أنماط البرمجة الرئيسية

مدخل للبرمجة

سعيد عبد الله

تصميم الغلاف:

تم تصميم الغلاف عن طريق الذكاء الاصطناعي.

نوع الطبعة: الطبعة الإلكترونية

للإستفسارات أو غيرها يمكن التواصل مع ناشر هذا المصنف وهو الكاتب نفسه.

Saeedal7mmadi@gmail.com

الفهرس

ما قبل المقدمة

إن النشاط البشري سبب رئيسي في التغييرات الحاصلة على كوكبنا، والصراعات البشرية اللا نهائية تعتبر أكثر النشاطات سيادة على مر العصور مع اختلاف المراحل. أما تصنيفات القوة والضعف فقد اختلفت مع تطور الصراعات؛ لتتجه من قوة تقاس بعدد الأيدي العاملة إلى قوة تقاس بكمية المعرفة ونوعية العقول، أي أننا ننتقل من مرحلة حروب بالسلاح إلى حرب معلوماتية فكرية.

إن سعي العالم الحثيث نحو امتلاك المعرفة أو إنتاجها يعتبر سلوك بديهي في خضم التطور المعلوماتي السريع. وصناعة العقول هو الهدف الرئيسي العالمي الذي به تتحول الدول من مستورد إلى مصدر للأفكار والعلم. والوصول إلى هذا المستوى يتطلب خطة تعليمية مدروسة ومحكمة وممنهجة بطريقة احترافية تمسح فكرة الصندوق الذي عادة ما يكون المبدع خارجه.

في النصف الثاني من القرن العشرين، أصبحت أمريكا اللاتينية محط أنظار المثقفين في الأدب الكتابي المسمى بـ "اللاتينو أميركي". والمهم معرفته أن وصول أمريكا اللاتينية إلى هذا المستوى لم يأت إلا عن طريق حركة إصلاح التعليم التي ترجع إلى أواخر القرن التاسع عشر، وهذا يدل على أن نوعية التعليم تحدد نوعية الإنتاج الفكري.

وفي هذا السباق المعرفي الذي يخوضه العالم نجد أن هناك من يجلب المستقبل، متعديا حدود الغلاف الجوي، متخذا من حدود الفضاء اللا محدود حدودا لفكره؛ لينهي عصر الإحسان العلمي ويأخذ بالأيدي نحو التقدم العلمي المرجو؛ لنبدأ عصر علم جديد.

عند ذكر العلم نسترجع العصور الذهبية التي مرت بها الأمة، أما الآن نحن لا نسترجع، بل نصنع ملامح جديدة للعالم الحديث. لن نُرجع العصور المنصرمة، بل هناك زمن جديد بانتظارنا يطلق عليه "زمن المعرفة"، ستقوده الإمارات بعقول منتجة وفريدة من نوعها.

إن الحرب المعلوماتية لا تزال قائمة ولا نعلم متى ستنتهي، وما دام أن رأي العلم في نشاطات الإنسان على الأرض أنها لا تؤثر على الفضاء الخارجي، فهذا يأخذنا إلى التساؤل: ماهي المرحلة القادمة؟

سعيد عبد الله

٩ رمضان ١٤٤٥ هـ

3/19/2024

المقدمة

علم البرمجة هو أحد العلوم الأكثر انتشارا في السنوات العشر الأخيرة، ويعتبر من المجالات سريعة النمو، وتعتبر أيضا الحركة التطويرية في أساليبها المتنوعة سريعة جدا مقارنة بالمجالات الأخرى. وإنه بالرغم من العقبات التي مر بها هذا المجال منذ نشأته إلا أنه أثبت مقدرته في التأثير على النمو البشري بشكل ملحوظ.

إن تاريخ البرمجة قديم جدا، ودراساته المبدئية التي شكلت ثورة علمية آنذاك تعتبر اليوم بسيطة، لكنها كانت مدخلا لما وصلت إليه التقنية المعلوماتية، وكانت حجر الأساس في الثورة التي نشهدها اليوم. وقد انتقلنا اليوم من الزمن الأول الذي وإن أسعفني التعبير "زمن البرمجة البدائية" –التي كانت تعتمد على اللغة الثنائية، وكانت آنذاك مجال نادر وصعبا– إلى "زمن الذكاء الاصطناعي وتعلم الآلة " حيث أصبح المجال متاح وسهل، وما زال العلم يحاول تبسيط هذا المجال؛ ليجعل التواصل بين الإنسان والألة أكثر سهولة ويسر.

وإن سعي الإنسان للوصول إلى آلة تتعلم ذاتيا من الخطأ والصواب يجعلنا نصل إلى نقطة نبدل فيها الآلة بالإنسان، ويتجه الاعتماد كليا إلى ما يسمى بالرجل الآلي؛ لأنه بكل بساطة سيقوم بالأعمال دون كلل أو ملل أو توقف، وهذا ما يحاول العلم إنجازه.

ذكر في NewScientist حول الآلات التي تفكر أن أحد آباء ثورة الحساب المدعو "جون فون نويمان" قد قال في عام 1948: "إنكم تصرون على أن هناك شيئا لا يمكن للآلة أن تفعله؛ فإذا أخبرتموني بدقة ما الذي لا تستطيع الآلة أن تفعله، عندها يمكنني دائما أن أصنع آلة تفعل هذا الأمر بالضبط". ويبين هذا أن ثورة الذكاء الاصطناعي ماهي إلا مسألة وقت لتتغلب على البشر بالفعل. وقد تعمق الباحثون في هذه المجالات كثيرا حتى بدأوا بدراسة عمل الدماغ البشري في محاولة لصناعة دماغ صناعي، ولكن هذا لا يعقل وبالتأكيد جميع محاولاتهم بائت بالفشل؛ لأن الدماغ البشري معقد لدرجة الإعجاز وهذه قدرة الله عز وجل، وإن هذا لا ينطبق فقط على الدماغ، بل على خلق الإنسان. قال الله تعالى في سورة الذاريات الآية (21) "وَفِي أَنفُسِكُمْ ۚ أَفَلَا تُبْصِرُونَ".

إن استدلالي بهذه الآية وبيان إعجاز خلق الإنسان الذي لا يخفى على أحد ماهي إلا للدلالة على أن الإنسان مهما جد واجتهد في مضاهاة خلق الإنسان ماهي إلا مضيعة للوقت والجهد.

وفي الجهة الأخرى قال بعض العلماء والباحثين في هذا المجال أن معرفة خوارزميات الذكاء الاصطناعي بشكل أعمق يمكننا من فهم الذكاء الإنساني وقد خالفهم علماء آخرين في هذا الرأي بقولهم إن ذكاء الآلة يكمن في البيانات وليس الخوارزميات.

لا أود التطرق إلى مثل هذا، وليس هذا بمقال أستطرد فيه حول تاريخ ذكاء الآلة، ولكن ما ذكرته فقط لأبين مدى أهمية البرمجيات في العلوم الصناعية منذ القدم وأن فكرة عمل البرمجيات كانت ومازالت تتطور باستمرار بحيث تتماشى مع العجلة الصناعية وتساعد في تسريع النمو الصناعي، وساهمت في السنوات الأخيرة في تسهيل حياة الناس وتسريع كثير من الأعمال، ونتجه حاليا إلى الهوة إذا لم تتمكن البشرية من السيطرة على الانغماس البطيء غير الارادي في هذا.

في هذا الكتاب سنتعلم سويا حول مفهوم وأساليب البرمجة وأنماطها وإذا أمكن سنتطرق إلى بعض الأمثلة في محاولة لتسهيل الفهم.

أسأل الله أن يبارك في هذا العمل وأن يفيد به إخواني القراء.

ماهي البرمجة

يحتم علينا قبل الشروع في الشرح والتعليم في هذا المجال أن نشرح ونعلل معنى البرمجة وأنواعها، أما فروقها سنتطرق إليه في الصفحات التالية بشكل يمكننا من فهمها فهما يتيح لنا التعمق فيها فيما بعد.

عند البحث عن مفهوم البرمجة وجدت العديد من التعريفات غير المقنعة بالنسبة لي، فعمدت لسؤال الذكاء الاصطناعي حول مفهومها فذكر لي ثلاث تعريفات من ثلاث مصادر مختلفة ولم أكن مقتنعا تماما حول جميع التعريفات والمفاهيم التي قرأت أما ما كنت متفقا فيه وكان متشابها بين جميع المفاهيم هو أن البرمجة عملية كتابة تعليمات للحاسوب لحل مشكلات أو تنفيذ أوامر ووظائف معينة باستخدام لغة برمجية محددة. وهذا يأخذنا إلى سؤال أخر أكثر عمقا، ما هو مفهوم لغة البرمجة؟

ما تنفق عليه هو أن لغة البرمجة هي التي يتم كتابة التعليمات والأوامر باستخدامها، ولكن ما نود معرفته، هل لها شكل أو صفة معينة مشتركة بين جميع اللغات التي تستخدم في هذا الشأن؟

الجواب هو نعم، لغة البرمجة يجب أن تكون مفهومة ومقروءة ويمكن استيعابها حتى نتمكن من التواصل مع الآلة وهي عبارة عن لغة وسيطة بين الآلة والإنسان؛ لأن المعروف في علم الحاسوب أن دماغ الآلة والذي يطلق عليه بالمعالج لا يستوعب إلا لغة واحدة فقط وهي اللغة الثنائية المكونة من رقم **0** و**1**، وأما ما بين المقروء والغير مقروء فهناك ما يطلق عليه المترجم البرمجي المعروف في مناهج العلوم الحاسوبية ب **compiler**. وإن ما ينتج لنا بعد عملية

الترجمة البرمجية ما يسمى ب البرنامج، وهو بكل بساطة مجموعة من الأوامر والمتغيرات والدوال تنتج لنا حلقة مستمرة أو غير ذلك لمهمة أو مجموعة مهام.

وكما ذكرنا آنفا أن العلوم الحاسوبية تتطور باستمرار وعلوم البرمجة تحديدا متغيرة ومتنوعة الأنماط، ولكنها ثابتت الأسس والمعايير، وقد أُختلف في عدد هذه الأنماط وأُختلف في عدد الأساسي منها، ولكن يمكننا قول إن هناك ثلاث أنماط أساسية وهي:

البرمجة التوجيهية (**Procedural Programming**)

البرمجة الكائنية (**OOP Object–Oriented Programming**)

البرمجة الوظيفية (**Functional Programming**)

ومن خلال دراستي للبرمجة تعلمت وعملت على **C#**، و**java**، و**python** وجميعها كانت تستخدم نمط البرمجة الكائنية بشكل أساسي، وأما في مجال عملي فقد عملت وتعلمت لغة **dart** التي تم تطويرها من شركة **google**، وتعتبر هذه اللغة ذات سمات جامعة بين الأنماط الثلاثة.

سننطلق سويا في رحلة للتعرف والتعلم حول هذه الأنماط الثلاثة، وسآخذ على عاتقي محاولة التبسيط حتى يتسنى لي ولكم فهم كامل حولها، وحتى يكون لك هذا الكتاب مرجع تستطيع الاستعانة به في أي زمان ومكان حتى يغير العلم هذه الأساسيات الثلاث.

البرمجة الكائنية أو الشيئية

(OOP Object–Oriented Programming)

يعتمد هذا النمط البرمجي على تقسيم البرنامج إلى وحدات تحتوي على بيانات متغيرة أو ثابتة، وتتميز نظرة المبرمج الذي يتخذ هذا النمط في تحويل جميع الأشياء من حوله إلى كائنات أو وحدات تحتوي على خصائص وسلوكيات غير متكررة، وإذا نظرنا عن كثب في هذا المفهوم فسنجد أن الكائن أو الشيء الذي يطلق عليه في علم البرمجة (**Object**) عبارة عن قسمين:

- خصائص
- سلوكيات

وهذان القسمان يتوفران في جميع الكائنات من حولنا كالسيارات، والبشر، والحيوانات، وغير ذلك، حيث إن خصائص وأفعال كل منهم تختلف عن نظيراتها من الكائنات، ولتسهيل عملية الفهم سنعمد إلى تعريف هذان المفهومان حتى يتسنى لنا استيعاب كل منهما.

الخصائص: هو كل ما يمكن أن نصف به الكائن، كأن نقول هذه السيارة لونها كذا، أو هذا الشخص طوله كذا، أي أن طول الشخص أو اسمه أو وزنه ... الخ جميعها خصائص تميز شخص معين، ولون السيارة ووزنها ونوعها ...الخ خصائص تميز سيارة بعينها.

أما السلوكيات: فهي الأفعال التي تميز هذا الكائن، كأن نقول هذا الشخص يجري أو يمشي ...الخ، وأما السيارة فيمكننا القول عن سلوكها، تشغيل، إيقاف، انعطاف ...الخ.

وعلى حسب علمي واطلاعي حول المفاهيم الرئيسية لهذا النمط وجدت العديد من المقالات المختلفة بتحديد هذه المفاهيم وعددها وأغلب ما قرأت كان يصنفها إلى أربعة مفاهيم رئيسية،

لكن توصلت بعد المطالعة إلى تقسيمها إلى ثلاثة مفاهيم، وأظنها كافية لأن تلخص لنا المميز في هذا النمط، وهي التي سنشرحها في هذه الصفحات القليلة القادمة

- التجريد (**Abstraction**)
- التوريث (**Inheritance**)
- الإخفاء أو التغليف (**Encapsulation**)

وذكري لهذه الثلاثة مفاهيم بدلا من غيرها هو لأنها متصلة بعملية البرمجة بخلاف ما اطلعت عليه، ولأني في هذا الكتاب أركز على شرح ما هو متعلق بالشيفرة وترتيبها وليس ما هو متعلق بتصميم صفحة المستخدم أو كما يطلق عليه في العلوم الحاسوبية ب **User Interface** (**UI**).

قبل الخوض في هذه المفاهيم يجب فهم معنى "**class**" ونستطيع أن نطلق عليه باللغة العربية "فصل"، ووظيفته الأساسية تخزين المعلومات والبيانات حول فئة معينة من الكائنات التي تحمل خصائص وسلوكيات متشابهة، كأن يكون لك متجر يحتوي على عدة فئات من المنتجات وفي كل منها عدة أشكال من المنتجات تحتوي نفس السمات.

واختيار شرح وظيفة "الفصل" الأساسية قبل الشروع بتفاصيل المفاهيم الثلاثة الرئيسية؛ لنوضح مدى اتصال هذا المفهوم فيما نود شرحه وتعليله فيما بعد.

التجريد (Abstraction)

نستطيع أن نعرف التجريد على أنه عملية يتم فيها تسهيل استخدام الشيفرات الأساسية في عملية البرمجة حتى تكون سهلة الوصول والإدارة، ولفهم أعمق حول هذا المفهوم، لنتخذ عملية برمجة نظام **windows** مثال:

لو فرضنا أن هذه العملية أنجزت بكتابة أكثر من مليون سطر من الشيفرة –وهي أكثر من ذلك بكثير– ولم يكن الفريق قد عمل على تجريد الشيفرات، وقرر المبرمجون إضافة تحديث معين على إحدى المميزات الموجودة في النظام، تصور الطريقة والوقت والجهد الذي سيقضيه الفريق في إضافة التحديث. ولو فرضنا أن التحديث سيكون بعد السطر رقم **500,000**، كيف سيتعامل الفريق في حال تصحيح الأخطاء (**debug**).

وأنا أكتب هذا المثال لم أستطيع تخيل الصعوبة التي يواجهها هذا الفريق البرمجي في إضافة **10** إلى **20** سطر من الشيفرة في هذا الكم الهائل من السطور.

إذا لتجنب الجهد الإضافي المبذول وتوفير الوقت للفريق حتى يحقق كم أكبر من الإبداع في البرمجة، يقوم هذا النمط البرمجي بتوفير فكرة التجريد للشيفرة، أي فصل الشيفرات إلى مجموعات مترابطة تحقق غاية، حتى يسهل التعامل معها. كأن يتم تقسيم شيفرة نظام **windows** إلى عدة وظائف (**functions**)، ومثال على ذلك:

إنشاء وظيفة في النظام يطلق عليها (إظهار الإشعارات)

ShowNotifications()

وتكون هذه الوظيفة عبارة عن كتلة تحتوي على الشيفرات الأساسية لإظهار الإشعار، حيث أن هذه الكتلة تسهل عملية استدعاء هذه المجموعة من الشيفرات والتعامل معها.

بمعنى أدق من خلال المثال السابق، بدلا من أن يبحث المبرمج عن الشيفرة حتى يتمكن من إضافة تحديث، يتوجه بشكل مباشر إلى الوظيفة المعنية في البرنامج.

التوريث (Inheritance)

يعتبر التوريث صفة موجودة في جميع الكائنات في العالم الحقيقي، وتعريفه هو أن يرث كائن بعض أو جميع الصفات من الأم أو الأب مع وجود صفات خاصة تميزه، كأن يرث الكرم والشجاعة، ولكنه انفعالي وعجول.

وكما عرفنا التوريث في الحياة الحقيقية، فهو ينطبق على عالم البرمجة، فيكون إمكانية بناء صنف جديد بناء على صنف تم بناءه سابقا، والمقصود منه في عالم البرمجة ليس التكرار، بل لترتيب بيروقراطية[1] الشيفرة. مثال على ذلك:

إذا كان لديك معرض لبيع جميع أنواع المركبات وقررت إنشاء نظام إلكتروني يسهل عملية البيع والشراء والصيانة –إذا كنت تقدم هذه الخدمة– فماهي الأصناف التي يمكن أن تعمل عليها في النظام؟

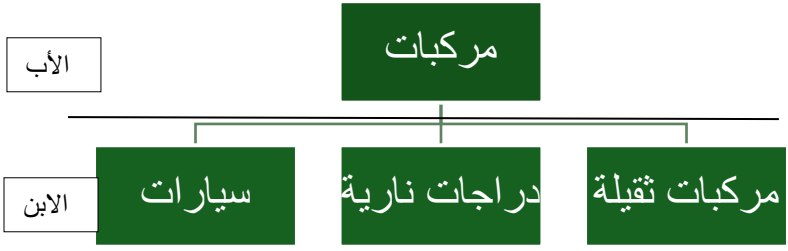
- **سيارات**
- **دراجات نارية**
- **مركبات ثقيلة**

ولو فرضنا أن هذه الأصناف الثلاثة هي الأصناف المرادة، فكيف سيكون شكل الشيفرة؟

[1] البيروقراطية: مفهوم يستخدم في علم الاجتماع والعلوم السياسية يشير إلى تطبيق القوانين بالقوة في المجتمعات المنظمة، ويعتمد على توزيع المسؤوليات بطريقة هرمية، والمقصود ب بيروقراطية الشيفرة أي الأسلوب الهرمي في الشيفرة.

من الأخطاء الفادحة في عملية بناء النظام أن يتم بناء الكائنات بشكل يدوي بدون الاستعانة ببناء "الفصول" (**classes**) خاصة إذا ما كان هناك كم هائل من البيانات التي تحتوي على أكثر من صفة أو سلوك، وبالإضافة إلى ما ذكرناه عن مفهوم "الفصل" في الصفحات الماضية واستخداماته، فهو يستخدم في ترتيب وإدارة البيانات الهائلة، ويسمح لها بالتوسع بدون أن يتم اجراء تحديث في الشيفرة عند أي إضافة، ويسهل عملية التخزين والاسترجاع.

أما عملية التوريث فهي تعتبر مهمة بقدر كبير إذا ما كان هناك عدة فئات لهم صفات وسلوكيات أساسية مشتركة، وفي المثال المذكور سيكون هناك "فصل" رئيسي يحمل صفات وسلوكيات المركبات الأساسية حتى يتسنى لنا توريثها إلى ثلاث فصول مختلفة كما هو موضح في الشكل التالي:



شكل **101**

في البرمجة يطلق على الفصل الرئيسي اسم "أب" لأن منه يتم اشتقاق الفصول الأخرى التي يطلق عليها اسم "أبناء"، كما عرفنا التوريث في الصفحات الماضية، والجدير بالذكر هو أن أي تغيير في الفصل الأساسي ترثه الفصول الأخرى مما يجعل الإدارة سهلة وبسيطة، بخلاف من يرى هذه الميزة من عيوب صفة التوريث، وبناء على رأيي الشخصي حول هذا فإن من

الطبيعي أن يتم توريث التغييرات ولا تعتبر من العيوب بل هذا يعتمد على قدرة المبرمج على بناء وإدارة الفصل الأساسي "الأب"، ومقدرته على إيجاد ما قد يكون مشترك فعلا بين الفصول الأبناء، وهذا يتيح له توسع مستمر بدون أي مشكلات تحول بين ذلك.

الإخفاء أو التغليف (Encapsulation)

عملية جعل المتغير مخفي لا يمكن الوصول إليه إلا عن طريق دوال[2] في نفس الفصل أو يتم تحديدها عن طريق المبرمج على حسب الأهداف والغايات، والهدف الأساسي من استعمال هذه الصفة في هذا النمط هو تمكين التحكم بالوصول إلى متغير معين؛ لتجنب الوصول غير المرغوب فيه والعبث به عن طريق فصول أخرى أو من خلال صفحة المستخدم.

ومن أمثلة حد الوصول إلى المتغير هو استخدام دوال **getter** و**setter**، حيث تسمح هذه الدوال للمبرمج من التحكم في الوصول للمتغير، على سبيل المثال قام المبرمج من جعل المتغير "رقم الهوية الوطنية" غير مخفي، وقام بعد ذلك باستخدام بعض الدوال التي تسمح باسترجاع وعرض المتغير على المستخدم من غير الحد من الوصول، وبدون وعي حول أخطار هذا الفعل، ونجد قسم آخر من المبرمجين يعمل على تغليف جميع المتغيرات رغم عدم الضرورة، والعجيب يصر على هذا الفعل بحجة جعل البرنامج أكثر أمانا.

إن معرفة المبرمج مهارة التعامل مع البيانات مطلوبة مع الاعتماد على الهدف والغاية؛ لتحديد طريقة عرض وتخزين المتغيرات والثوابت؛ لأن البرمجة عبارة عن بيانات.

[2] دوال جمع دالة وهي في لغة علوم الحاسوب تعني method

من الأمثلة الحية حول تغليف البيانات هو تطبيق **UAE PASS**، فقد استخدم المبرمجون فيه التغليف على درجتين:

الأولى: استخدام الدالة **getter** فقط عند عرض المستخدم بيانات الشخصية المتعلقة ب "رقم الهوية الوطنية"، و"تاريخ الميلاد" لكيلا يستطيع تغير أي معلومات من شأنها تعتبر وصول غير مرغوب فيه للمتغير، وتعتبر هذه المتغيرات غير قابلة للتغيير في المستقبل.

الثاني: استخدام الدالة **getter** و**setter** عند عرض المستخدم البيانات الشخصية المتعلقة ب "رقم الهاتف المتحرك"، و"البريد الإلكتروني"؛ لأنهما قابلان للتغير في المستقبل، وعدم استخدام **setter** قد يجعل الموضوع أكثر تعقيدا في المستقبل عند تغيير أحدهما – لك أن تتخيل حجم ما قد تواجهه حين إذ– وهذا يأخذنا لما ذكرنا آنفا حول مهارة التعامل مع البيانات حسب الهدف والغاية.

إن ما ذكرته حول الدوال المذكورة ما هو إلا لشرح مبدأ تقييد الوصول في عملية التغليف التي تعتبر من أهم الصفات التي قد تعطى للبيانات الحساسة غير القابلة للتغيير، وقد تحتوي هذه الصفة على دوال مختلفة تُنشأ على حسب الهدف والغاية من العملية.

يعتبر تجنب الدوال غير الضرورية والفصول كذلك من ذكاء المبرمج، وقد تختصر الوقت والجهد في انتاج وبناء التطبيقات والأنظمة الفعالة ذات الكفاءة العالية بخلاف زج الدوال والفصول غير المخطط لها؛ لينتج في النهاية أنظمة ثقيلة التشغيل وذات كفاءة متدنية، وفقد خاصية التوسع في البيانات والبرنامج بشكل مريح وسلس.

البرمجة التوجيهية

(Procedural Programming)

يعتمد هذا النمط البرمجي على تحقيق سلسلة من المهام والأوامر بترتيب محدد، يتم تحديده عن طريق المبرمج، وقد عملت على هذا النمط البرمجي فترة معينة في بداية استخدام لغة **python**، ويعتبر من الأنماط السهلة والبسيطة التي يمكن فيها إدارة المهام بشكل سهل جدا.

ويتميز هذا النمط باستخدام لغة مباشرة في تحديد واتمام الوظائف المعنية، والتوسع بشكل سهل وبسيط مع وجود محدودية، حيث لا يمكن بناء أنظمة وبرامج معقدة باستخدامه، وهذا ما قد يجعله نمط برمجي مبتدئ –إن صح التعبير– إلا إذا اقترن أو اتحد مع أنماط أخرى.

أستطيع أن اقترح هذا النمط لمبتدئ يحاول تعلم لغة البرمجة؛ لأنها ستكوّن له فهم سريع وواضح حول طريقة العمل والتسلسل في عملية البرمجة.

وعلى حسب خبرتي البسيطة، فإن الفترة القصيرة التي استخدمت فيها هذا النمط كنمط أساسي لكتابة الشيفرة كانت كافية لفهم طبيعة العمل البرمجي من مدخلات ومخرجات، واستيعاب الهدف من التسلسل في الأوامر حتى الوصول للغاية المرجوة منه.

البرمجة الوظيفية

(Functional Programming)

خلال اطلاعي وقراءاتي حول هذا النمط؛ لفهمه بشكل أوضح. حيث أن هذا النمط موجود في أغلب اللغات البرمجية ولكن لا يعتبر نمط أساسي فيها، وهناك عدة لغات تتبنى هذا النمط كنمط أساسي فيها. في هذا الجزء يمكنني الإشارة إلى لغة البرمجة **JavaScript** التي تدعم هذا النمط البرمجي ولا تتخذه كنمط أساسي، وهي لغة برمجة تستخدم على نطاق واسع، وتجربتي البسيطة فيها كانت على نطاق صفحات الشبكة العنكبوتية "**web**"، وتطوير الألعاب.

وقد اطلعت على عدة مقالات لتكوين مفهوم أستطيع شرحه، ووضعه بين يديك بشكل بسيط وسهل، لكن وجدت اختلاف ملحوظ بين جميع المقالات حول هذا النمط مع وجود بعض النقاط المتشابهة القليلة مثل الاعتماد على الدوال فقط في عملية البرمجة وهذا ما يدعو للولوج إلى العمق والتساؤل عن القصد وراء هذا.

في بعض المقالات لاحظت تقسيم النمط الوظيفي إلى عدة صفات ومفاهيم، ومن الصفات التي أتفق عليها هي الوضوح وسهولة الفهم –أقصد الأسلوب الوظيفي في اللغة المذكورة، وأما بعض المفاهيم فهي:

- الوظائف الأولية (**First-Class Functions**)
- عدم وجود حالة متغيرة (**Immutable Variables**)

تمهيدا لشرح هذان المفهومان يجب تبيين المفهوم الرئيسي لهذا النمط وهو أن يُعامل البرنامج كمجموعة من الوظائف التي يمكن تمرير البيانات لها والحصول على نتائج.

الوظائف الأولية (First-Class Functions)

وظائف يمكن تخزينها في متغيرات وتمريرها إلى وظائف أخرى، وهي دوال ترجع نفس النتيجة في جميع الأحوال طالما أن المدخلات لا تتغير، ويعني هذا أنها وظائف واضحة غير معقدة تنجز وظائف محددة بخطوات معينة، كأن يكون هناك وظيفة "هـ" وهي عرض "مرحبا + الاسم"، ثم يتم تخزين هذه الوظيفة في المتغير "ترحيب"، ثم يتم استخدام المتغير كدالة وتمريره إلى دالة أخرى ليتم عرض الوظيفة "هـ" على الشاشة مع تغير الاسم دائما ولا يمكن أن يتغير الاسم ما دامت المدخلات لم تتغير. وما تتميز به الوظائف الأولية أنها ليست ذات تأثير جانبي على البرنامج.

عدم وجود حالة متغيرة (Immutable Variables)

وبناء على ما ذكرنا فإن قيمة المتغير الثابتة غير القابلة للتغيير تعتبر جزءا مهما في البرمجة الوظيفية، وهذا يسهل عملية البرمجة، ويجعلنا نتفادى المشكلات. وبناء المتغيرات يتم عن طريق التخمين، وجعلها غير قابلة للتغيير هو للحد من المدخلات. واضافة قيمة أخرى لمتغير ثابت لا يكون إلا عن طريق إنشاء متغير جديد وهذا لتفادي الأخطاء.

الخاتمة

يعتبر علم البرمجة من العلوم المطلوبة جدا في عصرنا الحالي، وقد اتجهت المجالات الأخرى إلى الاندماج فيه والاعتماد عليه بشكل جزئي أو كلي وهذا ما شكل لنا الثورة المعلوماتية المعاصرة والتطور السريع في أغلب المجالات –إذا لم تكن جميعها.

أخذ العلم في السنوات الأخيرة إلى تسليم الحوسبة زمام الأمور، في اكتشاف الأخطاء وتصحيح المفاهيم، وحل المسائل الرياضية الصعبة.

أذكر في عام **2011** كانت وزارة التربية والتعليم تسلم عملية تصحيح الامتحانات إلى الحاسب الآلي لتسهيل وتسريع العملية، وكانت آنذاك تتخذ هذا الإجراء تجاه نوع واحد من الأسئلة وهي الخيارات المتعددة. أما الآن فقد تطور الأمر وأصبح أكثر شمولا وسهوله.

يتجه العالم الحديث إلى الرقمنة إن صح التعبير وهو تحويل جميع الأشياء إلى أشياء رقمية غير ملموسة، وأنا شخصيا وجدت ما يسمى بالسلع الرقمية، أي أنك تبيع سلعة غير ملموسة مثل الصور، والخطوط، والتصاميم، وتسمى أيضا سلع قابلة للتحميل.

إن أنماط البرمجة التي ذكرت تعتبر رئيسية بناء على تجربتي في هذا المجال، وعلى الرغم من وجود لغات برمجية تعتمد على كل أسلوب على حدة بشكل أساسي إلا أن هناك لغات أخرى أكثر شيوعا وأشمل بالأساليب مثل **java** و**python** و**dart** وعلى المبرمج أن ينوع في الأساليب وأن يختار الأساليب والأنماط التي تخدم هدفه، ولا يحصر نفسه.

لا يجب على البرنامج أو التطبيق أن يكون معقد في الشيفرة ولا في الاستخدام، فإن الذكاء البرمجي يكمن في تصميم شيفرة سهلة وعملية، وهذا المنطلق يتم العمل عليه في مجال الذكاء

الاصطناعي أيضا، ومن خلال هذا المنطق نصل إلى أن الذكاء البرمجي يكمن في البساطة والعملية.

آمل أن تفيد هذه الصفحات القليلة القراء، وأرجو أن تكون قد أضافت حتى ولو الشيء البسيط والمفيد لعلمكم الغزير.