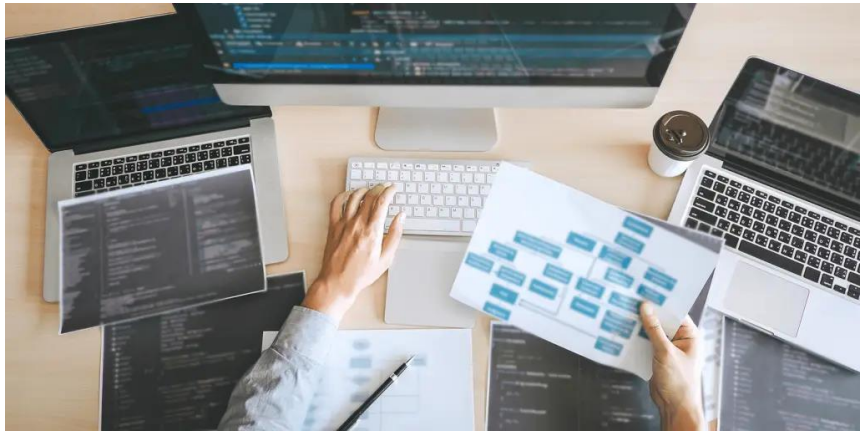
الأكاديمية العربية الدولية
Arab International Academy

اسم المادة: مقدمة في هندسة البرمجيات

المحاضِر: م. خليل المحمد

الأكاديمية العربية الدولية – منصة أعد

ما هي هندسة البرمجيات Software engineering



هندسة البرمجيات هي تطبيق المفاهيم الهندسية لتطوير البرمجيات، وهدفها الرئيسي هو إنشاء البرامج (software) وتحسينها وصيانتها، تأخذ هندسة البرمجيات في الحسبان الجوانب الهندسية مثل بيئة الأجهزة (hardware) والبرامج (software) عند العمل على أحد المشاريع البرمجية، غالبًا ما يتداخل الوصف الوظيفي لمهندس البرمجيات بشكل كبير مع مطوري البرامج، لكنها مختلفة، والاختلاف الرئيسي هو أن مهندسي البرمجيات يطبقون المفاهيم والمبادئ الهندسية لتطوير البرمجيات، وينظرون في القيود البيئية التي سيقيم فيها المشروع البرمجي.

ماذا يفعل مهندس البرمجيات؟

ماذا يفعل مهندس البرمجيات؟

يقوم مهندسو البرمجيات بإنشاء تطبيقات البرامج وصيانتها وإدارتها، وهم مسؤولون عن إنشاء تطبيق فعال يلبي المتطلبات التي حددها العميل أو الشركة، مع مراعاة بيئة الأجهزة والبرامج التي سيقيم فيها المشروع، سيضمن مهندسو البرمجيات تشغيل البرامج النشطة بسلاسة وتحديث البرامج وإصلاح الأخطاء البرمجية وإنشاء برامج جديدة، يكتب أيضاً مهندسو البرمجيات برامج لتكنولوجيات متنوعة، من الأجهزة المنزلية الذكية إلى المساعدين الافتراضيين، اعتمادًا على وظيفتهم.

هل هندسة البرمجيات مناسبة لك؟

هل هندسة البرمجيات مناسبة لك؟

بينما يمكن لأي شخص تعلم هندسة البرمجيات، قد تجد أنك ستنجح وتستمتع بعملك أكثر إذا كانت لديك هذه الصفات:

- عقلية منطقية.
- حب لاستكشاف الأخطاء، وإصلاحها وحل الألغاز.
- القدرة على العمل بشكل جيد ضمن فريق.
- الرغبة في التعلم باستمرار.
- شغف بالتكنولوجيا.

كيف تصبح مهندس برمجيات؟

أولاً : اختيار مسار وظيفى مناسب فى هندسة البرمجيات:

المسارات الأكثر شيوعًا لمهندسي البرمجيات:

مهندس البرمجيات: مهندسو البرمجيات يطورون برمجيات للأجهزة الإلكترونية.

مهندس الأنظمة المدمجة: مهندسو الأنظمة المدمجة مسؤولون عن تصميم الأنظمة المدمجة وتطويرها واختبارها وصيانتها.

مهندس أمن البرمجيات: مهندسو الأمن مسؤولون عن إنشاء الأنظمة والأساليب والسياسات لضمان تلبية نظام المعلومات لمعايير معينة وعدم وجود عيوب أمنية.

مهندس ضمان الجودة: يقوم مهندسو ضمان الجودة بكتابة البرامج ومراجعتها واختبارها وصيانتها.

كيف تصبح مهندس برمجيات؟

ثانياً :التعرف على مسار هندسة البرمجيات من خلال الدراسة الذاتية أو الجامعة:

هناك مسارات لتعلم هندسة البرمجيات، اما اختياره كتخصص في الكلية أو الجامعة، أو يمكنك تعلم هندسة البرمجيات من خلال الدراسة الذاتية (من خلال دورات عبر الإنترنت).

ثالثاً :تطوّر وصقل مهاراتك الفنية أثناء بناء سيرتك الذاتية:

في حين أن جميع خيارات التعليم المذكورة أعلاه هي نقطة انطلاق رائعة، فإن العديد من أصحاب العمل يحتاجون إلى مستوى معين من الخبرة، علاوة على ذلك، من المهم توسيع مهاراتك وتنمية سيرتك الذاتية بمشاريع حقيقية، فيما يلي بعض الطرق لاكتساب الخبرة كمهندس برمجيات: المشاريع الشخصية، العمل الحر، العمل المجاني.

رابعاً :الاستعداد والبدء بالبحث عن وظيفة:

الآن بعد أن حصلت على تعليم، ومجموعة جيدة من المشاريع، حان الوقت لبدء البحث عن أول وظيفة في هندسة البرمجيات لك.

المهارات الفنية الأساسية لمهندسي البرمجيات:

1- **معرفة لغة برمجة :**يجب عليك معرفة لغة برمجة واحدة على الأقل مستخدمة في تطوير البرامج.

2- **اختبار البرمجيات والتصحيح :**بعد كتابة الكود، ستحتاج إلى اختباره والتأكد من أنه يعمل على النحو المنشود.

3- **التصميم الموجه للكائنات: Object-Oriented** أصبح التصميم الموجه للكائنات شائعًا بشكل متزايد خلال السنوات القليلة الماضية، وهي طريقة تصميم تتضمن تخطيط نظام لتفاعل الكائنات لحل مشكلة برمجية.

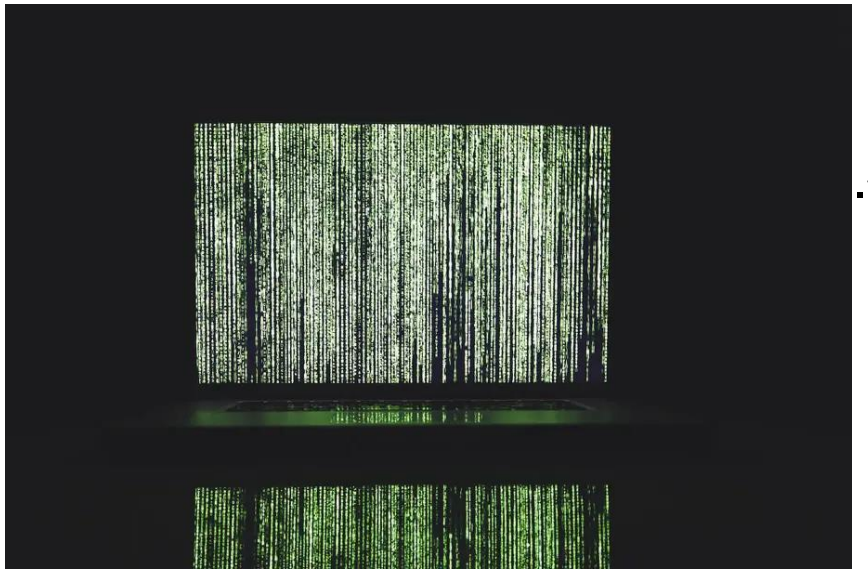
4- **هياكل البيانات والخوارزميات :**يجب أن يكون لديك فهم قوي للخوارزميات الأساسية، مثل خوارزميات الفرز والبحث من خلال مجموعات البيانات.

5- **خدمات الويب وواجهات برمجة التطبيقات: APIs** بصفتك مهندس برمجيات، قد تقوم ببناء أدوات تتفاعل مع الخدمات المستندة إلى الويب وقواعد البيانات.

المهارات الشخصية الأساسية لمهندسي البرمجيات:

- **العمل بروح الفريق الواحد :**كمهندس برمجيات سيكون عليك العمل كجزء من فريق، لذلك، يجب أن تكون قادرًا على العمل بفعالية مع الآخرين، وأن تكون على استعداد لدعم الآخرين، ومعرفة كيفية الاستجابة للبيئات التعاونية.
- **انتبه للتفاصيل :**يحتاج مهندسو البرمجيات إلى أن يكون قادراً على الاهتمام بكل التفاصيل والتفكير في تأثير كل تغيير يقوم به على البرنامج كاملاً، فعلى سبيل المثال، يمكن لتواجد متغير واحدة في غير مكانه أن يمنع البرنامج من العمل، أو من العمل بشكل صحيح.
- **حل المشاكل :**للنجاح في أي دور في البرمجة، عليك أن تكون جيدًا في حل المشكلات، يجب أن تكون قادرًا على تقسيم المشكلات إلى مكونات أصغر، والقدرة على التفكير الناقد للتوصل إلى حل فعال للمشكلة، في الواقع، هندسة البرمجيات تدور حول حل المشكلات من خلال التعليمات البرمجية.

الخوارزميات في برمجة الحاسوب Algorithms



ما هي الخوارزميات Algorithms ؟

الخوارزمية هي مجموعة تعليمات مفصلة خطوة بخطوة أو صيغة لحل مشكلة أو إكمال مهمة.

في مجال الحوسبة، يكتب المبرمجون خوارزميات ترشد الحاسوب إلى كيفية أداء مهمة ما.

ولتقريب المعنى للأذهان يمكنك تشبيه الخوارزمية البرمجية بوصفة الطعام التي تصف (المكونات المطلوبة، الخطوات لكيفية تحضير الوجبة الموصوفة، والنتائج)، فالخوارزمية هي الخطوات، وتسمى المكونات بالمدخلات(Inputs) ، وتسمى النتائج بالمخرجات (Outputs).وعندما تفكر في الخوارزمية بالطريقة الأكثر عمومية (ليس فقط فيما يتعلق بالحوسبة)، فإن الخوارزميات موجودة في كل مكان.

فوائد الخوارزمية في برمجة الحاسوب:

تقليل من التعقيد: complexity

عندما يكتب المبرمج الخوارزمية والبرنامج كل منهم على حدا (بشكل منفصل)، فإنه سيسهل المهمة الكلية بتقسيمها إلى مهمتين أبسط. علاوة على ذلك، أثناء كتابة الخوارزمية، يمكننا التركيز على حل المشكلة بدلاً من التركيز على لغة معينة.

زيادة المرونة:

من أجل كتابة الكود بأي لغة، يكتب المبرمج خوارزمية. تمكن الخوارزمية المبرمج من كتابة البرنامج بأي لغة مثل Visual Basic أو Java أو C++

سهولة الفهم:

لست بحاجة إلى فهم لغة برمجة معينة لفهم الخوارزمية. لأن المبرمج يكتبها باللغة الإنجليزية بطريقة مماثلة.

خصائص الخوارزميات:

الدقة :الخطوات وتفاصيل المدخلات والمخرجات يجب ان تكون مذكورة بدقة (محددة).

التفرد :يتم تحديد نتائج كل خطوة بشكل فريد وتعتمد فقط على المدخلات ونتائج الخطوات السابقة.

المدخلات :تستقبل الخوارزمية المدخلات.

المخرجات :الخوارزمية تنتج المخرجات.

العمومية :تنطبق الخوارزمية على مجموعة من المدخلات.

الفعالية :ان تكون فعالة عند استخدامها لحل المشكلة البرمجية.

الوضوح :يجب ان تكون كل خطوة مكتوبة بطريقة واضحة ومعبرة وخالية من الغموض.

هياكل البيانات Data Structure

غالبًا ما يتم تصنيف هياكل البيانات حسب خصائصها .الخصائص الممكنة هي:

خطي أو غير خطي : تصف هذه الخاصية ما إذا كانت عناصر البيانات مرتبة في تسلسل في الذاكرة "مثل مصفوفة"، أو مرتبة بطريقة عشوائية "مثل الأشجار."(trees)

متجانسة أو غير متجانسة: تصف هذه الخاصية ما إذا كانت جميع عناصر البيانات في مخزن معين من نفس النوع أو من أنواع مختلفة.

ثابت أو ديناميكي : تصف هذه الخاصية كيفية تجميع هياكل البيانات .هياكل البيانات الثابتة لها أحجام وهياكل ومواقع ذاكرة ثابتة في وقت الترجمة (compile time)، وهياكل البيانات الديناميكية لها أحجام وهياكل ومواقع ذاكرة يمكن أن تتقلص أو تتوسع حسب الاستخدام.

أنواع هياكل البيانات:

المصفوفات (Arrays): المصفوفة تخزِن مجموعة من العناصر في مواقع ذاكرة متجاورة.

الستاك (Stack): يخزن مجموعة من العناصر بالترتيب الخطي الذي يتم تطبيق العمليات عليه.

الكيو (Queue): تقوم الكيو بتخزين مجموعة من العناصر المشابهة بعملها للستاك.

القوائم المترابطة (Linked lists): تقوم القائمة المرتبطة بتخزين مجموعة من العناصر بترتيب خطي.

الأشجار (Trees): الشجرة تخزن مجموعة من العناصر بطريقة هرمية.

الرسوم البيانية (Graphs): يخزن الرسم البياني مجموعة من العناصر بطريقة غير خطية

أشجار الجذر (Tries): تُعرف أيضًا باسم أشجار البادئة أو شجرة الكلمات الرئيسية.

جداول التجزئة (Hash tables): يخزن جدول التجزئة أو خريطة التجزئة مجموعة من العناصر في مصفوفة ترابطية.

أهمية هياكل البيانات:

- تسمح هياكل البيانات للمبرمجين بإدارة البيانات بالطرق الأكثر فعالية والتي تعزز أداء الخوارزمية.
- أنها تسمح بالتنفيذ نهج علمي على الأنظمة التقليدية .عندما يتعلق الأمر بجوانب مثل سرعة المعالجة أو المعالجة أو الطلبات المتعددة والبحث من خلال العديد من السجلات، فإن هياكل البيانات تثبت فائدتها.
- أنها تسمح بالاستخدام الفعال للذاكرة .تعمل هياكل البيانات على تحسين استخدام الذاكرة، ويحدث هذا تأثيرًا عادةً في المواقف التي تتضمن معالجة مجموعات بيانات ضخمة.
- هياكل البيانات قابلة لإعادة الاستخدام .يمكن دمجها كحزمة (package) واحدة في مكتبة، ويمكن استخدام هذه المكتبات في مواقف مختلفة من قبل مختلف أصحاب المصلحة حسب الحاجة.
- إنها تتيح حلولًا متعددة لمشكلة معينة، ممّا يسمح للمستخدم بتحديد بنية البيانات التي ستحل المشكلة بأفضل طريقة.

العمليات الرئيسية:

البحث :يمكننا البحث عن أي عنصر في بنية البيانات.

الفرز :يمكننا فرز عناصر بنية البيانات إما بترتيب تصاعدي أو تنازلي.

الإدراج :نستطيع إدراج العنصر الجديد في بنية البيانات.

التحديث :كما يمكننا تحديث العنصر، أي يمكننا استبدال العنصر بعنصر آخر.

الحذف :يمكننا كذلك إجراء عملية الحذف لإزالة العنصر من بنية البيانات.

هندسة متطلبات البرمجيات Software Requirements

ما هي هندسة المتطلبات؟

هندسة المتطلبات(Requirements engineering) ، أو للاختصار(RE) ، وتشير إلى عملية تحديد وتوثيق وصيانة المتطلبات في عملية التصميم الهندسي، وتوفر هندسة المتطلبات الآلية المناسبة لفهم ما يريده العميل، وتحليل حاجته، وتقييم الجدوى الاقتصادية، والتفاوض مع العميل على حلول منطقية وتحديد الحل بوضوح، والتحقق من المواصفات وإدارة المتطلبات أثناء تحويلها إلى نظام عمل، بالتالي، فإن هندسة المتطلبات هي التطبيق المنضبط للمبادئ والأدوات والتدوين التي أثبتت جدواها لوصف السلوك المقصود للنظام المقترح والقيود المرتبطة به.

عمليات هندسة المتطلبات:

1- استنتاج المتطلبات: **Requirements Elicitation** يُعرف هذا أيضًا باسم تجميع المتطلبات، ويتعلق بالطرق المختلفة المستخدمة لاكتساب المعرفة حول مجال المشروع ومتطلباته

2- مواصفات متطلبات البرنامج: **Requirements Specification** يستخدم هذا النشاط لإنتاج نماذج متطلبات البرامج الرسمية بواسطة محلل البيانات، بعد أن تم جمع المتطلبات من مصادر مختلفة، المتطلبات التي تلقاها من العميل مكتوبة باللغة العادية.

3- التحقق من متطلبات البرامج: **Requirements verification and validation**

يشير التحقق (Verification) إلى مجموعة المهام التي تضمن تنفيذ البرنامج لوظيفة معينة بشكل صحيح

4- إدارة متطلبات البرمجيات: **Requirements Management** إدارة المتطلبات هي عملية التحليل والتوثيق والتتبع وتحديد الأولويات والاتفاق على المتطلبات والتحكم في الاتصال بأصحاب المصلحة المعنيين، هذه المرحلة تهتم بالطبيعة المتغيرة للمتطلبات

أنواع المتطلبات في هندسة البرمجيات Software Requirements Types

مفهوم متطلبات البرمجيات:

المتطلب هو شرط أو قدرة التي يمتلكها البرنامج أو مكون من مكونات النظام من أجل حل مشكلة في العالم الحقيقي، ويمكن أن تتمثل المشاكل في جعل جزء من النظام يعمل بشكل آلي والتقليل من التدخل اليدوي، وتصحيح أوجه القصور الموجودة في النظام الحالي، والتحكم في جهاز، وما إلى ذلك. كما تعرّف منظمة "IEEE" المتطلب بأنه:

(1) شرط أو قدرة يحتاجها المستخدم لحل مشكلة أو لتحقيق هدف.

(2) شرط أو قدرة يجب أن يفي بها أو يمتلكها أحد مكونات النظام أو النظام لتلبية العَقد أو معيار أو مواصفات أو مستندات أخرى مفروضة رسميًا.

(3) تمثيل موثق لشرط أو قدرة كما في 1 أو 2.

أنواع المتطلبات:

1- المتطلبات الوظيفية: Functional Requirements

هذه هي المتطلبات التي يطلبها المستخدم النهائي على وجه التحديد كمرافق أساسية يجب أن يوفرها النظام، ويجب بالضرورة دمج كل هذه الوظائف في النظام كجزء من العقد.

2- متطلبات غير الوظيفية: Non-functional requirements

هذا النوع من المتطلبات في الأساس هو قيود الجودة التي يجب أن يستوفيها النظام وفقًا لعقد المشروع، وتختلف أولوية أو مدى تنفيذ هذه العوامل من مشروع إلى آخر، وتسمى أيضًا بالمتطلبات غير السلوكية.

3- متطلبات المجال: Domain requirements

متطلبات المجال هي المتطلبات التي تميز فئة معينة أو مجال معين من المشاريع، وتندرج الوظائف الأساسية التي يجب أن يعرضها نظام مجال معين بالضرورة ضمن هذه الفئة.

الاختلافات بين هندسة البرمجيات وتطوير البرمجيات
Software Engineer vs Software Developer

تتضمن هندسة البرمجيات :

1- جمع وتحليل المتطلبات.

2- دراسة البرمجيات الموجودة وتحديد مجالات التحسين.

3- تقييم رغبات البرمجة للمستهلكين.

4- مراقبة مبرمجي الكمبيوتر أثناء قيامهم بكتابة كود البرنامج.

5- إلقاء نظرة على الكود لتجد أنه يعمل بشكل صحيح.

6- التحقق من أنه سيتم تثبيت البرنامج الجديد

7- صنع تصميم النظام.

8- عمل نماذج أولية.

9- تطوير البرمجيات وكتابة الكود.

10- مناقشات مع أصحاب المصلحة.

11- استكشاف الأخطاء وإصلاحها.

12- التعامل بشكل عام مع أجزاء الأجهزة والشبكات.

13- الاختبار وقيادة الفريق وما إلى ذلك.

الاختلافات بين هندسة البرمجيات وتطوير البرمجيات
Software Engineer vs Software Developer

يتضمن تطوير البرمجيات

1. عمل الواجهة الخلفية والمعلومات.
2. تطوير تطبيقات الواجهة الأمامية.
3. التعاون مع أصحاب المصلحة والمطورين المختلفين لإنشاء البرامج.
4. تقييم البرامج الحالية والدعوة إلى ترقيات البرامج.
5. إنتاج البرامج التي ستُنشئ تطبيقات الكمبيوتر التي تعمل بشكل صحيح.
6. إعطاء المبرمجين مخططات حتى يكتبوا الكود بطريقة صحيحة.
7. إنتاج الطبقة الوسطى من البرمجية.
8. تقديم اقتراحات للمستخدمين عند إساءة معاملة البرنامج.
9. إعطاء بدائل لمتطلبات المستخدمين.
10. التكامل مع برامج الجهات الخارجية.
11. نشر التطبيق

هندسة البرمجيات العكسية
Software Reverse Engineering

ما هي هندسة البرمجيات العكسية؟

منذ طفولتنا، نعلم أنه من الأسهل فهم كيفية عمل الأشياء من خلال تفكيكها ودراستها، من هنا جاءت فكرة هندسة البرمجيات العكسية، إذ تُعرف هندسة البرمجيات العكسية بأنها دراسة شاملة لبرامج أو وثائق أو كود معينة تهدف إلى فهم مبدأ التشغيل وإعادة إنتاج منتج مشابه دون استنساخه.

خطوات هندسة البرمجيات العكسية:

1- جمع معلومات :تركز هذه الخطوة على جمع كل المعلومات حول البرنامج والوثائق الممكنة لتعرّف عليه، على سبيل المثال، وثائق تصميم المصدر.

2- فحص المعلومات :تتم دراسة المعلومات التي تم جمعها في الخطوة الأولى للتعرف على النظام.

3- استخراج الهيكل :تتعلق هذه الخطوة بتحديد هيكل البرنامج على شكل مخطط هيكلي، حيث تتوافق كل عقدة مع بعض الإجراءات الروتينية.

هندسة البرمجيات العكسية
Software Reverse Engineering

4- **تسجيل الوظيفة :**أثناء معالجة هذه الخطوة لتفاصيل كل وحدة من وحدات الهيكل، يتم تسجيل الرسوم البيانية باستخدام لغة منظمة، مثل جدول القرار.

5- **تسجيل تدفق البيانات :**من المعلومات المستخرجة في الخطوة الثالثة والخطوة الرابعة، يتم اشتقاق مجموعة من الرسوم البيانية لتدفق البيانات لإظهار تدفق البيانات بين العمليات.

6- **تسجيل تدفق التحكم :**يتم تسجيل هيكل التحكم عالي المستوى للبرنامج.

7- **مراجعة التصميم المستخرج :**تتم مراجعة وثيقة التصميم المستخرجة عدة مرات لضمان الاتساق والصحة، كما يضمن أن التصميم يمثل البرنامج.

8- **إنتاج الوثائق :**أخيرًا، في هذه الخطوة، يتم تسجيل الوثائق الكاملة بما في ذلك وثيقة (SRS) ووثيقة التصميم والتاريخ والنظرة العامة، وما إلى ذلك، للاستخدام في المستقبل.

فوائد هندسة البرمجيات العكسية:

- الهدف الرئيسي هو الدراسة وليس فقط نسخ البرامج أو الأجهزة.
- دراسة الكود الحالي وآليات منتج الطرف الثالث، إذ يمكن أن تساعدك على خلق الابتكارات.
- خفض التكلفة عند إنشاء منتج أو وظيفة جديدة وتسليمها إلى السوق، والهندسة العكسية دائما أرخص من بدء العمل من نقطة الصفر.
- تسمح الهندسة العكسية باكتشاف نقاط الضعف في المنتج وتحديد مصادرها.
- ليست هناك حاجة إلى "إعادة الاختراع"، ويمكن أن يساعد ذلك في توفير الكثير من الوقت.
- اكتساب المعرفة، التي يمكن أن تكون مفيدة في المستقبل حتى لمنتج جديد تمامًا.

شكراً لحضوركم

آمل ان تكونوا قد حققتم الفائدة