطقي (Logical Design)، و التصميم الفيزيائي (Physical Design).

التصميم النطقي (Logical Design): يعتني بالمخرجات، و واجهة التخاطب، و الإجراءات البرمجية.

التصميم الفيزيائي (Physical Design): يتبع الشق الأول، و يركز على جمع و تنسيق جميع أجزاء البرمجية، لتحقيق مواصفات الأداء الجيد، و إمكانية الصيانة، و ما إلى ذلك من المواصفات التي تحقق الوضع الأفضل للبرمجية.

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

جزء التصميم المنطقي ينقسم إلى قسمين:

أ- هيكلية المخرجات المنطقية (Logical Output Structure: LOS)

في هذا القسم يتم تنظيم وحدات البيانات التي تتضمنها المسألة قيد الدراسة بشكل متسلسل أو هرمي (hierarchically)، ويتم ذلك من خلال الخطوات التالية:

١) يتم تقييم المسألة أو المعلومات ذات العلاقة بها، ويتم بعد ذلك رصد وتسجيل وحدات البيانات التي لا تقبل التجزئة أكثر.

٢) يتم تسجيل عدد مرات تكرار هذه الوحدات.

٣) يتم تقييم وحدات البيانات القابلة للتجزئة.

٤) يتم استنتاج تمثيل بالأشكال لمرحلة LOS.

٥. طرق التصميم المستندة إلى البيانات Data Oriented Design Method

ب- هيكلية المعالجة المنطقية (Logical Process Structure: LPS)

هذا القسم إجرائي، وتتم فيه عملية معالجة للجزء الناتج من القسم السابق LOS، حيث تتم إضافة تعليمات للمعالجة لكل وحدة بيانات من النوع القابل للتجزئة، ويتم تنفيذ القسم LPS هذا من خلال الخطوات التالية:

١) بعد تحديد وحدات البيانات المعنية، تتم إضافة المحددات البدء (BEGIN) والنهاية (END) لها جميعاً.
٢) يتم تعريف جميع تعليمات البدايات والنهايات وعمليات المعالجة اللازمة.
٣) يتم تحديد جميع الحسابات أو المعالجة غير العددية.
٤) يتم تحديد جميع تعليمات المخرجات وعمليات المعالجة ذات العلاقة.
٥) يتم تحديد جميع تعليمات المدخلات وعمليات المعالجة ذات العلاقة.

٦. التصميم الواقعي Real-Time Design

- يخضع هذا التصميم لمجموعة قوية من القيود ويرتبط بشكل قوي بمجال المسألة ويأخذ الاطار الحقيقي والزمني لعمل وتنفيذ البرمجية بعين الاعتبار.
- بسبب خضوع هذا النوع لمجموعة قوية من القيود كان لا بد أن تنتج مثل هذه البرمجيات وفي الذهن انها ستنفذ ضمن بيئة متعددة الجوانب مثل:
  - معمارية المعدات والبرمجيات معاً
  - مواصفات نظام التشغيل ومتطلبات المسألة التطبيقية نفسها
  - لغة البرمجة المختارة ومواصفات التصميم

Questions or Comments?