الدرس الثالث:

والان لنعود الى الحديث النظري.....

ولنفهم مفهوم الغرضي التوجه:

تطور النظم البرمجية :

IBM من الشركات الرئدة في مجال تطوير النظم الحاسوبية بحيث كانت تصمم جهاز حاسب الي وتضع فيه نظام تشغيل صغير سمي (**o,s**)

Operating system وتطويره وهو نظام يستخدم الموجودة لدى الحاسب ويتحكم بها وذالك بهدف الحصول على اكبر قدر ممكن من الكفاءة

في ذالك الوقت ظهر طالبين من الجامعات صمما نظام تشغيل خاص بهما عرف باسم نظام **DOS..... Ibm**تبنت هذا النظام واقمت على تشجيعهما....

كانت البرمجة انذاك بواسطة الاسيمبلي كان العمل صعبا وطويل جدا حيث ان المبرمج كان يتعامل مع الذاكرة مباشرة بالوضع والجمع والحذف..... الخ فقد كان على المبرمج مثلا لطباعة كلمة ان يكتب 15 او 20 سطر برمجي لتنفيذ ذالك ونتيجة ذالك ظهرت الحاجة لايجاد طريقة جديدة وهي طريقة الدوال **procedure** حيث وضعو اكثر من امر **insturtion** بمكان واحد فقط ومن هنا نشا مفهوم الكتبات **Library** والتي تضم اكثر من دالة وقد عرفت بالبرمجة التركيبية **Structured Programming** اذ كان ينظر الى البرامج على انها سلسلة من الاجراءات التي تستجيب للبيانات . والاجراء ماهو الا مجموعة من الاوامر المحددة التي يجري تنفيذها واحدا تلو الاخر . كان يتم فصل البيانات عن الاجراءات وكان جوهر البرمجة يكمن في معرفة الاجراءات التي استدعت اجراءات اخرى ومعرفة البيانات التي تم تغيرها ونتيجة لذالك برزت في فترة الستينات الكثير من العقبات القاسية في تطوير برمجيات ضخمة مما كان يؤدي الى تاخر في مواعيد إنجاز الاعمال المطلوبة والى تكاليف مالية كبيرة تتجاوز الميزانيات المخصصة والى الخروج بمنتج برمجي غير موثوق مما دعى الى التاكد من ان عملية التطوير البرمجي هي عملية اكثر تعقيدا مما كانو يغتقدون سايقا

وكان ذالك يسبب الكثير من الارباك والضياع والارهاق وخاصة للبرامج والنظم الضخمة وقد اثمرت جهود البحث العلمي في تلك الفترة على تطوير اسلوب البرمجة المهيكلة والذي هو عبارة عن طريقة كتابة برامج اكثر وضوحا من مثيلاتها الغير مهيكلة واكثر قابلية للفحص والتصحيح والتعديل وقد صممت لغة الباسكال اساسا لتعليم طلاب الجامعات على البرمجة المهيكلة في المؤسسات التعليمية والاكاديمية

اذ ان الفكرة الاساسية المبنية عليها البرمجة الهيكلية تماثل في بساطتها فكرة فرق تسد ....اذ يمكن تخيل البرنامج على انه مركب من مجموعة من مهام واي مهمة اعقد من ان توصف بسهولة وسيتم تقسيمها الى مجموعة من المهام الاصغر الى ان تصبح المهام الصغيرة بالدرجة الكافية لفهمها واستيعابها بسهولة

على سبيل المثال ستجد ان حساب متوسط مرتبات الموظفين باحدى الشركات مهمة معقدة ومع ذالك بمقدورك تقسيم هذه المهمة الى المهام التالية

1. معرفة مرتب كل موظف
2. حساب عدد الموظفين
3. جمع كافة المرتبات
4. تقسيم مجموع المرتبات على عدد الموظفين ومن الممكن تقسيم مهمة جمع المرتبات الى ثلاث خطوات التالية:

1- احضار السجل الخاص بكل موظف

2- الوصول الى قيمة المرتب

3- اضافة المرتب الى اجمالي المرتبات

4-احضار سجل الموظف التالي

وهكذا يتم تقسيم المهام المعقدة الى مهام فرعية اصغر منها حتى تصبح المشكلة يسيرة ويتم حل أي مسالة برمجية من خلال تنفيذ سلسلة من الافعال وفق ترتيب معين نطلق تسمية الخوارزمية **Algorithm** على اية اجرائية لحل مسالة ما تقوم بتحديد

1. الافعال الواجب تنفيذها
2. الترتيب الواجب اتباعه من اجل تنفيذ الافعال السابقة

لناخذ على سبيل المثال مانسميه خوارزمية الاستيقاظ والصحو **Algorithm rise-and-shine** تحدد هذه الخوارزمية ترتيب الافعال التي يقوم بها شخص منذ نهوضه من السرير حتى ذهابه الى العمل

1. النهوض من السرير
2. خلع لباس النوم
3. ارتداء ملابس اخرى
4. اخذ حمام صباحي
5. تناول الفطور
6. الذهاب الى العمل

في هذه الحالة يذهب الشخص المنفذ للخوارزمية الى العمل وهو مبتل تسمى عملية تحديد وترتيب التعليمات ضمن برنامج ما بالتحكم الموافق للبرنامج .....

لغة الخوارزمايات:

بالحقيقة انها تسمية مغلوطة اذا انها ليست لغة بمفهومها الكامل فهي عبارة عن لغة مصطنعة وغير متعارف عليها تساعد المبرمجين على تطوير الخوارزميات تشبه لغة الخوارزميات اللغة الإنكليزية المتداولة فهي لغة مبسطة ومفهومة ولكنها ليست لغة برمجية فعلية

اذا لايمكن تنفيذ البرامج المكتوبة بلغة الخوارزميات على الحاسب لكنها تساعد المبرمجين على التفكير ببرامجهم قبل محاولة كتابتها باية لغة اخرى كالدلفي مثلا .....

بنى التحكم:

يتم عادة تنفيذ تعليمات برنامج ما تعليمة تلو الاخرى حسب ورودها في نص البرنامج نسمي هذا الاسلوب بالتنفيذ التسلسلي **Sequential execution** ويمكن ان تاتي في موضع اخر من نص البرنامج ومختلف عن موضع التعليمة التالية لها مباشرة نسمي هذ ه العملية بعملية نقل التحكم **Transfer of control** لقد اصبح واضحا منذ فترة الستينات ان الاستخدام الغير مقيد لعمليات نقل التحكم هو المسبب الرئيسي للكثير من الصعوبات التي تواجه مجموعات تطوير البرامج فلقد تم توجيه اصبع الاتهام الى تعليمة **goto** التي تسمح للمبرمج بنقل التحكم الى أي نقطة يريدها ضمن البرنامج يمكن ان نقول اذا ان تعبير البرمجة المهيكلة يصبح تقريبا مكافئا لتعبير ازالة تعليمة **goto**في نص البرنامج .

ولقد برهنت الابحاث التي قام بها كل من **Bohm**و **Jacopini** بانه من الممكن كتابة برامج بدون استخدام اية تعليمة **goto** ولكن لم يجري الحديث عن اسلوب البرمجة المهيكلة بشكل جدي الا في فترة السبعينات لقد اظهر هذا الاسلوب نتائج مبهرة حيث لاحظت مجموعات تطوير البرامج وجود امكانيات لخفض زمن تطوير البرامج ولتسليم الاعمال وانجازها في المواعيد المحددة اضافة الى تزايد فرص انجاز المشاريع وفقا للميزانيات المرصودة مسبقا يعود الفضل الى ذالك الى كون البرامج المهيكلة برامج واضحة وسهلة الفحص والتعديل بالاضافة الى تزايد امكانية خلوها من العيوب بالدرجة الاولى

برهنت تلك الابحاث بانه من الممكن كتابة البرامج باستخدام ثلاثة بنى للتحكم فقط وهي البنية التسلسلية **Sequential executure** وبنية الاختيار **Selection executure** وبنية التكرار **repetition executure**

والسؤال الذي يطرح نفسه :ماهو علاقة ذالك الحديث بعنوان الموضوع...؟؟؟؟

اقول انتظر ياصديقي لانهي كلامي لتبيان سبب ظهور مفهوم الغرض ....

ففي البرمجة الاجرائية تم حل العديد من المعضلات التي كانت تواجه المبرمجين ولكن ليس جميعها .....؟؟؟؟

اذ كان يوجد احباط كبير تعاني منه هيئات تطوير البرمجيات وخاصة تلك التي كانت تقوم بتطوير برمجيات كبيرة

اذ تكمن المشكلة الرئيسية للبرمجة الاجرائية في قصور هذا الاسلوب على عكس العالم الخارجي ضمن الوحدات البرمجية المختلفة المؤلفة للنظام المراد تطويره يؤدي ذالك الى خفض القدرة على اعادة استخدام هذه الوحدات

البرمجية مع برامج اخرى ويجب على المبرمج في الكثير من الحالات اعادة كتابة هذه الوحدات عند البدأ بتطوير برنامج جديد

يشكل ذالك اضاعة للوقت والمال اثناء محاولة المبرمجين في كل مرة اعادة اختراع الدولاب اما عند التعامل مع تقنية الغرض تسمى الوحدة البرمجية بالصف **class** واذا كانت هذه الوحدة مصممة جيدا فان ذالك يساعد على اعادة استخدامها مستقبلا .....

وتشير بعض المؤسسات والمنظمات الى ان الاستفادة من قابلية اعادة الاستخدام ليست الفائدة الاساسية الاولى التي نحصل عليها بالاسلوب الغرضي التوجه وانما الفائدة بالقدرة على فهم واستيعاب الانظمة المطورة وينعكس ذالك بالقدرة على صيانة وتطوير وتدقيق البرمجيات عند الحاجة بكل سهولة وتنظيم .ونلمس ذالك اذ علمنا ان نسبة 75 % من كلفة بناء البرمجيات ليست مرتبطة بجهود التطوير الاولى وانما بمتابعة التقيم والصيانة التي تمر بها تلك البرمجيات طيلة فترة استخدامها .

تكمن الميزة الرئيسية في انشائك لنصوصك البرمجية في معرفتك الجيدة لها وقدرتك على تفحصها لكنك ستكون قد ضيعت وقتا طويلا وبذلت جهدا كبيرا لتطوير هذه البرامج .....؟؟؟؟؟

تهدف البرمجة غرضية التوجه **oop** الى تضمين المعطيات والتوابع ضمن مكونات تدعى الصفوف وهما مرتبطان ببعضهما البعض

لنشبه الصف بتصميم معماري على الورق تم بنائه وتنفيذه اذا يستطيع أي معماري ان يقوم ببناء بناء انطلاقا من التصميم الاول بدلا من تصميم واحدا جديدا او بالتعديل عليه وادخال اضافاته التي يريدها وبنفس الطريقة يمكن للمبرمج ان ينشئ اغراضا انطلاقا من الصفوف يمكن استخدام الصف عدة مرات لانشاء العدد الذ ي نريده من الاغراض المشتقة من نفس الصف

ارجو انه تكون مفهوما عاما واضحا لخد ما عن البرمجة الاجرائية او غرضية التوجه اما عن غرضية التوجه في الدلفي فسوف نتحدث عنه بالتفصيل فيما بعد .

1. تحدثنا في هذا الفصل عنالمفاهيم العامة للبرمجة الاجرائية وماكانت الحاجة اليها
2. تحدثنا عن مفهوم العام للبرمجة غرضية التوجه
3. ومر معنا ايضا بعض الشروح العامة للغة الخوارزميات

اريد ان انوه ان هذا الدرس اعتبره ركيزة اساسية في توطيد الفهم الحقيقي للبرمجة بصورة عامة