العنوان:	مساهمة في دراسة وتصميم وتنفيذ لغات البرمجة: دراسة حالة: لغات البرمجة غرضية التوجه
المؤلف الرئيسي:	الدهني، شادي سمير
مؤلفين آخرين:	صبح، محمد، فلوح، غسان(مشرف)
التاريخ الميلادي:	2000
موقع:	دمشق
الصفحات:	1 - 161
رقم MD:	573587
نوع المحتوى:	رسائل جامعية
اللغة:	Arabic
الدرجة العلمية:	رسالة ماجستير
الجامعة:	جامعة دمشق
الكلية:	كلية العلوم
الدولة:	سوريا
قواعد المعلومات:	Dissertations
مواضيع:	لغات البرمجة، الخوارزميات، الحاسبات الإلكترونية ، تصميم البرامج
رابط:	http://search.mandumah.com/Record/573587

الجمهورية العربية السورية
جامعة دمشق
كلية العلوم
قسم الرياضيات

مساهمة في دراسة وتصميم وتنفيذ لغات البرمجة
دراسة حالة : لغات البرمجة غرضية التوجه

دراسة أعدت لنيل درجة الماجستير في البرمجة

إعداد الطالب

شادي سمير الدهني

إشراف

الدكتور محمد صبح
أستاذ في كلية العلوم - قسم الرياضيات

الدكتور غسان فلوح
أستاذ مساعد في كلية الهندسة الكهربائية

عام ٢٠٠٠

الجمهورية العربية السورية
جامعة دمشق
كلية العلوم
قسم الرياضيات

مساهمة في دراسة وتصميم وتنفيذ لغات البرمجة
دراسة حالة : لغات البرمجة لغرضية التوجه

دراسة أعدت لنيل درجة الماجستير في البرمجة

إعداد الطالب
شادي سمير الدهني

إشراف

الدكتور محمد صبح
أستاذ في كلية العلوم - قسم الرياضيات

الدكتور غسان فلوح
أستاذ مساعد في كلية الهندسة الكهربائية

تمت مناقشة هذه الأطروحة بتاريخ ٢٠٠٠/٨/١٧ . أمام لجنة الحكم المؤلفة من السادة :

الدكتور محمد صبح	أستاذ في كلية العلوم	" رئيسا "
الدكتور نايف طلي	أستاذ مساعد في كلية العلوم	" عضوا "
الدكتور جمال اللبن	مدرس في كلية العلوم	" عضوا "

الإهداء

إلى ينبوع العطف والحنان الذي لا ينضب .....

إلى من أمضت شبابها وعمرها لتوطني إلى بر الأمان .....

إلى عينيها الغاليتين وقلبها الذي أحب وأعطى بلا حدود .....

إلى أروع كلمة قلدتها الإنسانية واحتضنتيا .....

أمي الحبيبة

إلى الجبهة السمراء واليد المعطاء .....

إلى من كان لي قدوة مثالية وزرع بي التصميم والإرادة القوية في مواجهة الصعاب .....

إلى من تصمت الكلمات خجلاً أمام عطائه .....

أبي الغالي

إلى من معهم تحلو الأيام وتكبر الفرحة .....

إلى رفاق الصبا والدرب الطويل .....

إلى أغاني الطفولة وقصيدة المستقبل .....

اخوتي الأعزاء

إلى الأخوة التي لم تلدهم أمهاتنا .....

إلى من شاركوني درب الدراسة وأيام الفرح والسرور .....

إلى كل صديقة وصديق .....

أصدقائي الكرام

إلى كل مربي وكل باني .....

إلى من كان رمزاً للعطاء والإنسانية .....

إلى من كاد أن يكون رسولا .....

أساتذتي الأفاضل

شكر وتقدير

ترى بأي كلمات قدسية أستطيع التعبير عما يجيش بخاطري من مشاعر الشكر والامتنان ، إلى من جاد بالكثير الكثير من علمه ومد لي يد العون وأنارني بفكره ، وكان مرجعاً وعوناً لي في كل معضلة وفي كل مشقة ، ملأ كفيه مراراً وتكراراً من معين العلم لأستنير بعلمه ، فكان نور معرفتي وامتدت أياديه البيضاء مضيئة لتظهر آثار فكره الفريد في علمي هذا ليتم بعون الله وعونه .

أوجه شكري هذا إلى

الأستاذ الفاضل الدكتور
محمد صبح

الأستاذ الفاضل الدكتور
غسان فلوح

كما يطيب لي أن أوجه جزيل الشكر إلى لجنة الحكم لمساهمتهم البناءة في تنقيح هذه الرسالة .

د. محمد صبح　　د. نايف طلي　　د. جمال البن

الفصل الأول

الدراسات النظرية عن لغات البرمجة التقليدية

الفصل الثاني

البرمجة غرضية التوجه

Object Oriented Programming

أولا- مقدمة ٥٤

٥٤١٢٦٣

ثانيا- الأغراض OBJECTS

ثالثاً- دراسة تخصيصات البرمجة غرضية التوجه .
SPECIFICATIONS OF OOP

رابعاً- اللغات الصورية VISUAL LANGUAGES

خامساً- دراسة تطبيق البرمجة غرضية التوجه
IMPLEMENTATION OF OOP

الفصل الثالث
تصميم النموذج العملياتي

أولا- القائمة The List

ثانيا- الأرتال والمكدسات Stacks & Queue

الفصل الرابع

التمثيل البرمجي للنموذج العملياتي

الفصل الخامس

نتائج البحث

مقدمة

لم يكن لأحد أن يتصور تغيير وجه العالم على النحو الذي نشهده اليوم ، حيث تحـــول هـــذا العالم المترامي الأطراف إلى قرية معلومات صغيرة Information Village بعد أن أصبحـت المعلوماتية اليوم عصب الحياة بحيث أنها باتت تشكل مقياساً دقيقاً لتقدم المجتمعات ، فالمجتمع الأكثر تطوراً هو المجتمع الذي يملك أكثر المعلومات تنظيماً .

لقد دخل الحاسوب معظم مجالات الحياة العملية والعلمية ، وأصبحت المعلوماتية لغة العصـــر ومرتكزاً لنهضة حضارية جديدة على أبواب القرن الحادي والعشرين الذي يشهد تحولاً فـــي أنماط التنمية البشرية وخططها وأساليبها في شتى فعاليات المجتمع . ســـواء فـــي المجـــالات الثقافية والإدارية أو القطاعات الخدمية والصناعية والإنتاجية . ويضع العالم على أبواب ثورة حضارية علمية ستنعكس على إحداث تغيرات جذرية في أنمـــاط العمـــل الإداري والمكتبـــي والتجاري وإدارة المشاريع الإنتاجية والصناعية .

يقوم هذا البحث على دراسة موديلات البنى الأساسية للغات البرمجة من وجهة نظر تكاملـــها وعلاقاتها وكيفية تنفيذها التي تقوم على بنية قواعـــد المعطيـــات Data Structures ولغـــات البرمجة غرضية التوجه Object Oriented Programming Languages ، وعلى البحث فـــي المواضيع الأساسية المتعلقة بالنظم غرضية التوجه بشكل عام من حيث بناها الأساسية وكيفية تنفيذها .

يعالج الفصل الأول من هذا البحث الدراسات النظرية عن لغات البرمجة التقليدية ، حيث تـــم في هذا الفصل دراسة بنى المعطيات التقليدية مثل المتحولات الصحيحة والحقيقية Real and Integer Variables والتي تشكل حجر الأساس لبنى المعطيات الديناميكية ، حيث يمكن لـــهذه البنى الأساسية أن تغير قيمها أثناء تنفيذ البرنامج ولكنها تبقى محافظة على شـــكلها ، وهـــذا يعني أن الحجم المخصص لها في ذاكرة الحاسوب سيبقى ثابتاً أثناء تنفيذ البرنامج . ومن ثـــم دراسة بنى المعطيات الديناميكية التي يمكنها أن تغير شكلها أثناء دورة حياة البرنامج . وتـــم أيضاً في هذا الفصل دراسة تخصيـــص وتطبيـــق Specification and Implementations أغراض بنى المعطيات الأساسية والديناميكية مع الأمثلة الموضحة لتلك الدراسة .

ويبحث الفصل الثاني في المواضيع المتعلقة بلغات البرمجة غرضية التوجه ودراسة تخصيصات أغراض معطيات البرمجة غرضية التوجه ، ومن ثم إسقاط الدراسة التي تمت في الفصل الأول على أغراض معطيات البرمجة غرضية التوجه بهدف استنباط خوارزمية تطبيق أغراض وصفوف البرمجة غرضية التوجه Classes and Objects Implementations .

ويبحث الفصل الثالث في تصميم نموذج معطيات عملياتي عن طريق دراسة تخصيص بنى المعطيات الديناميكية Specifications of data structures (مثل القائمة List والمكدس Stack والرتل ...Queue الخ) ومن ثم دراسة تطبيق هذه المعطيات باستخدام خواص البرمجة غرضية التوجه ووفقا للخوارزمية المقترحة في نهاية الفصل الثاني .

ويبين الفصل الرابع التمثيلات البرمجية للنموذج العملياتي المدروس في الفصل الثالث ، ويكون التمثيل البرمجي لجميع أجزاء النموذج العملياتي بشكل خوارزمية برمجية أو شيفرة برمجية Code أو شبه برنامج Pseudo Code وباستخدام صفات البرمجة غرضية التوجه .

وأتمنى من قارئ هذا البحث أن يزودنا بالإقتراحات المفيدة حول محتوياته . وآمل أن أكون قد قدمت لمكتبتنا العربية مرجعا مفيدا في علوم الحاسوب ، ولي كبير الأمل أن يحقق هذا البحث الغاية والفائدة المرجوتين منه في خدمة قارئيه والوطن .

والله ولي التوفيق

شادي سمير الدهني

دمشق ٢٠٠١

الفصل الأول

الدراسات النظرية عن لغات البرمجة التقليدية

أولا- مقدمة :

يمكن أن يقود ترتيب أرقام أو رموز لخوارزميات وبنى معطيات إلى تصميم لغة برمجية على افتراض تطبيق هذا الترتيب على الحاسوب وهذا هو مفهوم تصميم اللغات البرمجية ببساطة بالغة.

صممت المئات من لغات البرمجة وطبقت واستخدمت إلا أن معظم المبرمجين لا يخاطرون باستخدام أكثر من القليل من اللغات والكثير منهم قد حصر أعماله على لغة أو اثنتين ، والسؤال الذي يطرح نفسه هنا ، ما الذي يكون أكثر فائدة هل هو دراسة لغات برمجية متنوعة ومختلفة علما انه قد يكون إحداها غير محببا بالنسبة لنا أو جعل المبرمج يهتم فقط باللغة التي يرغبها ؟

في الحقيقة هناك أسباب كثيرة تجعلنا نقوم بدراسة لغات متنوعة ومختلفة ومن أهم هذه الأسباب ما يلي :

١- **التعمق بفهم اللغة التي يستخدمها المبرمج** : جهزت العديد من اللغات بخصائص ذات فائدة كبيرة للمبرمج إذا استخدمت بشكل صحيح ، ولكنها تضيع الكثير من وقت الحاسوب إذا استخدمت بشكل خاطئ وتقود المبرمج إلى أخطاء هو بغنى عنها ، حتى أن المبرمج الذي استخدم لغة معينة عدة سنوات فيمكن أن يكون فهمه لبعض الخصائص ضعيفا أو ليس كاملا ، والمثال الأكثر وضوحا هنا هو خاصة التعاودية ، إن هذه الخاصة موجودة في عدة لغات وعندما تستخدم بشكل صحيح فإننا سوف نحصل على خوارزميات قوية ومتكاملة ، لكن في حالات أخرى يمكن أن تؤدي إلى ازدياد ضخم في وقت التنفيذ للخوارزمية البسيطة ، والأكثر من ذلك دراسة تكلفة هذه الخاصة يعتمد على اللغة المستخدمة .

٢- **من أجل زيادة المفردات المفيدة في بناء البرمجة** : عند البحث في المعطيات وبنى البرنامج المناسبة لحل المشكلة ، يقودنا شيئا ما للتفكير فقط بالبنى القابلة للتعبير عما نرغب به فورا وبشكل مألوف . يزيد المبرمج من قوته في البرمجة عن طريق دراسة البنى المجهزة في عدة لغات "Vocabulary". إن فهم تقنية التطبيق هو أمر هام لدى المبرمج لأنه من المحتمل أن يستخدم بنية معينة غير موجودة في اللغة المستخدمة لديه

بشكل مباشر ، وعلى المبرمج في هذه الحالة أن يجهز بنيته الخاصة من العناصر الأولية المجهزة في اللغة .

٣- **السماح باختيار أفضل لغة برمجية :** تعطي المعرفة الجيدة بعدة لغات برمجية للمبرمج فرصة في اختيار أفضل لغة متاحة لإنجاز المشروع . مثلاً إعادة ترتيب معطيات مدخلة، إن الشيفرة لمثل هذا البرنامج بلغة فورتران أو كوبول يمكن أن تكون مزعجة وتحتاج إلى وقت من أجل كتابتها أما كتابة الشيفرة بلغة تتصف بمعالجة شريطية (مثل لغة سنوبول٤) فيمكن أن تأخذ دقائق من العمل وأسطر قليلة .

٤- **تبسيط تعلم لغة جديدة :** تسمح المعرفة الجيدة في بنى اللغات المختلفة وتقنيات التطبيق للمبرمج أن يتعلم لغة برمجية جديدة وبشكل أسرع .

٥- **تبسيط عملية تصميم لغة جديدة :** يفكر القليل من المبرمجين في البحث عن المواضيع المتعلقة بتصميم لغات البرمجة ، إن أي برنامج يستخدم مفهوم واجهة المخاطبة (user interface) هو في الحقيقة عبارة عن شكل من أشكال لغات البرمجة ، إن مصمم البرامج التي تعتمد على مفهوم التخاطبية مثل محرر النصوص ، برامج الرسوم ، يهتم بنفس المفاهيم التي يهتم بها مصمم لغات البرمجة ، وإذا كان المبرمج متفهماً لبنى اللغات وطرق تطبيقها فإن عملية تصميم لغة برمجية جديدة لن تكون صعبة بالنسبة له .

مما سبق نلاحظ أن دراسة تنوع واختلاف لغات البرمجة هو أفضل بكثير من الاهتمام بلغة أو اثنتين ، لأنه في الحقيقة قد يكون لدينا نفس الصفة لكن في لغتين مختلفتين يمكن أن تطبق بطريقتين مختلفتين وبالتالي الحصول على اختلاف في تكلفة الاستخدام والنتيجة ذاتها .

ثانياً-لغات البرمجة

١-تطور لغات البرمجة Development of Programming Languages :

تطور تصميم لغات البرمجة وطرق تطبيقها بشكل سريع منذ عقد الخمسينات حيث وجدت لغات فورتران وليسب وكوبول واستخدمت في الستينات لغات مثل PL/I وسنوبول٤ و APL. وتمثل لغة باسكال وADA و JAVA اللغات الأكثر حداثة ، إن أهم التأثيرات الأساسية في تطور تصميم اللغة هي :

١- **تطور الأجزاء المادية للحاسوب وأنظمة التشغيل :** لقد تطورت أجهزة الحاسوب بشكل سريع بدءاً من البطء والكلفة إلى السرعة والسهولة في التعامل ، وفي نفس الوقت تطور

أنظمة التشغيل قد أثر بشكل كبير على بنى وتكلفة استخدام خصائص اللغات عالية المستوى .

٢- **التطبيقات** : دخلت تطبيقات الحاسوب الهائلة معظم ميادين الحياة العلمية والعملية والعسكرية ، وأثرت هذه التطبيقات بشكل كبير في تصميم لغات جديدة وإعادة بناء اللغات القديمة .

٣- **طرق البرمجة** : إن تطور طرق البرمجة لكتابة برامج طويلة ومعقدة قد أثر في عملية تصميم لغات البرمجة .

٤- **طرق التطبيق** : لقد أثر تطور طرق تطبيق أفضل في اختيار الصفات التي يرغب بوجودها في التصميمات الجديدة .

٥- **الأبحاث الدراسية** : عمق البحث في المفاهيم الأساسية لتصميم اللغة وتطبيقها باستخدام القواعد الرياضية فهمنا لنقاط الضعف وقوة صفات اللغة وبالتالي أثر في تصميم اللغات الجديدة .

٦- **توحيد المقياس** : من الضروري وجود لغات قياسية من أجل أن يتم تطبيقها بسهولة مع اختلاف أنظمة الحاسوب ولكي تسمح بنقل البرامج من حاسوب إلى آخر بسهولة ، هذا المفهوم قد أثر بشكل كبير في البحث عن تصميمات جديدة للغة .

٢-ما الذي يجعل اللغة جيدة What makes a good language :

إن تصميم اللغات عالية المستوى يبدو حالياً تاماً ، فالعديد من اللغات يستحق عبارة النجاح ، ويعود العامل الأساسي في عملية نجاح أو فشل لغة معينة إلى أسباب خارجية عن اللغة نفسها، فمثلاً من أهم أسباب نجاح لغة كوبول في الولايات المتحدة يمكن أن تعود إلى التعليمات الحكومية لاستخدامها في مناطق محددة ، **لنبحث الآن في بعض الأسباب التي تجعل المبرمجين يهتمون بلغة معينة أكثر من أخرى :**

١- **الوضوح ، البساطة ، وحدت مفهوم اللغة** : اللغة يجب أن تكون مساعدة للمبرمج قبل أن يبدأ بعملية تصميم الشيفرة ، ويجب أن تزوده بمجوعة مفاهيم واضحة وبسيطة الاستخدام والتي بواسطتها يستطيع المبرمج تطوير خوارزمياته ، واللغة المرغوبة هي اللغة التي لا تختلف كثيراً عن غيرها في استخدامها ، إن وضوح اللغة يحدد مدى أهميتها .

٢- وضوحية قواعد اللغة (Syntax) : يؤثر تركيب اللغة بشكل واضح على سهولة كتابة البرامج واختبارها وفهمها وإعادة معالجتها ، ويجعل سياق اللغة الجيد كتابة البرامج أكثر

سهولة لكن أكثر صعوبة للقراءة عندما نحتاج إلى إعادة تجهيز البرنامج ، إن المهم لدى المبرمج هو أن يعكس سياق اللغة مباشرة مفهوم الخوارزمية ، حيث يؤدي كتابة برنامج كامل بعبارات بسيطة كالعبارات التكرارية والعبارات الشرطية إلى الحصول على خوارزمية بسيطة وسهلة الفهم والتصحيح والتجديد مستقبلاً. على كل ، من الصعب إيجاد لغة برمجية متكاملة تسمح للتعبير عن البرامج بخوارزميات بسيطة . إن من أكبر حجج لغة باسكال على لغة فورتران هي أن برامج لغة فورتران تستخدم عبارات غير بنيوية (مثل goto) وبالتالي لا يعكس شكل الخوارزمية بشكل مباشر مفهومها .

٣- **التخصيص للتطبيقات** : يجب أن تجهز اللغة ببنى معطيات وعمليات وعبارات تحكم وسياق طبيعي للمسألة لكي تحل . إن من أهم الأسباب الرئيسية لعملية تزايد اللغات هي الحاجة للتخصيص أي أن لغة معينة سوف تستخدم من أجل مجموعة معينة من التطبيقات وبالتالي الحصول على برامج أفضل . فمثلاً لغة كوبول هي لغة جيدة لتطبيقات العمل ومعالجة الملفات أما لغة سنوبول٤ فهي جيدة للمعالجات الشريطية .

٤- **سهولة تعامل اللغة مع البنى الجديدة** : سوف نوضح هذه الفكرة بالمثال التالي بفرض أنه طلب منا برنامج لمعالجة تصنيف الطلاب في جامعة معينة عندئذ سوف يكون لدينا معطيات مجردة مثل " الطالب " ، " القسم " ، " المدرس " ، " قاعة المحاضرات " وسوف يكون لدينا عمليات مجردة أيضاً مثل " إسناد الطالب إلى قسم معين " ، " تخصيص قاعة المحاضرات إلى قسم معين " ... الخ . المعطيات السابقة عبارة عن معطيات ليست مجهزة بشكل مباشر بالحاسوب ، وهنا على المبرمج أن يصمم تجريدات مناسبة لحل المشكلة ومن ثم تطبيق هذه التجريدات باستخدام الخصائص المجهزة بلغة البرمجة المستخدمة لديه . ويمكن هنا للغة البرمجة أن تكون مساعدة أو معيقة للتعبير عن هذه التجريدات . إن على لغة البرمجة السماح للمبرمج بأن يستخدم هذه التجريديات الجديدة مع برامج أخرى دون الحاجة إلى الخوض في تفاصيل هذه التجريدات .

٥- **سهولة التحقق من البرنامج** : إن عملية التحقق من صحة أداء البرنامج هو موضوع هام ، هناك عدة طرق للتحقق من صحة أداء البرنامج فيمكن أن يتم ذلك عن طريق قراءة نصوص البرنامج وفحصها أو عن طريق اختبار إنجاز عمل البرنامج وذلك بإدخال معطيات معينة وفحص النتائج . إن اللغة التي تجعل عملية اختبار البرامج صعبة هي بعيدة الاستخدام عن اللغة التي تجعل عملية فحص البرامج سهلة .

٦- **بيئة البرمجة** : يجعل وجود الأدوات البرمجية التي تساعد اللغة في معالجة المعطيات البرمجة باللغة الضعيفة المحاطة بتلك الأدوات أبسط بالعمل من البرمجة باللغة القوية التي لا تحتوي على أدوات محيطة مساعدة .

٧- **نقل البرامج** : إن اللغة التي تتوافق خصائصها مع أنظمة تشغيل الحواسيب تجعل من عملية نقل البرامج المصممة في حاسوب ما إلى آخر أكثر بساطة .

٨- **تكلفة الاستخدام** : كانت عملية حساب كلفة استخدام البرامج مهملة سابقا ، لكن حاليا هي عنصر أساسي في عملية نشر لغات البرمجة ، إلا أنه هناك مقاييس مختلفة لمفهوم التكلفة :

آ- كلفة إنجاز البرنامج : إن البحث في تصميم مترجمات ومسجلات ذات تكلفة قليلة هو أمر هام ، ويأخذ بعين الاعتبار تكلفة إنجاز البرنامج في إنتاج برامج ضخمة سوف تنفذ عدة مرات .

ب- كلفة ترجمة البرنامج : عندما تستخدم لغة ما للتدريس مثل لغة فورتران ، يكون السؤال الأهم في كفاءة المترجم (compiler) أكثر من قوة الإنجاز لأن برامج الطلاب سوف تترجم مرات عديدة من أجل التنقيح إلا أنها تنفذ مرات قليلة ، ففي مثل هذه الحالة من الضروري أن يكون لدينا مترجمات قوية وسريعة .

ج- تكلفة إنتاج البرنامج وفحصه واختباره : يكون لدينا هذا واضحا في لغة مثل APL من أجل صف محدد من المسائل فالحل يمكن أن يكون بتصميم وتشفير واختبار البرنامج ومن ثم وضعه تحت الاستخدام بأقل وقت ضائع من وقت وطاقة المبرمج ، يؤثر بشكل كبير مثل هذا النوع من التكلفة الكلية في اختيار اللغة المستخدمة .

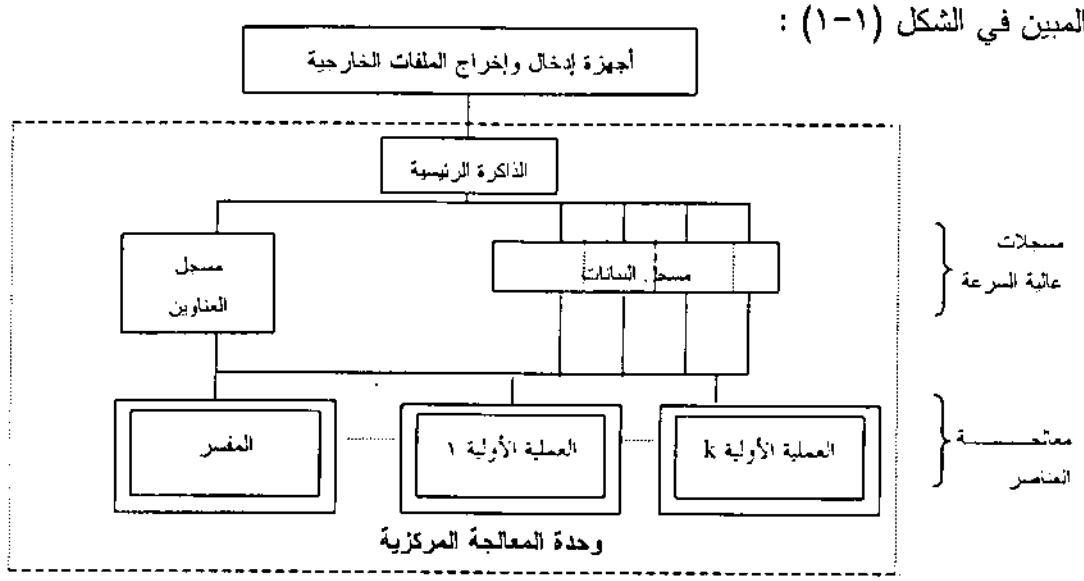
د- تكلفة الحفاظ على البرنامج : وجد الكثير من الدارسين أن التكلفة الأكبر في أي برنامج يستخدم لعدة سنيين هي ليست في تكلفة تصميمه وفحصه واختباره لكن هي تكلفة المحافظة على البرنامج طيلة فترة الاستخدام ونقصد بالمحافظة على البرنامج تكلفة تصحيح الأخطاء المكتشفة بعد وضع البرنامج في الاستثمار وتكلفة التغيرات المطلوبة لتتوافق مع النسخ الحديثة من أنظمة التشغيل أو من الأجزاء المادية للحاسوب . إن اللغة التي تجعل من عملية تطوير أو تعديل البرنامج من قبل مبرمجين آخرين خلال عدة سنوات عملية مبسطة سوف تكون أقل كلفة للاستخدام من أي لغة أخرى .

ثالثاً – بنية الحاسوب Computer Architecture :

١- بنية الحاسوب :

لنأخذ فكرة عن بنية الحاسوب قبل التعمق بمفاهيم لغات البرمجة .

إن بنية الحاسوب هي واسعة بشكل كبير إلا أنه نستطيع توضيحها بالمخطط الصندوقي التالي المبين في الشكل (١-١) :



الشكل (١-١) مخطط صندوقي لبنية التعامل

يوضح هذا المخطط الذاكرة الرئيسية التي تتوضع فيها البرامج والمعطيات من أجل المعالجة، حيث أن المعالجة تتم عن طريق المفسر interpreter الذي يأخذ كل تعليمة لغة آلة بشكل متتالي ، ثم يفك شفرتها ، ويستدعي العمليات الأولية المصممة مع المعاملات كإدخالات ، هذه العمليات تعالج المعطيات في الذاكرة الرئيسية وفي مسجلات عالية السرعة ، وأيضاً يمكن أن تنقل البرامج أو المعطيات بين الذاكرة وأجزاء خارجية ، **نستطيع تقسيم الحاسوب إلى ستة أقسام رئيسية من وجهة النظر البرمجية وهي :**

١- **المعطيات Data :** لدينا في الحاسوب ثلاثة أجزاء رئيسية لتخزين المعطيات هي – الذاكرة الرئيسية – المسجلات عالية السرعة – الملفات الخارجية . كما يمكن أن يحتوي الحاسوب أيضاً على أنواع معطيات مبيتة تعالج بشكل مباشر عن طريق العمليات الأولية المعرفة في البنية المادية للحاسوب ، مثل الأعداد الصحيحة ، الحقيقية ، الأنواع المحرفية . ويمكن اعتبار البرنامج شكل من أشكال المعطيات لأن برامج لغة الآلة هي عبارة عن مواقع متتالية في الذاكرة ويتألف كل موقع من تعليمة أو أكثر وكل تعليمة هي عبارة عن شيفرة عملية ومعاملات محددة .

٢- **العمليات Operations** : يجب أن يبنى في البنية المادية للحاسوب مجموعة من العمليات الأولية . مثلاً يجب أن تتوفر العمليات الأربعة من أجل كل نوع من المعطيات (عملية الإضافة للأعداد الحقيقية والصحيحة ، الطرح ، .....) ، عمليات أولية لاختبار النوع ،(لا يجوز القسمة على صفر مثلاً) عمليات أولية للتحكم بأجهزة الإدخال والإخراج .

٣- **تتابع التحكم Sequence control** : بعد جلب التعليمة من الذاكرة ، يتم تفكيك جزء الأمر ثم تفعل عمليات التحكم المناسبة لهذا الأمر ، وتقوم وحدة التحكم بإرسال إشارة تفيد بدء تنفيذ العملية ، (أما جزء العنوان فترسله إلى الذاكرة (في الحالات التي يتطلب تنفيذ الأمر استرجاع معلومات من الذاكرة) ، حيث تكون الذاكرة على استعداد للوصول إلى موقع محدد فيها. تقوم وحدة التحكم بإرسال إشارة انتهاء التنفيذ لذا عليها أن تعرف متى ينتهي تنفيذ عملية ما بحيث تستطيع مباشرة بدء تنفيذ التعليمة التالية ، وبالتالي يجب أن تكون قادرة على توقيت العمليات المختلفة . وفي النهاية وعندما تتحسس وحدة التحكم انتهاء التعليمة ، تبدأ حلقة معالجة التعليمة التالية في السلسلة .

٤-**إدارة عملية التخزين Storage management** : يمكن القول بكل بساطة إن تقنية إدارة التخزين هي غير متوفرة فلا يوجد في التصميمات الحالية تسهيلات إدارة تخزين مبنية في البنية المادية لأن البرامج والمعطيات تقيم خلال إنجاز البرنامج في مكان ثابت من الذاكرة، لكن من الشائع إضافة تسهيلات في الحواسيب الأكثر تعقيداً من أجل توضعات ديناميكية مباشرة للبرامج في البنية المادية للحاسوب .

٥-**بيئة التشغيل Operating environment** : تتألف بيئة التشغيل من مجموعة من وحدات التخزين الملحقة وأجهزة الإدخال والإخراج ، تعرف هذه الأجهزة باسم العالم الخارجي للحاسوب Outside World ويجب أن تتم أي اتصالات مع الحاسوب عن طريق بيئة التشغيل .

٢- **التراكيب والمعاني Syntax and Semantics** :

يعني مفهوم التركيب في لغات البرمجة الشكل الذي سوف تكتب به البرامج ، وإن إعطاء القواعد للتراكيب في لغة برمجية ما تعني كيفية عمل العبارات ، التصريحات ، وكيفية كتابة بنى جديدة .

أما مفهوم المعاني للغة برمجية فيقصد به المعنى المعطى إلى بنى تراكيب مختلفة حيث مثلاً من أجل التصريح عن مصفوفة من النوع الحقيقي بعشرة عناصر سوف يكون التركيب بالشكل التالي :

V : array [1 .. 10] of real ;

من أجل أن نفهم معاني التصريحات (في لغة باسكال مثلاً) يجب أن نعرف المعاني من أجل التصريح عن متجهة مثلاً ، حيث يجب أن نعرف مكان التصريح في بداية البرنامج وهذا يعني أنه يتم إنشاء المصفوفة في بداية البرنامج وإبادتها في نهايته، وأن هناك شعاع يخزن عشرة عناصر يشار له ب v أثناء إنجاز البرنامج .

٣- الربط وأزمنة الربط Binding and Binding times :

في الحقيقة ليس لمفهومي الربط وزمن الربط تعريف محدد ، لكن هناك العديد من المحاولات لكتابة تعريف ثابت لمفهوم الربط وزمن الربط . فيدعى اختيار خاصة ما من مجموعة الخواص المتاحة وربطها بعنصر برنامج بالربط Binding، ويدعى الزمن اللازم من أجل أن تتم تلك العملية السابقة بزمن ربط تلك الخاصية بذلك العنصر .

٣-١- أنواع أزمنة الربط Classes of Binding Times :

لا يوجد تصريحات واضحة حول الأنواع المختلفة للربط لكن يكون الربط أثناء عملية الترجمة المتبوعة بإنجاز البرنامج المترجم ، وأهم أنواع أزمنة الربط ما يلي :

١- **زمن التنفيذ Execution time** : يتم الكثير من عمليات الربط أثناء إنجاز البرنامج في زمن التنفيذ وهذا يتضمن ربط المتغيرات مع قيمها وربط المتغيرات مع مواقع التخزين وهنا يمكن أن نميز صنفين هامين من عمليات الربط :

آ- عند الدخول إلى البرنامج الجزئي أو البلوك البرمجي : تحدث أصناف الربط الهامة في معظم اللغات في وقت الدخول إلى البرنامج الجزئي أو البلوك البرمجي أثناء إنجاز البرنامج .

ب- في نقاط معينة أثناء الإنجاز : هنا يمكن أن يكون الربط في أي نقطة أثناء إنجاز البرنامج والمثال الأكثر وضوحاً في هذه الحالة هو ربط المتغيرات مع قيمها أثناء عملية الإسناد .

٢- **زمن الترجمة Compile time** : هنا يمكن أن نميز نوعين هامين من عمليات الربط أثناء الترجمة :

آ- عمليات الربط التي تتم عن طريق المبرمج : يعمد المبرمج أثناء كتابة البرنامج إلى وضع عمليات الربط الخاصة به في البرنامج ، مثلاً أسماء المتغيرات ، أنواع المتغيرات ، ... الخ ، التي تمثل عمليات الربط أثناء الترجمة . ويعطي مترجم اللغة بعد عمليات الربط السابقة البرنامج الناتج (object program) .

ب- عمليات الربط التي تتم عن طريق المترجم : تتم بعض عمليات الربط عن طريق المترجم دون تدخل مباشر للمبرمج ، مثلاً عمليات ربط المتغيرات مع مواقع التخزين في الذاكرة ، وعمليات إنشاء متحولات بيئية للمتحولات الوسيطة وغيرها ...

٣- **زمن تطبيق اللغة Language Implementation time:** يمكن أن يكون هناك إختلافات متعددة بين اللغات في تطبيق العناصر الأساسية ، مثلاً حددت التفصيلات المرتبطة مع تمثيلات الأعداد والعمليات الرياضية عن طريق العمليات الرياضية وتمثيلات الأعداد المبنية في البنية المادية للحاسوب Underlying hardware computer ، وإذا استخدم برنامج ما صفات لغة برمجية ثبتت في وقت تطبيقها لن ينفذ بالضرورة على تطبيق آخر لحاسوب آخر لنفس اللغة والأكثر من ذلك يمكن أن ينفذ ويعطي نتائج مختلفة .

٤- **زمن تعريف اللغة Language definition time :** حددت معظم بنى لغات البرمجة في زمن تعريف اللغة ، مثلاً أشكال العبارات الممكنة ، أنواع بنى المعطيات ، بنى البرنامج ... الخ ، حددت عادة في زمن تعريف اللغة .

٣-٢- مثال توضيحي :

لندرس الآن المثال التوضيحي التالي : بفرض لدينا العبارة البرمجية التالية :

$$X := X + 10$$

ولنفرض ظهور هذه العبارة في برنامج ما كتب بلغة ما ولنبحث في عمليات الربط وأزمنة الربط على الأقل بالنسبة لعناصر العبارة السابقة .

١- مجموعة الأنواع المتاحة للمتغير X : إن للمتغير X في العبارة السابقة نوع معطيات مرتبط معه ، مثلاً نوع حقيقي ، نوع صحيح ، أو بولياني . وعرفت مجموعة الأنواع المتاحة للمتغير X في وقت تعريف اللغة مثلاً الأنواع الحقيقية ، الصحيحة ، البوليانية ، المجموعة ، المحرفية ... الخ ، وتسمح اللغة لكل برنامج أن يعرف أنواع جديدة ، كما هو الحال في لغة باسكال مثلاً ، عندئذ تثبت مجموعة الأنواع المتاحة للمتغير X في وقت الترجمة .

٢- نوع المتغير X : أي ما هو نوع المعطيات الذي سوف يرتبط مع المتغير X ، يحدد ذلك عادة في وقت الترجمة من خلال تصريح واضح في البرنامج كما هو في لغة باسكال:

X: Real ;

في لغات أخرى ، مثل لغة APL ولغة LISP يحدد نوع المعطيات في وقت التنفيذ من خلال إسناد قيمة للمتغير X . يمكن أن تشير X في مثل هذه اللغات إلى متجهة في مرحلة ما وإلى عدد صحيح في مرحلة أخرى لاحقة في نفس البرنامج .

٣- مجموعة القيم المتاحة للمتغير X : إذا كان X له نوع معطيات حقيقية (real) ، تكـــون قيمته عندئذ في أي نقطة أثناء إنجاز البرنامج واحدة من مجموعة البيتات الممثلة للأعـداد الحقيقية . تحدد مجموعة القيم الممكنة للمتغير X عن طريق الأعداد الحقيقية التي مثلـــت في الحاسوب والتي هي عادة مجموعة الأعداد الحقيقية الممثلة في البنية المادية للحاسوب. وبالتالي تحدد مجموعة القيم المتاحة للمتغيرX في وقت تطبيق اللغة وينتج عن اختـــلاف التطبيق مجالات مختلفة في القيم المتاحة للمتغير X .

٤- قيمة المتغير X : تحدد قيمة معينة للمتغيرX في أي نقطة أثناء إنجاز البرنامج وتحـــدد هذه القيمة أثناء وقت التنفيذ من خلال إسنادX . يغير الإسناد X := X + 10 عمليات الربط للمتغير X ، فيغير القيمة القديمة بقيمة جديدة هي أكبر بعشرة من القديمة .

٥- تمثيل الثابت 10 : هناك تمثيلين للعدد الصحيح ١٠ ، فهو يمثل كثابت فـــي البرنـــامج يستخدم الشريط ١٠ ويمثل أثناء زمن التنفيذ كمتتالية من البيتات . ويتم اختيـــار التمثيـــل العشري في البرنامج أثناء زمن تعريف اللغة ، بينما يتم اختيار متتالية من البتات لتمثيـــل عشرة في زمن التنفيذ أثناء زمن تطبيق اللغة .

٦- خصائص المعامل + : لنبحث في أزمنة الربط للصفات المختلفة للمعامل + في العبـــارة السابقة . يتم اختيار الرمز+ لتمثيل عملية الإضافة أثناء زمن تعريف اللغة ، ويسمح عادة لنفس الرمز + أن يمثل إضافة أعداد حقيقية ، إضافة أعداد صحيحة ، إضافة أعداد عقدية ... الخ . ويتم تحديد ماهية العملية التي تستخدم المعامل + في اللغات التي تعتمـــد علـــى الترجمة في زمن الترجمة . وتدعى التقنية المتبعة لتخصيص الربط بتقنية " النوع حســـب المتغير Typing mechanism for variables " . ويعني هذا إذا كان X من النوع الصحيـــح عندئذ يمثل الرمز+ في X + 10 إضافة أعداد صحيحة أما إذا كانت X من النوع الحقيقـــي فإن + تمثل إضافة أعداد حقيقية وهكذا ...

باختصار : من أجل لغة مثل لغة باسكال ، يحدد الرمز+ مجموعة من عمليات الإضافة فـــي وقت تعريف اللغة ، وتعرف كل عملية إضافة في المجموعة في وقت تطبيق اللغة ويحدد كـل استخدام للرمز+ في البرنامج عملية إضافة أثناء زمن التنفيذ . تمثل مجموعة عمليات الربـــط السابقة اختيار واحد من عمليات الربط الممكنة وأزمنة الربط باختلاف لغات البرمجة .

السؤال الذي يدور دائما في ذهننا : هل تحدث عمليات الربط في وقت الترجمة أم في وقـــت التنفيذ ؟ ليست كل لغات البرمجة متساوية في حل المشاكل فمثلا : بفرض أننا بحاجـــة لحـــل مسألة برمجية ما إلى معالجة متجهات متعددة من الأعداد . نحتاج في لغة سنوبول٤ لحل هذه

المسألة إلى الكثير من المتجهات الضخمة والعمليات الرياضية التي معظمها لن يكون مطلوبا إذا تم الحل بلغة فورتران وإذا بحثنا عن السبب نجد أنه تتم معظم عمليات الربط في لغة سنوبول٤ في وقت التنفيذ (وذلك بعد المقارنة بين صفات لغة سنوبول٤ ولغة فورتران) بينما تتم عمليات الربط نفسها في لغة فورتران في زمن الترجمة . وهكذا يتم في لغة سنوبول٤ في وقت التنفيذ إنشاء وحذف عمليات ربط متعددة بينما تتم نفس عمليات الربط في لغة فورتران مرة واحدة أثناء الترجمة وتترك القليل من عمليات الربط إلى وقت التنفيذ ، وبالتالي يكون البرنامج بلغة فورتران أكثر كفاءة (قوة) منه بلغة سنوبول٤ . وهنا نستطيع القول أن لغة ما مثل لغة فورتران التي تجعل معظم عمليات الربط تصبح في وقت الترجمة – أي بشكل مبكر أثناء معالجة البرنامج – أنها ذات ربط مبكر Early binding ، ونقول عن لغة مثل لغة سنوبول٤ والتي تقوم بتأخير عملية الربط حتى وقت التنفيذ بأنها لغة ذات ربط متأخر Late binding .

أما فوائد وأضرار عمليات الربط المتأخرة أو المبكرة فهي في صراع دائم حول مفهوم القوة والمرونة . حيث يعتمد تصميم اللغات التي تتطلب حل لمسألة ما بشكل قوي مثل لغات فورتران وباسكال وكوبول على أن يكون معظم عمليات الربط في وقت الترجمة ، وتؤخر معظم عمليات الربط إلى وقت التنفيذ إذا طلب الحل بشكل اكثر مرونة للتعامل مع المستخدم كما هو الحال في لغات سنوبول٤ وLisp ، ويكون لدينا خيارات متعددة تسمح باختيار أزمنة الربط في لغة مصممة من أجل القوة والمرونة معا لحل المسائل مثل لغة Ada أو لغة PL / I.

رابعا–تصميم وتنفيذ المتحولات وبنى المعطيات التقليدية:

إن أي برنامج (دون الاهتمام باللغة المستخدمة) هو عبارة عن مجموعة من العمليات طبقت على معطيات محددة بتتالي محدد ، وتختلف لغة عن أخرى في أنواع المعطيات المستخدمة ، أنواع العمليات المتاحة للاستخدام ، والتقنية المجهزة للتحكم بتتالي العمليات المطبقة على المعطيات .

١–**غرض المعطيات** : نشير إلى مجموعة من المعطيات الموجودة في البنية المادية للحاسوب أثناء التنفيذ **بغرض المعطيات Data Object .**

يعرف المبرمج بعض أغراض المعطيات مثل المتغيرات الثوابت المصفوفات الملفات،.... الخ، بحيث يعرفها المبرمج بشكل واضح ويعالجها من خلال التصريحات والعبارات في البرنامج . ويعرف النظام أغراض معطيات أخرى وهي عبارة عن أغراض معطيات