

# تصميم برامج الحاسب الالى

تم بواسطة :دكتور مهندس/سالى محمد على المرسى

## فهرس المحتويات

المحور الاول : المفهوم النظري لتصميم برامج الحاسب الالى	
مقدمة عامه	
الفصل الاول	تحليل المشروع التخطيط والتصميم الأولي
الفصل الثاني	تحليل المتطلبات والتصميم العميق تصميم تفصيلي
الفصل الثالث	تنفيذ ونشر البرمجيات تكوين الخوادم وقواعد البيانات صيانة وحماية البيانات الذكاء الاصطناعي وتعلم الآلة وكيفية دمجهما في التصميم
المحور الثاني : التطبيق العملي	
مقدمه عن التطبيقات العمليه	
الفصل الاول	الخرائط التدفقيه المتغيرات جمل التحكم والحلقات التكراريه
الفصل الثاني	المصفوفات
الفصل الثالث	نبذه عن البرمجه الكائنيه الوراثه وتعدد الاشكال

# المحور الاول : المفهوم النظري لتصميم برامج الحاسب الالى

## 1. مقدمة فى تصميم البرمجيات

تصميم البرمجيات هو عملية تخطيط وإنشاء البرامج الحاسوبية بطريقة منهجية ومنظمة. يهدف تصميم البرمجيات إلى إيجاد حلاً تقنياً لمشكلة معينة أو تحقيق هدف محدد باستخدام البرمجة والتقنيات المناسبة. تعتبر هذه المرحلة من أهم مراحل عملية تطوير البرمجيات، حيث يتم فيها تحديد متطلبات النظام وتصميم هيكله وواجهته.

### تصميم البرمجيات

تصميم البرمجيات هو عملية حاسمة في تطوير البرمجيات حيث يتم التخطيط لهيكل البرمجية وتحديد كيفية تنفيذ الأفكار والمتطلبات بطريقة منهجية ومنظمة. إن تصميم البرمجيات الجيد يلعب دوراً أساسياً في تحقيق الأهداف المنشودة وضمان تنفيذ البرمجية بفعالية وكفاءة.

### أهمية تصميم البرمجيات:

1. **\*\*ضمان تحقيق الأهداف\*\***: تصميم البرمجيات يساعد في تحديد الأهداف والمتطلبات بوضوح، مما يجعل من السهل تنفيذ البرمجية وضمان أن تلبي احتياجات المشروع.

2. **\*\*تقليل التكاليف والوقت\*\***: عندما يتم التخطيط بعناية، يمكن تجنب الأخطاء الشائعة والتعديلات الكبيرة في وقت متأخر من عملية التطوير، مما يوفر الموارد والوقت.

3. **\*\*تحسين جودة البرمجية\*\***: يسمح التصميم الجيد بتحسين هيكل البرمجية وتجنب التشويش والتعقيد غير الضروري، مما يؤدي إلى جودة عالية في البرمجية النهائية.

4. **\*\*تسهيل التعاون\*\***: تصميم البرمجية يمكن أن يساعد في توضيح الأفكار والمفاهيم بشكل مشترك بين أعضاء الفريق والعملاء، مما يعزز التعاون والتفاهم.

عناصر تصميم البرمجيات:

عملية تصميم البرمجيات تشمل العديد من العناصر المهمة:

1. **\*\*تحليل المتطلبات\*\***: هذه المرحلة تتضمن فهم احتياجات المشروع وجمع المتطلبات من العميل. يجب توثيق هذه المتطلبات بدقة واستخدامها كمرشد لتصميم البرمجية.

2. **\*\*تصميم الهيكل المعماري\*\***: في هذه المرحلة، يتم تحديد هيكل البرمجية وتنظيمه إلى مكونات ووحدات. يشمل هذا أيضًا تحديد التقنيات والأدوات المستخدمة.

3. **\*\*تصميم واجهة المستخدم\*\***: يتم هنا تصميم الجزء الذي يتفاعل به المستخدمون مع البرمجية. يجب أن تكون واجهة المستخدم سهلة الاستخدام وجذابة.

4. **\*\*تصميم قاعدة البيانات\*\***: يتعين تصميم هيكل قاعدة البيانات بحيث يمكنه تخزين واسترجاع البيانات بكفاءة وأمان.

5. **\*\*تصميم الأمان\*\***: يتعين تضمين تدابير الأمان المناسبة لحماية البرمجية والبيانات من التهديدات الأمنية.

6. **\*\*أداء البرمجية\*\***: يتعين تحسين أداء البرمجية لضمان استجابة سريعة وكفاءة في استخدام الموارد.

7. **\*\*استدامة وصيانة\*\***: يجب تصميم البرمجية بحيث يمكن صيانتها وتطويرها بسهولة على مر الزمن.

تتفاعل هذه العناصر معًا لإنشاء تصميم شامل للبرمجية تنطلق منه عملية تطويرها بنجاح. يمكن توسيع وتفصيل كل عنصر بشكل مفصل في الكتاب لفهم أعمق لكل جزء من جوانب تصميم البرمجيات.

## 2. أهمية فهم متطلبات المشروع

فهم متطلبات المشروع هو الخطوة الأساسية في تصميم البرمجيات.  
يشمل هذا العمل معرفة احتياجات العملاء وفهم الغرض الرئيسي للبرمجية  
المراد تطويرها. بدون فهم جيد للمتطلبات، يمكن أن يكون التصميم  
والتنفيذ للبرمجية عرضيًا وغير ملائم لاحتياجات المشروع.

تحديد الأولويات والمواصفات الدقيقة للمشروع هو جزء مهم جدًا من عملية تصميم البرمجيات. يشير هذا إلى تحديد أهمية وترتيب المتطلبات المختلفة وتحديد المواصفات التفصيلية التي يجب تنفيذها لضمان تحقيق الأهداف المطلوبة. إليك شرح مفصل لهذه العملية:

### 1. \*\*تحديد الأولويات\*\*:

- **\*\*تصنيف المتطلبات\*\***: يتعين تصنيف المتطلبات إلى فئات أو مستويات مختلفة بناءً على أهميتها. عادة ما يتم تقسيم المتطلبات إلى متطلبات أساسية (Critical) ومتطلبات اختيارية (Optional) ومتطلبات متوسطة (Intermediate) وما إلى ذلك.

- **\*\*تحديد الأهمية\*\***: يجب على فريق المشروع تحديد الأهمية النسبية لكل متطلب. مثلاً، متطلبات الأمان والأداء قد تكون أكثر أهمية من متطلبات واجهة المستخدم.

- **\*\*تحديد التسلسل\*\***: بعد تصنيف المتطلبات وتحديد الأهمية، يجب تحديد ترتيب تنفيذها. أي متطلب يجب تنفيذه أولاً وأيهم يمكن تأجيله إلى وقت لاحق.

## 2. **\*\*تحديد المواصفات الدقيقة\*\***:

- **\*\*تفصيل المتطلبات\*\***: يجب تفصيل المتطلبات بدقة بحيث تكون قابلة للتنفيذ. على سبيل المثال، إذا كان المتطلب هو "النظام يجب أن يكون سريعاً"، فيجب تحديد ماذا يعني بالضبط "سريعاً" وكيف سيتم قياس السرعة.

- **\*\*المواصفات التقنية\*\***: يجب تحديد المواصفات التقنية بشكل دقيق، مثل أنواع التقنيات والأدوات التي ستستخدم واللغات البرمجية وأجهزة الأنظمة المستهدفة.

- **\*\*المواصفات الوظيفية والأداء\*\***: يجب تحديد بالتفصيل كيف ستعمل ميزات المشروع وما هي الأداء المتوقع للبرمجية في مختلف السيناريوهات.



- \*\*تحديد المعايير والمعايير المرجعية\*\* : قد تكون هناك معايير أو معايير مرجعية تحتاج إلى الامتثال بها، مثل معايير الأمان أو معايير التوافق.

3. \*\*استعراض والتحقق\*\* :

- \*\*الاستعراض الداخلي\*\* : يجب أن يتم استعراض المواصفات من قبل فريق المشروع والمهندسين والمهندسين المعماريين للتأكد من دقتها واكتمالها.

- \*\*المراجعة من قبل العميل\*\* : يجب عرض المواصفات على العميل أو الجهة الممولة للمشروع للمراجعة والموافقة.

- \*\*التحقق المستمر\*\* : يجب أن يتم التحقق المستمر من تقدم المشروع ومدى توافقه مع المواصفات المحددة.

تحديد الأولويات والمواصفات الدقيقة للمشروع يساعد في توجيه جهود فرق التطوير وضمان أن المشروع ينفذ بفعالية وينتج في النهاية في نظام برمجي يلبي احتياجات وتوقعات العميل بشكل كامل ورضاء.

## عملية فهم المتطلبات تشمل:

### - التحدث مع العملاء والمستفيدين لجمع المعلومات.

التحدث مع العملاء والمستفيدين لجمع المعلومات هو أحد الخطوات الرئيسية في فهم متطلبات المشروع عند تصميم البرمجيات. يعتمد نجاح هذه الخطوة على التواصل الفعال مع الأشخاص الذين سيستخدمون البرمجية أو لديهم احتياجات واضحة من المشروع. إليك شرحاً مفصلاً لهذه الخطوة:

1. **\*\*تحديد الأطراف المعنية\*\***: في البداية، يجب تحديد الأشخاص والجهات الذين لهم صلة بالمشروع. هذه الأطراف تشمل العملاء (المنظمة أو الأفراد الذين يمولون المشروع) والمستفيدين (الأشخاص الذين سيستخدمون البرمجية) وأي أطراف أخرى ذات صلة.

2. **\*\*ترتيب لقاءات واجتماعات\*\***: يتم تنظيم اجتماعات ولقاءات مع هذه الأطراف للتواصل المباشر. قد تكون هذه الاجتماعات وجهاً لوجه أو عبر وسائل الاتصال عن بعد مثل مكالمات الفيديو أو الهاتف.

3. **\*\*تحضير للقاءات\*\***: قبل اللقاءات، يجب أن يتم التحضير بعناية. ذلك يتضمن دراسة الوثائق والمعلومات المتاحة عن المشروع وإعداد أسئلة وأجندة للمناقشة.

4. **\*\*الاستماع بعناية\*\***: خلال اللقاءات، يجب على المهندسين و فرق التصميم أن يكونوا على استعداد للاستماع بعناية إلى ما يقوله العملاء والمستفيدين. ينبغي عدم الانصات فقط للكلمات بل أيضاً للرسائل المخفية والاحتياجات غير المعبر عنها بوضوح.

5. **\*\*طرح الأسئلة الصحيحة\*\***: يجب على المهندسين طرح أسئلة استفسارية للعملاء والمستفيدين لفهم تفاصيل المشروع والمتطلبات بشكل دقيق. يمكن أن تشمل هذه الأسئلة الأهداف المرجوة، والعمليات الحالية، والمشكلات المحتملة، والمتطلبات الوظيفية وغير الوظيفية.

6. **\*\*التوثيق والمتابعة\*\***: بعد اللقاءات، يجب توثيق جميع المعلومات التي تم جمعها بدقة. هذا التوثيق يمكن أن يكون في شكل مذكرات اجتماعات أو وثائق توصيف المتطلبات. كما يجب إجراء متابعة منتظمة لضمان فهم صحيح للمتطلبات وللتأكد من أنها لا تتغير مع تقدم المشروع.

7. **\*\*التفاعل المستمر\*\***: ينبغي أن يتم التفاعل المستمر مع العملاء والمستفيدين خلال عملية تصميم البرمجيات. يمكن طرح استفسارات إضافية وتوضيحات وتحديثات على الوثائق حسب الحاجة.

فهم متطلبات المشروع من خلال التحدث مع العملاء والمستفيدين بشكل فعال هو الأساس لضمان تصميم برمجية تلبي احتياجات وتوقعات المشروع بدقة. هذه الخطوة تساعد في تحديد الأهداف والمتطلبات بوضوح، مما يمهد الطريق لتصميم وتنفيذ ناجح للبرمجية.

## - توثيق المتطلبات بشكل دقيق ومفهوم.

توثيق المتطلبات بشكل دقيق ومفهوم هو خطوة حيوية في عملية فهم متطلبات المشروع عند تصميم البرمجيات. يهدف هذا التوثيق إلى توثيق جميع المعلومات المتعلقة بمتطلبات المشروع بشكل يجعلها مفهومة وقابلة للتنفيذ. فيما يلي شرح مفصل لهذه الخطوة:

1. **\*\*ترتيب المعلومات\*\***: قبل البدء في توثيق المتطلبات، يجب تنظيم المعلومات بشكل منظم. ذلك يعني فهم المعلومات التي تم جمعها من العملاء والمستفيدين وتحليلها بعناية.
2. **\*\*استخدام قوالب\*\***: يمكن استخدام قوالب جاهزة لتوثيق المتطلبات. هذه القوالب تسهل عملية التوثيق وتساعد في ضمان أن لا يتم نسي أي معلومات مهمة.
3. **\*\*توثيق المتطلبات الوظيفية وغير الوظيفية\*\***: يجب توثيق كل من المتطلبات الوظيفية (ماذا يجب أن تفعل البرمجية) والمتطلبات غير الوظيفية (كيفية يجب أن تكون البرمجية) بشكل منفصل.
4. **\*\*استخدام لغة بسيطة وواضحة\*\***: يجب أن تكون الوثائق مكتوبة بلغة بسيطة وواضحة تفهمها جميع الأطراف المعنية بالمشروع، بغض النظر عن خلفيتهم التقنية.

5. **\*\*ترقيم وتسلسل المتطلبات\*\***: يمكن استخدام نظام ترقيم لترقيم المتطلبات بشكل فعال. على سبيل المثال، يمكن ترقيم المتطلبات الوظيفية بـ F1، F2، F3، وهكذا، والمتطلبات غير الوظيفية بـ NF1، NF2، NF3، وهكذا.

6. **\*\*تحديد المصدر والمسؤولية\*\***: يجب تحديد من هو مسؤول عن كل مطلب ومن أين تمت مصادر هذه المتطلبات. ذلك يساعد في حالة الحاجة إلى توضيح أو تعديل.

7. **\*\*الرسوم التوضيحية والمخططات\*\***: قد يكون من المفيد استخدام رسوم توضيحية أو مخططات لتوضيح بعض المتطلبات بشكل أفضل، خاصة إذا كانت تتعلق بتصميم واجهة المستخدم أو هياكل البيانات.

8. **\*\*التحقق والتدقيق\*\***: يجب أن يتم التحقق من دقة التوثيق ومطابقته لمتطلبات المشروع بشكل دوري. يمكن استخدام عمليات التدقيق لضمان أن لا توجد متطلبات متناقضة أو مفقودة.

9. **\*\*تحديث مستمر\*\***: يجب تحديث وثائق المتطلبات بشكل مستمر بمرور الوقت. قد تتغير المتطلبات أو تظهر احتياجات جديدة أثناء تطور المشروع.

10. **\*\*المشاركة والمراجعة\*\***: ينبغي أن تكون عملية توثيق المتطلبات عملية تشاركية تشمل المهندسين والعملاء والمستفيدين. يمكن للمراجعة المشتركة ضمان أن جميع الأطراف متفقة على المتطلبات الموثقة.

توثيق المتطلبات بشكل دقيق ومفهوم هو جزء أساسي من عملية تصميم البرمجيات. إنه يساعد في تجنب الارتباك والتفسيرات المضللة ويضمن أن البرمجية تنفذ بنجاح وتلبي توقعات واحتياجات المشروع بدقة.

### **- تحليل المتطلبات لفهم كيفية تحقيقها بواسطة البرمجيات.**

تحليل المتطلبات هو عملية حاسمة في عملية تصميم البرمجيات تأتي بعد جمع وتوثيق المتطلبات. يهدف هذا الخطوة إلى فهم كيفية تحقيق هذه المتطلبات باستخدام البرمجيات. يتضمن التحليل تحليل البيانات والمعلومات الموجودة وترتيبها وتصنيفها بشكل مناسب لضمان تصميم نظام برمجي يلبي الاحتياجات بكفاءة وفعالية. إليك شرح مفصل لهذه الخطوة:

1. **\*\*استيضاح المتطلبات\*\***: في هذه المرحلة، يجب فهم المتطلبات بشكل عميق. يعني ذلك الاطلاع على الوثائق والمعلومات الموجودة والتحقق من فهمها بشكل صحيح. إذا كان هناك أي استفسارات أو توضيحات مطلوبة، يجب طرحها على العملاء أو فرق المشروع.

2. **\*\*تحليل البيانات\*\***: يتعين على محلي النظام تحليل البيانات المتاحة والتي ترتبط بالمتطلبات. ذلك يشمل فحص البيانات المعنية، وتصنيفها وترتيبها، وتحليلها لفهم كيف يتم استخدامها وكيف يمكن تجميعها ومعالجتها من خلال البرمجية.

3. **\*\*تقييم الأثر\*\***: يتم تقييم الأثر الذي ستكون له المتطلبات على نظام البرمجيات بأكمله. يجب تحديد مدى تأثير تلك المتطلبات على هيكل البرمجية الحالي ومدى توافقها مع الهيكل المعماري المخطط له.

4. **\*\*تحديد الحلول الممكنة\*\***: بناءً على تحليل المتطلبات والتقييم السابق، يجب تحديد الحلول الممكنة لتلبية تلك المتطلبات. يمكن أن تشمل الحلول تصميم وتنفيذ ميزات جديدة أو تعديل البرمجية الحالية.

5. **\*\*توثيق القرارات\*\***: يجب توثيق القرارات المتخذة بشكل دقيق. ذلك يشمل تسجيل القرارات المتعلقة بتصميم البرمجية وكيفية تحقيق المتطلبات.

6. **\*\*تحديد التكاليف والجداول الزمنية\*\***: يتم تحديد التكاليف المتوقعة لتنفيذ الحلول المقترحة وإعداد جداول زمنية لتنفيذ المشروع. هذا يساعد في التخطيط للموارد وتحديد الخطوات التالية.

7. **\*\*التواصل مع فرق التطوير\*\***: يجب التواصل الفعال مع فرق التطوير لضمان تنفيذ الحلول بشكل صحيح وفقاً للمتطلبات.

8. **\*\*استعراض ومراقبة\*\***: يجب أن يتم استعراض الحلول المقترحة ومراقبتها بشكل منتظم للتأكد من تقدم المشروع بالشكل المطلوب وفقاً للمتطلبات.

تحليل المتطلبات هو جزء حاسم من عملية تصميم البرمجيات حيث يتم تحويل المتطلبات من مستوى المفهوم إلى تصميم وتنفيذ فعلي للبرمجية. هذه الخطوة تضمن تنفيذ البرمجية بكفاءة وفعالية وتحقيق الأهداف المرجوة من المشروع.

### **- تحديد الأولويات والمواصفات الدقيقة للمشروع.**

تحديد الأولويات والمواصفات الدقيقة للمشروع هو جزء مهم جداً من عملية تصميم البرمجيات. يشير هذا إلى تحديد أهمية وترتيب المتطلبات المختلفة وتحديد المواصفات التفصيلية التي يجب تنفيذها لضمان تحقيق الأهداف المطلوبة. إليك شرح مفصل لهذه العملية:

#### **1. **\*\*تحديد الأولويات\*\***:**

- **\*\*تصنيف المتطلبات\*\***: يتعين تصنيف المتطلبات إلى فئات أو مستويات مختلفة بناءً على أهميتها. عادة ما يتم تقسيم المتطلبات إلى متطلبات أساسية (Critical) ومتطلبات اختيارية (Optional) ومتطلبات متوسطة (Intermediate) وما إلى ذلك.



- **\*\*تحديد الأهمية\*\***: يجب على فريق المشروع تحديد الأهمية النسبية لكل مطلب. مثلاً، متطلبات الأمان والأداء قد تكون أكثر أهمية من متطلبات واجهة المستخدم.

- **\*\*تحديد التسلسل\*\***: بعد تصنيف المتطلبات وتحديد الأهمية، يجب تحديد ترتيب تنفيذها. أي مطلب يجب تنفيذه أولاً وأيهم يمكن تأجيله إلى وقت لاحق.

## 2. **\*\*تحديد المواصفات الدقيقة\*\***:

- **\*\*تفصيل المتطلبات\*\***: يجب تفصيل المتطلبات بدقة بحيث تكون قابلة للتنفيذ. على سبيل المثال، إذا كان المطلب هو "النظام يجب أن يكون سريعاً"، فيجب تحديد ماذا يعني بالضبط "سريعاً" وكيف سيتم قياس السرعة.

- **\*\*المواصفات التقنية\*\***: يجب تحديد المواصفات التقنية بشكل دقيق، مثل أنواع التقنيات والأدوات التي ستستخدم واللغات البرمجية وأجهزة الأنظمة المستهدفة.

- **\*\*المواصفات الوظيفية والأداء\*\***: يجب تحديد بالتفصيل كيف ستعمل ميزات المشروع وما هي الأداء المتوقع للبرمجية في مختلف السيناريوهات.

- \*\*تحديد المعايير والمعايير المرجعية\*\* : قد تكون هناك معايير أو معايير مرجعية تحتاج إلى الامتثال بها، مثل معايير الأمان أو معايير التوافق.

### 3. \*\*استعراض والتحقق\*\* :

- \*\*الاستعراض الداخلي\*\* : يجب أن يتم استعراض المواصفات من قبل فريق المشروع والمهندسين والمهندسين المعماريين للتأكد من دقتها واكتمالها.

- \*\*المراجعة من قبل العميل\*\* : يجب عرض المواصفات على العميل أو الجهة الممولة للمشروع للمراجعة والموافقة.

- \*\*التحقق المستمر\*\* : يجب أن يتم التحقق المستمر من تقدم المشروع ومدى توافقه مع المواصفات المحددة.

تحديد الأولويات والمواصفات الدقيقة للمشروع يساعد في توجيه جهود فرق التطوير وضمان أن المشروع ينفذ بفعالية وينتج في النهاية في نظام برمجي يلبي احتياجات وتوقعات العميل بشكل كامل ورضاء.

### 3. عناصر تصميم البرمجيات

عناصر تصميم البرمجيات هي المكونات والجوانب التي يتعين على المطورين النظر فيها عند تصميم البرمجيات. هذه العناصر تشمل:

**- \*\*الهندسة المعمارية\*\*:** تصميم الهيكل العام للبرمجية وتحديد كيفية تنظيم المكونات والأنظمة المختلفة في النظام. تشمل هذه العملية اختيار التقنيات المناسبة وتحديد الاتصالات بين الأجزاء المختلفة.

**- \*\*تصميم واجهة المستخدم\*\*:** تصميم وتطوير واجهة المستخدم الرسومية أو الواجهة النصية التي يتفاعل من خلالها المستخدم مع البرمجية. يجب أن تكون الواجهة سهلة الاستخدام ومستجيبة لاحتياجات المستخدمين.

**- \*\*قاعدة البيانات\*\*:** تصميم هيكل قاعدة البيانات التي تحتفظ بالبيانات المستخدمة في البرمجية. يجب تحديد كيفية تخزين واسترجاع البيانات بكفاءة وأمان.

**- \*\*أمان البرمجية\*\*:** وضع استراتيجيات وتدابير لحماية البرمجية والبيانات من التهديدات الأمنية مثل الاختراقات وسرقة المعلومات.

**- \*\*أداء البرمجية\*\*:** تحسين أداء البرمجية لضمان استجابة سريعة وكفاءة في استخدام الموارد. يتضمن ذلك تحسين وقت الاستجابة واستهلاك الذاكرة واستخدام المعالج.

**- \*\*استدامة وصيانة\*\*: تصميم البرمجية بحيث يمكن صيانتها**  
وتطويرها بسهولة على مر الزمن. يتضمن ذلك استخدام أفضل الممارسات  
في البرمجة وتوثيق الشيفرة وإدارة الإصدارات.

## الفصل الاول: عملية تصميم البرمجيات



## عملية تصميم البرمجيات

### 1.1. تحليل المشروع

عملية تصميم البرمجيات هي عملية تخطيط وإعداد الخطوط العريضة لكيفية تنفيذ نظام أو تطبيق برمجي. تحليل المشروع هو المرحلة الأولى والمهمة في هذه العملية، وتشمل فهم المشروع وتحديد متطلباته وأهدافه. إليك شرح مفصل لعملية تحليل المشروع في عملية تصميم البرمجيات:

#### 1. \*\*فهم المشروع\*\*:

- في هذه المرحلة، يتعين على فريق تصميم البرمجيات فهم المشروع بشكل كامل. هذا يشمل فهم طبيعة العمل الذي سيقوم به البرمجية والغرض منه. يتعين على الفريق معرفة من سيكون المستخدمين النهائيين للبرمجية وماهي احتياجاتهم.

#### 2. \*\*جمع المعلومات\*\*:

- يجب جمع معلومات تفصيلية حول المشروع. ذلك يشمل إجراء مقابلات مع العملاء والمستخدمين وجمع المتطلبات منهم. يمكن أيضاً تحليل الوثائق المتاحة والمصادر الأخرى للمعلومات.

#### 3. \*\*تحديد المتطلبات\*\*:

- بناءً على المعلومات التي تم جمعها، يجب تحديد المتطلبات الرئيسية للمشروع. هذه المتطلبات تشمل وظائف البرمجية وميزاتها وأدائها المتوقع وأي متطلبات إضافية مثل الأمان والتوافق.

#### 4. \*\*توثيق المتطلبات\*\*:

- يجب توثيق المتطلبات بشكل دقيق وواضح. هذا يشمل كتابة وثائق تحتوي على وصف مفصل لكل متطلب وما يجب تحقيقه. يمكن استخدام أدوات توثيق مثل وثائق المتطلبات الوظيفية والغير وظيفية.

#### 5. \*\*تحليل المتطلبات\*\*:

- في هذه المرحلة، يتم تحليل المتطلبات لفهم كيفية تحقيقها باستخدام البرمجة. يشمل ذلك تفصيل المتطلبات إلى عمليات وخوارزميات وتصميم واجهة المستخدم إذا كانت ذلك جزءاً من المشروع.

#### 6. \*\*تحديد الأولويات والمواصفات\*\*:

- يجب تحديد الأولويات بناءً على أهمية المتطلبات ومرتبتها. يجب أيضاً تحديد المواصفات الدقيقة للمشروع، بما في ذلك المواصفات التقنية والأداء المتوقع ومعايير الجودة.

#### 7. \*\*المراجعة والموافقة\*\*:

- يجب مراجعة المتطلبات والمواصفات مع العملاء والمستفيدين للتأكد من توافقها مع متطلباتهم. يجب الحصول على موافقتهم على المتطلبات المحددة.

تحليل المشروع هو الخطوة الأساسية في عملية تصميم البرمجيات حيث يتم تحديد ما يجب تحقيقه بدقة قبل الشروع في التصميم والتنفيذ. إذا تم تنفيذ هذه العملية بشكل صحيح، سيتم توجيه عملية التصميم والتطوير بشكل أكثر فعالية وفعالية.

#### - تحديد الأهداف والمتطلبات

تحديد الأهداف والمتطلبات هي إحدى الخطوات الرئيسية في عملية تصميم البرمجيات. تهدف هذه الخطوة إلى تحديد ماهية الهدف الرئيسي للبرمجية والمتطلبات الضرورية لتحقيق هذا الهدف بشكل كامل وفعال. إليك شرح مفصل لهذه الخطوة:

## 1. \*\*تحديد الأهداف\*\*:

- \*\*فهم الغرض\*\* : يجب أولاً وقبل كل شيء فهم الغرض الرئيسي للبرمجية. لماذا يجري تصميم البرمجية؟ ما الهدف الرئيسي الذي تهدف إلى تحقيقه؟ هل البرمجية تهدف إلى حل مشكلة معينة أو تقديم خدمة محددة أو تحسين عملية معينة؟

- \*\*تحديد الأهداف الرئيسية\*\* : بناءً على فهم الغرض، يجب تحديد الأهداف الرئيسية التي يجب تحقيقها بواسطة البرمجية. يمكن أن تشمل هذه الأهداف تحقيق أداء محدد، أمان معين، واجهة مستخدم مميزة، توافق مع متطلبات محددة، وغيرها من الأهداف.

## 2. \*\*تحديد المتطلبات\*\*:

- \*\*تحديد المتطلبات الوظيفية\*\* : هذه المتطلبات تحدد ما يجب أن تفعله البرمجية. مثلاً، إذا كان هدف البرمجية هو نظام إدارة المستودعات، فإن



المتطلبات الوظيفية قد تتضمن إمكانية إضافة وحذف الأصناف، وإدارة المخزون، وإنشاء تقارير عن المخزون، وما إلى ذلك.

- **\*\*تحديد المتطلبات غير الوظيفية\*\***: هذه المتطلبات تتعامل مع الجوانب الغير وظيفية للبرمجية مثل الأمان، والأداء، والتوافق، والقابلية للتوسع، واستهلاك موارد النظام، وغيرها. على سبيل المثال، يمكن أن تتطلب المشروع الامتثال لمعايير الأمان وحماية البيانات.

### 3. **\*\*توثيق الأهداف والمتطلبات\*\***:

- **\*\*إعداد وثائق الأهداف والمتطلبات\*\***: يجب توثيق الأهداف والمتطلبات بشكل دقيق وواضح. يتم ذلك عادة من خلال كتابة وثائق تحتوي على وصف مفصل لكل هدف ومتطلب.

- **\*\*مشاركة الوثائق\*\***: يجب مشاركة وثائق الأهداف والمتطلبات مع أعضاء الفريق والعملاء والمستفيدين للحصول على موافقتهم وتوضيح توقعاتهم.

### 4. **\*\*التحقق والمراجعة\*\***:

- **\*\*التحقق من المتطلبات\*\***: يجب التحقق من أن المتطلبات تتوافق مع الأهداف المحددة وأنها كافية لتحقيق الغرض من البرمجية.

- **\*\*مراجعة وثائق المشروع\*\***: يجب مراجعة وثائق الأهداف والمتطلبات بانتظام والتأكد من تحديثها إذا كان هناك تغييرات أو تعديلات.

تحديد الأهداف والمتطلبات هو خطوة أساسية في تصميم البرمجيات حيث تساعد في تحديد الاتجاه الصحيح للمشروع وضمان تلبيته لاحتياجات العميل والمستخدمين بشكل كامل وفعال. يمكن أن تقوم هذه الخطوة بتوجيه عملية تصميم البرمجيات وتجنب المشاكل في وقت لاحق.

### - استطلاع الجمهور المستهدف

استطلاع الجمهور المستهدف (Audience Research) هو عملية جمع المعلومات والفهم العميق للفئة أو الجمهور الذي سيكون مستخدماً للبرمجية أو المنتج. يعتبر هذا الاستطلاع جزءاً مهماً من عملية تصميم البرمجيات وتطوير المنتجات بشكل عام، حيث يمكن أن يؤثر بشكل كبير على نجاح وجاذبية المنتج للجمهور المستهدف. إليك شرح مفصل لهذه الخطوة:

#### 1. **\*\*تحديد الجمهور المستهدف\*\***:

- الخطوة الأولى هي تحديد بدقة من هم أفراد الجمهور المستهدفين. يجب معرفة سماتهم الرئيسية مثل العمر، والجنس، والتعليم، والاهتمامات، والاحتياجات، والعادات، والتوقعات.

## 2. \*\*جمع المعلومات\*\*:

- يتم جمع المعلومات من مصادر متعددة. يمكن أن تشمل هذه المصادر استبيانات، ومقابلات مع العملاء المحتملين أو المستخدمين النهائيين، ودراسات السوق، والبيانات الإحصائية، وتقارير البحث.

## 3. \*\*تحليل البيانات\*\*:

- بمجرد جمع البيانات، يتعين تحليلها بعمق لفهم الاتجاهات والأنماط والاحتياجات. يمكن استخدام أدوات تحليل البيانات مثل الإحصاءات والرسوم البيانية لتبسيط البيانات واستخراج القرارات الهامة.

## 4. \*\*تحديد احتياجات الجمهور\*\*:

- يجب تحديد الاحتياجات والمتطلبات الخاصة بالجمهور المستهدف. ما هي الميزات أو الوظائف التي يتوقعونها من المنتج أو البرمجية؟ ما هي المشاكل التي يواجهونها والتي يمكن أن يساعد المنتج في حلها؟

## 5. \*\*تحديد التحديات والفرص\*\*:

- بناءً على فهم احتياجات الجمهور المستهدف، يمكن تحديد التحديات والفرص المتعلقة بالمنتج أو البرمجية. ما هي الصعوبات التي يمكن أن تواجه تصميم المنتج وكيف يمكن التعامل معها بشكل فعال؟ وما هي الفرص التي يمكن استغلالها لجعل المنتج أكثر جاذبية للجمهور المستهدف؟

## 6. \*\*توجيه عملية التصميم والتطوير\*\*:

- يتم استخدام معرفة الجمهور المستهدف لتوجيه عملية تصميم وتطوير المنتج أو البرمجية. هذا يتضمن اتخاذ قرارات تصميمية مبنية على احتياجات وتوقعات الجمهور.

## 7. \*\*الاستمرار في المراقبة والتقييم\*\*:

- لا تنتهي عملية استطلاع الجمهور المستهدف بانتهاء مرحلة التصميم والتطوير. يجب الاستمرار في مراقبة ردود الفعل والمراجعات والتقييمات من الجمهور لضمان استمرار تلبية احتياجاتهم وتحسين المنتج.

استطلاع الجمهور المستهدف هو أساسي لضمان تطوير منتج أو برمجية ناجحة وملائمة للسوق. فهو يساعد على تجنب الاستثمار في مشروع قد لا يكون مطابقاً لتوقعات الجمهور وبالتالي يزيد من فرص نجاح المنتج.

## 1.2. التخطيط والتصميم الأولي

التخطيط والتصميم الأولي (Planning and Initial Design) هما جزءان مهمان من عملية تصميم البرمجيات وتطوير المنتجات. في هذه الخطوة، يتم تحديد كيف سيتم تنفيذ البرمجية أو المنتج بناءً على المتطلبات واحتياجات الجمهور المستهدف. إليك شرح مفصل لهذه الخطوة:

**\*\*التخطيط\*\*:**

1. **\*\*تحديد الأهداف الإستراتيجية:\*\*** تبدأ الخطوة بتحديد أهداف البرمجية أو المنتج بالنسبة للشركة أو المشروع. هل الهدف هو تحقيق الربح؟ هل الهدف هو توفير حلاً لمشكلة معينة؟ يجب تحديد هذه الأهداف بشكل واضح.

2. **\*\*تحليل الأسواق والمنافسين:\*\*** يجب دراسة السوق المستهدفة والتعرف على المنافسين وتحليل منتجاتهم. هذا يساعد في تحديد موقع المنتج المستقبلي في السوق وفهم مدى تنافسيته.

3. **\*\*تخطيط الموارد:\*\*** يتعين تحديد الموارد اللازمة لتطوير وتنفيذ المشروع. هذا يشمل الميزانية، والعمالة، والأجهزة والبرمجيات المطلوبة.

4. **\*\*جدولة المشروع:\*\*** يجب وضع جدول زمني للمشروع يحدد المهام والمواعيد النهائية لتسليم المنتج. هذا يساعد في إدارة المشروع وتنظيم الجهود.

**\*\*التصميم الأولي:\*\***

1. **\*\*تحديد التصميم الهيكلي:\*\*** في هذه المرحلة، يتم تحديد هيكل البرمجية أو المنتج بشكل عام. ما هي الأقسام الرئيسية للمنتج؟ كيف ستتفاعل مع بعضها البعض؟

2. **\*\*تصميم واجهة المستخدم:** \*\* إذا كان المنتج يتضمن واجهة مستخدم، فيجب تصميم هذه الواجهة بشكل أولي. هذا يشمل تحديد تخطيط الصفحات وعناصر التحكم وتجربة المستخدم.

3. **\*\*تحديد التقنيات والأدوات:** \*\* يجب تحديد التقنيات والأدوات التي ستستخدم في تطوير المشروع. مثل اللغات البرمجية وقواعد البيانات والأدوات التطويرية.

4. **\*\*تقديم مخططات أولية:** \*\* يمكن إعداد مخططات ورسومات أولية للمشروع توضح التصميم والهيكل المقترح. هذه المخططات تساعد في توضيح الفكرة لأعضاء الفريق والعملاء.

5. **\*\*تقديم نموذج أولي (Prototype):** \*\* في بعض الحالات، يمكن إنشاء نموذج أولي للمنتج لتجربته وجمع ردود فعل من أجل التحسين اللاحق.

هذه الخطوات تساعد في وضع أسس قوية لعملية تطوير البرمجيات أو تصميم المنتج. من خلال التخطيط الجيد والتصميم الأولي، يمكن تجنب مشاكل تأخر المشروع أو عدم تلبية احتياجات العملاء وضمان تقديم منتج ناجح.

## - تصميم مخطط عام للبرمجية

تصميم مخطط عام للبرمجية هو خطوة حاسمة في عملية تصميم البرمجيات. يتعين أن يكون هذا المخطط عبارة عن رؤية شاملة لهيكل

البرمجية وكيفية تنظيم مكوناتها وتفاعلها. إليك شرح مفصل لكيفية تصميم مخطط عام للبرمجية:

## **\*\*1. تحديد الأهداف والمتطلبات:\*\***

- يبدأ التصميم بفهم الأهداف والمتطلبات التي تم تحديدها خلال مرحلة تحليل المشروع. هذه الأهداف والمتطلبات تحدد ما يجب تحقيقه من خلال التصميم.

## **\*\*2. تحديد المكونات الرئيسية:\*\***

- قبل بدء تصميم المخطط، يجب تحديد المكونات الرئيسية للبرمجية. هذه المكونات قد تشمل الوحدات البرمجية، وقواعد البيانات، وواجهات المستخدم، ومكتبات البرمجة، وأنظمة الاتصال بين البرامج.

## **\*\*3. رسم المخطط العام:\*\***

- يتم رسم المخطط العام باستخدام أدوات تصميم خاصة مثل الرسم البياني أو البرمجيات المخصصة لتصميم البرمجيات. يجب تضمين جميع المكونات الرئيسية وتوصيلها بشكل منطقي.

## **\*\*4. توضيح التدفقات والتفاعلات:\*\***

- يجب وضع توضيح لكيفية تدفق البيانات والتفاعلات بين المكونات. مثلاً، كيف يتم نقل البيانات من قاعدة البيانات إلى واجهة المستخدم؟ كيف تتفاعل الوحدات البرمجية المختلفة مع بعضها البعض؟

## **\*\*5. تحديد واجهات المستخدم (UI):\*\***

- إذا كانت البرمجية تشمل واجهة مستخدم، يجب تصميم واجهات المستخدم بشكل دقيق. ذلك يشمل تحديد تخطيط الصفحات والعناصر المرئية مثل الأزرار والنصوص والصور.

## **\*\*6. توضيح الأمان والأداء:\*\***

- يجب أن يتم التوضيح بشكل واضح لكيفية تحقيق الأمان والأداء المطلوبين. مثلاً، كيف ستتم معالجة البيانات الحساسة؟ وكيف سيتم تحسين أداء البرمجية في حالة التحميل العالي؟

## **\*\*7. مراجعة وتقييم:\*\***

- يجب مراجعة المخطط العام مع الأعضاء المعنيين بالمشروع والعملاء لجمع ملاحظات وتحسين التصميم. يجب أيضاً التأكد من توافق المخطط مع المتطلبات.

## **\*\*8. التوثيق:\*\***

- يجب توثيق المخطط العام بشكل دقيق. هذا يشمل كتابة وثائق توضح التفاصيل المهمة والقرارات المتخذة





# الفصل الثانى : تحليل المتطلبات والتصميم العميق

## • تحليل المتطلبات والتصميم العميق تحليل المتطلبات والتصميم العميق

يعتبر تحليل المتطلبات والتصميم العميق من أهم مراحل عملية تطوير البرمجيات والأنظمة التقنية. إنها المرحلة التي تسهم بشكل كبير في تحديد نجاح المشروع وفاعليته في تلبية احتياجات المستخدمين. يتطلب هذا الأمر فهماً عميقاً للمشكلة المراد حلها وضبط الهدف بدقة قبل البدء في كتابة الشفرة أو بناء النظام.

في هذا البرنامج، سنتناول بعمق أهمية تحليل المتطلبات والتصميم العميق ودورهما الحيوي في تنفيذ مشاريع تكنولوجيا المعلومات وتطوير البرمجيات.

أحد أهم جوانب تحليل المتطلبات هو فهم الاحتياجات والمتطلبات الوظيفية والغير وظيفية للنظام المراد بناؤه. يتعين على فريق التطوير التفاعل مع المستخدمين واستجوابهم بعناية لضمان توثيق كل التفاصيل الضرورية. يشمل ذلك متطلبات الأداء، والأمان، والاستقرار، وسهولة الصيانة، والتوافق مع أنظمة أخرى إذا كان ذلك ضرورياً.

بعد جمع المتطلبات، يأتي دور التصميم العميق. هذه المرحلة تهدف إلى تحويل المتطلبات إلى تصميم فني يمكن تنفيذه. يشمل ذلك تحديد هيكل النظام، واختيار التقنيات المناسبة، وتحديد الواجهات والمكونات الرئيسية. يسهم التصميم العميق في تحديد كيفية تنظيم الشفرة وتجزئتها إلى وحدات منفصلة قابلة للإعادة استخدام.

تأتي أهمية التحليل والتصميم العميق في تقليل مخاطر الفشل في مشروع تطوير البرمجيات. فهم جيد للمتطلبات والتصميم الجيد يقللان من احتمالية العثور على مشاكل كبيرة في مراحل لاحقة من عملية التطوير. بالإضافة إلى ذلك، يمكن أن يقلل التحليل والتصميم الجيدين من التكاليف والجهد اللازمين لإصلاح الأخطاء والتعديلات في مراحل متقدمة من المشروع.

يمكن القول إن تحليل المتطلبات والتصميم العميق يشكلان الأساس القوي لأي مشروع تطوير تقني ناجح. يجب على فرق التطوير الاستثمار في هاتين المرحلتين بعناية كبيرة لضمان تنفيذ مشروع موفق يلبي احتياجات المستخدمين ويتماشى مع المتطلبات الفنية والتقنية.

### ○ تفصيل متطلبات المشروع

تفصيل متطلبات المشروع هو عملية حاسمة في مجال تصميم البرمجيات. إنها تهدف إلى فهم وتوثيق كل جوانب المشروع بدقة ووضوح قبل البدء في مراحل التنفيذ. فيما يلي شرح مفصل لكيفية تحقيق هذا الهدف في مجال تصميم البرمجيات:

## 1. المحادثات والاستجابات:

- يبدأ تفصيل متطلبات المشروع بالمحادثات المكثفة مع مالك المشروع أو العميل. يجب فهم أهدافهم ورؤيتهم بشكل جيد.
- يجب تحديد الوظائف الأساسية والمتطلبات الوظيفية التي يجب تنفيذها في البرمجيات.

## 2. استخدام الوثائق:

- يمكن اللجوء إلى الوثائق القائمة مثل وثائق العميل السابقة (إذا كانت متوفرة) للتعرف على تفاصيل المشروع السابقة.
- يمكن استخدام الوثائق الأخرى مثل الدراسات السابقة والتقارير للتعرف على احتياجات المشروع.

## 3. تحليل العمليات:

- يجب دراسة وتحليل العمليات التي ستكون جزءًا من البرمجيات. هذا يشمل تمييز مراحل العمل ومتطلبات البيانات وتدفق المعلومات.
- يمكن استخدام تقنيات مثل رسم الخرائط الذهنية ( Mind Mapping ) وتحليل العمليات لفهم السياق بشكل أفضل.

## 4. تحديد المتطلبات الوظيفية:

- يجب تحديد متطلبات الوظائف الأساسية والفرعية بدقة.
- هذه المتطلبات تصف ما يجب أن تقوم به البرمجيات.
- يمكن استخدام القوائم والمخططات والرسوم التوضيحية لتوثيق هذه المتطلبات.

## 5. تحديد المتطلبات غير الوظيفية:

- يجب أيضًا التركيز على المتطلبات غير الوظيفية مثل الأمان والأداء وسهولة الصيانة وتوافق النظام.
- يمكن تحديد هذه المتطلبات باستخدام معايير ومؤشرات محددة.

## 6. اقتراح حلول:

- بناءً على تحليل المتطلبات، يمكن اقتراح حلول ممكنة. هذا يشمل اختيار التقنيات والمكونات البرمجية المناسبة.
- يجب أن تتضمن الحلول تفاصيل فنية تساعد في توجيه فريق التنفيذ.

## 7. التوثيق:

- يجب توثيق جميع المتطلبات والحلول بشكل دقيق ومفهوم. هذا يسهل التواصل مع فريق التنفيذ والعميل.
- يمكن استخدام أدوات إدارة المشاريع والتوثيق مثل مستندات SRS (Software Requirements Specification).

## 8. التحقق والتقييم:

- يجب أن يشمل تفصيل المتطلبات جلسات تحقق ومراجعات دورية للتأكد من توافق البرمجيات مع المتطلبات.
- يتعين إجراء تقييمات دورية لضمان أن المتطلبات لا تتغير بشكل غير متوقع.

تفصيل متطلبات المشروع في مجال تصميم البرمجيات يسهم بشكل كبير في تحقيق نجاح المشروع وضمان تلبية احتياجات العميل بشكل كامل وموثوق. يجب الاهتمام بتوثيق ومتابعة المتطلبات على مدى مراحل تنفيذ المشروع لتحقيق الأهداف المحددة بنجاح.

### ○ تحليل المتطلبات الوظيفية وغير الوظيفية

تحليل المتطلبات الوظيفية وغير الوظيفية هو عملية أساسية في مجال تصميم البرمجيات وتطوير الأنظمة التكنولوجية. تهدف هذه العملية إلى فهم وتحديد كل ما يتعلق بالمشروع من متطلبات ومتطلبات وظيفية وغير وظيفية بدقة ووضوح. فيما يلي شرح مفصل للمتطلبات الوظيفية وغير الوظيفية وأهميتهما:

#### 1. المتطلبات الوظيفية:

- المتطلبات الوظيفية تركز على ما يجب أن تقوم به البرمجيات أو النظام. إنها تحدد الوظائف الأساسية التي يجب تنفيذها والأنشطة التي يجب أن يتمكن المستخدمون من القيام بها.
- على سبيل المثال، في تطبيق البريد الإلكتروني، يمكن أن تكون متطلبات وظيفية تتضمن إمكانية إرسال واستقبال الرسائل الإلكترونية، وإرفاق الملفات، وتصنيف وتصفية الرسائل.

#### 2. المتطلبات غير الوظيفية:

- المتطلبات غير الوظيفية تركز على كيفية أداء النظام أو البرمجيات بدلاً من ماذا يفعل. إنها تتعلق بجوانب مثل الأمان،

والأداء، والاستقرار، والتوافق، وسهولة الصيانة، والتواصل مع المستخدمين.

- على سبيل المثال، متطلبات غير وظيفية قد تتضمن أن يتم تنفيذ عملية البحث في تطبيق ويب في أقل من ثانية واحدة أو أن يكون النظام متوافقًا مع متصفحات معينة.

أهمية تحليل المتطلبات الوظيفية وغير الوظيفية:

- تضمن المتطلبات الوظيفية تحديد ما ينبغي تنفيذه ويوجه الفريق المطور نحو بناء الوظائف الأساسية التي يحتاجها المستخدمون.
- المتطلبات غير الوظيفية تضمن تحقيق جودة النظام وأدائه العالي وضمان أنه يتوافق مع المعايير والمتطلبات الأمنية.
- تساعد في تحديد القيود والتحديات التي يمكن مواجهتها خلال عملية التطوير وتجنب المشاكل المحتملة.
- توفر توثيقًا دقيقًا يمكن استخدامه للتواصل مع العملاء والفرق التقنية وضمان تفهم وتنفيذ المشروع بشكل سليم.
- تساهم في تحسين عمليات الاختبار والتحقق من تلبية المتطلبات وتتبع أداء النظام.

في الختام، تحليل المتطلبات الوظيفية وغير الوظيفية يعد أساسيًا لضمان تصميم وتنفيذ البرمجيات أو الأنظمة التكنولوجية بشكل يلبي احتياجات المستخدمين ويضمن الجودة والأمان والأداء الجيد.

○ إعداد وثائق المتطلبات

إعداد وثائق المتطلبات هو عملية أساسية في مجال تصميم البرمجيات وتطوير الأنظمة التكنولوجية. تتمثل هذه العملية في توثيق جميع المتطلبات الوظيفية وغير الوظيفية التي يجب تلبيتها من قبل البرمجيات أو النظام الذي سيتم تطويره. إليك شرح مفصل لأهمية وإجراءات إعداد وثائق المتطلبات:

### **\*\*أهمية إعداد وثائق المتطلبات:\*\***

1. **\*\*توضيح الأهداف:\*\*** تساعد وثائق المتطلبات في توضيح أهداف المشروع والغايات المحددة التي يجب تحقيقها من خلال التصميم والتنفيذ. هذا يساعد على توجيه جميع الأعضاء في الفريق نحو الهدف المشترك.
2. **\*\*تحديد نطاق المشروع:\*\*** تساعد الوثائق في تحديد نطاق المشروع بدقة. يشمل ذلك ما يجب تضمينه وما يجب استبعاده من البرمجيات أو النظام.
3. **\*\*تبسيط التواصل:\*\*** توفر وثائق المتطلبات وسيلة للتواصل بين العميل وفريق التطوير. يمكن للعميل مراجعة الوثائق لضمان أن متطلباته تم توثيقها بشكل صحيح.
4. **\*\*تحديد المخاطر:\*\*** تساعد عملية إعداد الوثائق في تحديد المخاطر المحتملة والتحديات التي يمكن أن تواجه المشروع. هذا يمكن من اتخاذ إجراءات مناسبة للتعامل معها مسبقًا.



5. **\*\*سهولة التحقق والاختبار:\*\*** تمثل وثائق المتطلبات الأساس للتحقق واختبار البرمجيات أو النظام. يمكن لفريق الاختبار استخدام هذه الوثائق للتحقق من أن النظام يلبي المتطلبات بشكل صحيح.

**\*\*إجراءات إعداد وثائق المتطلبات:\*\***

1. **\*\*جمع المتطلبات:\*\*** يجب أن يبدأ العملية بجمع المتطلبات من العميل أو المستخدمين المستقرين. هذا يشمل الاستماع إلى احتياجاتهم ومتطلباتهم وتوثيقها بعناية.

2. **\*\*تحليل المتطلبات:\*\*** بعد جمع المتطلبات، يتعين تحليلها بعمق لضمان تفهمها بشكل كامل. يتضمن ذلك تحليل الوظائف والعلاقات بين المتطلبات وتحديد الأولويات.

3. **\*\*توثيق المتطلبات:\*\*** يجب كتابة وثائق المتطلبات بشكل دقيق وواضح. يشمل ذلك استخدام قوائم وجداول ورسومات توضيحية عند الضرورة.

4. **\*\*التحقق والتراجع:\*\*** يتعين على العميل مراجعة وثائق المتطلبات والتحقق من صحتها واكتمالها. يمكن أن تتطلب هذه العملية تعديلات وتحسينات.

5. **\*\*إدارة التغيير:** في حالة وجود تغييرات في المتطلبات أثناء تطوير المشروع، يجب توثيقها ومتابعتها بعناية لضمان توافقها مع الأهداف الرئيسية للمشروع.

6. **\*\*التوثيق المستمر:** يجب أن تكون عملية إعداد وثائق المتطلبات عملية مستمرة طوال مدى حياة المشروع. تحتاج الوثائق إلى التحديث بمرور الوقت لتعكس أي تغييرات أو تحسينات في المتطلبات.

لذلك فإن إعداد وثائق المتطلبات هو عملية أساسية تسهم في نجاح المشروع وضمان توافق البرمجيات أو النظام مع احتياجات المستخدمين والمعايير والأهداف المحددة.

### • التصميم التفصيلي

التصميم التفصيلي هو مرحلة مهمة في عملية تطوير البرمجيات وتصميم الأنظمة التكنولوجية. تأتي هذه المرحلة بعد مرحلة تحليل المتطلبات، حيث يتم تحويل المتطلبات الوظيفية وغير الوظيفية إلى تصميم تفصيلي يمكن تنفيذه.

فيما يلي شرح مفصل للتصميم التفصيلي وأهميته:

1. **\*\*تحويل المتطلبات إلى تصميم قابل للتنفيذ:** في مرحلة تصميم المتطلبات الوظيفية، يتم توثيق ما يجب أن تقوم به البرمجيات أو النظام. ومع ذلك، لا يتم تحديد كيفية تنفيذ هذه المتطلبات بالضبط. في

التصميم التفصيلي، يتم تحويل هذه المتطلبات إلى تصميم تقني يشمل الهيكل والتفاصيل التقنية التي تمكن من تنفيذ المتطلبات بشكل فعال وفعال.

2. **\*\*تحديد هيكل النظام:** \*\* يشمل التصميم التفصيلي تحديد هيكل النظام والطريقة التي سيتم بها تنظيم المكونات والوحدات البرمجية. هذا يشمل أيضًا تحديد كيفية تدفق البيانات والمعلومات بين هذه المكونات.

3. **\*\*اختيار التقنيات والأدوات:** \*\* يتعين على فريق التصميم اختيار التقنيات والأدوات المناسبة لتنفيذ المشروع. يشمل ذلك اختيار اللغة البرمجية، وقواعد البيانات، والأطر البرمجية (Frameworks)، وأي أدوات تطوير أخرى ضرورية.

4. **\*\*تحديد واجهات المستخدم:** \*\* إذا كان المشروع يتضمن واجهة مستخدم، يتم تصميم تفاصيلها بالكامل في هذه المرحلة. ذلك يتضمن تحديد العناصر البصرية، وترتيبها، ووضع الأزرار والحقول، وتحديد كيفية التفاعل مع المستخدم.

5. **\*\*إعداد مستندات التصميم:** \*\* يجب توثيق التصميم التفصيلي بشكل دقيق وشامل. هذا يتيح لأعضاء الفريق الآخرين فهم كيفية تنفيذ المشروع والالتزام بالمعايير والأفضليات المحددة.

6. **\*\*الاهتمام بالأداء والأمان:\*\*** يجب أن يتم التفكير بعناية في مسائل الأمان والأداء في التصميم التفصيلي. يمكن تحديد كيفية حماية النظام من التهديدات الأمنية وتحقيق الأداء المطلوب.

بشكل عام، التصميم التفصيلي يمثل الجسر الذي يربط بين تحليل المتطلبات وتنفيذ البرمجيات أو النظام. إنه يجعل المتطلبات قابلة للتنفيذ ويمكن تنفيذها بفعالية وكفاءة. وبفضل التصميم التفصيلي، يمكن لفريق التنفيذ بناء البرمجيات أو النظام بناءً على أسس قوية وواضحة.

### ○ تحليل الهندسة المعمارية

تحليل الهندسة المعمارية هو عملية مهمة في مجال تصميم وتطوير البرمجيات والأنظمة التكنولوجية. تتمثل هذه العملية في فحص وتحليل الهيكل والتصميم العام للنظام أو البرمجيات المخططة للبناء أو التطوير. يهدف التحليل المعماري إلى تحديد كيفية تنظيم المكونات والأجزاء الفرعية للنظام بشكل فعال وكفاءة.

فيما يلي شرح مفصل لتحليل الهندسة المعمارية وأهميته:

### **\*\*أهمية تحليل الهندسة المعمارية:\*\***

1. **\*\*ضمان الاستدامة والتوسع:\*\*** يساعد التحليل المعماري في تصميم نظام يكون قابلاً للتطوير والتوسع في المستقبل دون الحاجة إلى تغييرات كبيرة في الهيكل. هذا يقلل من تكاليف الصيانة والتحديث.

2. **\*\*تحسين الأداء:** \*\* يمكن تحسين أداء النظام من خلال التحليل المعماري بتحديد النقاط الضعيفة والمواضع التي يمكن تحسينها. يمكن تحقيق أفضل استجابة وأداء عندما يتم تصميم الهيكل بعناية.

3. **\*\*توجيه القرارات التقنية:** \*\* يمكن لتحليل الهندسة المعمارية توجيه القرارات التقنية المهمة بشأن اختيار التقنيات والأدوات المناسبة للمشروع. يمكن أن يساعد في تقدير التكلفة والجدوى الفنية للاختيارات المختلفة.

4. **\*\*تحسين الأمان:** \*\* يمكن تحسين أمان النظام من خلال التحليل المعماري بتحديد وتقييم المخاطر الأمنية وتنفيذ الإجراءات اللازمة للوقاية منها.

5. **\*\*تحسين إعادة الاستخدام:** \*\* يمكن تصميم الهيكل المعماري ليكون قابلاً لإعادة الاستخدام، مما يسمح بمشاركة المكونات والوظائف بين مشاريع مختلفة وتقليل الجهد والتكلفة في المستقبل.

**\*\*إجراءات تحليل الهندسة المعمارية:** \*\*

1. **\*\*فحص الهيكل الحالي:** \*\* يتم البدء عادة بفحص الهيكل الحالي للنظام أو البرمجيات (إذا كان موجوداً) لفهم كيفية تنظيم المكونات والتفاعلات بينها.

2. **\*\*توثيق الهيكل العام:** \*\* يتم توثيق الهيكل العام المخطط للنظام أو البرمجيات باستخدام مخططات ورسوم بيانية ووثائق توضيحية.

3. **\*\*تحليل الاحتياجات المستقبلية:** \*\* يجب أخذ الاحتياجات المستقبلية في الاعتبار وتضمينها في التصميم المعماري لضمان استدامة النظام.

4. **\*\*اختبار الأداء والأمان:** \*\* يتم إجراء اختبارات أداء وأمان لضمان أن التصميم المعماري يلبي المعايير والاحتياجات المحددة.

5. **\*\*توثيق وتقرير:** \*\* يجب توثيق نتائج التحليل المعماري وإعداد تقارير توضح النتائج والتوصيات لتحسين الهيكل والأداء.

في النهاية، تحليل الهندسة المعمارية هو عملية أساسية لضمان تصميم نظام أو برمجيات تكنولوجية يتميز بالكفاءة والاستدامة والأمان والأداء الجيد. يمكن أن يس

هم هذا التحليل في تجنب مشاكل التصميم والتطوير في وقت لاحق وتحقيق النجاح في المشاريع التكنولوجية.

# الفصل الثالث: تنفيذ ونشر

## البرمجيات



## • تنفيذ ونشر البرمجيات

تنفيذ ونشر البرمجيات هو عملية مهمة في مجال تطوير البرمجيات وتكنولوجيا المعلومات. تشمل هذه العملية العديد من الخطوات والتفاصيل التي يجب أن تتم بعناية لضمان توفير برمجيات عالية الجودة وتحقيق أهداف العمل.

الخطوات الرئيسية لتنفيذ ونشر البرمجيات تتضمن:

### 1. تحليل المتطلبات:

- يجب أن يبدأ العمل بتحليل المتطلبات البرمجية. يجب على المطورين والمهندسين أن يتفاعلوا مع العملاء أو المستخدمين المستهدفين لفهم احتياجاتهم بدقة.

### 2. التصميم:

- بناءً على المتطلبات المحددة، يتم تصميم هيكل وواجهة البرمجيات. يجب على المصممين أن يضمنوا تصميم نظام يلبي المتطلبات وسهل الاستخدام.

### 3. التطوير:

- يتم برمجة البرمجيات بناءً على التصميم المعتمد. يشمل هذا الخطوة كتابة الشيفرة واختبارها بشكل متكرر لضمان أداء سليم.

### 4. اختبار واعتماد:

- يجب إجراء اختبارات متعددة للتأكد من أن البرمجيات تعمل بشكل صحيح وتلبي المتطلبات. يمكن أن تشمل هذه الاختبارات اختبارات ودية واختبارات تكاملية واختبارات أداء.

### 5. توثيق:

- يتم إعداد وثائق توثيقية تفصيلية للبرمجيات. تشمل هذه الوثائق كيفية استخدام البرمجيات وصيانتها ومعلومات تقنية أخرى.

### 6. نشر:

- بعد اجتياز البرمجيات لجميع مراحل الاختبار والتحقق من جودتها، يتم نشرها. يمكن أن يتم نشر البرمجيات على الخوادم السحابية أو الخوادم المحلية أو توزيعها على وسائط مثل الأقراص أو تنزيلها عبر الإنترنت.

### 7. صيانة وتحديث:

- بعد النشر، يجب متابعة البرمجيات وإصدار تحديثات لتحسين الأداء وإصلاح الأخطاء البرمجية. هذه الصيانة تضمن استمرارية عمل البرمجيات بشكل صحيح.



8. دعم المستخدمين:

- يجب تقديم دعم فني وتدريب للمستخدمين لضمان فهمهم لكيفية استخدام البرمجيات وحل المشكلات التي قد تظهر.

تنفيذ ونشر البرمجيات هو عملية معقدة تتطلب تنسيقاً جيداً بين مختلف الفرق والمهارات المتخصصة في مجال تطوير البرمجيات. يتعين على الفرق العمل بجد لضمان توفير منتج نهائي يلبي احتياجات المستخدمين ويتفوق في سوق التكنولوجيا المتنافس.

### • تكوين الخوادم وقواعد البيانات

تكوين الخوادم وقواعد البيانات هو عملية مهمة في مجال تكنولوجيا المعلومات وتطوير البرمجيات. تتضمن هذه العملية تحضير وتهيئة الأجهزة الخادمة وقواعد البيانات بحيث تكون قادرة على استضافة وإدارة تطبيقات البرمجيات بشكل فعال وآمن. فيما يلي شرح مفصل لهذه العملية:

#### 1. \*\*اختيار الأجهزة والبنية التحتية:\*\*

- أول خطوة في تكوين الخوادم وقواعد البيانات هي اختيار الأجهزة والبنية التحتية التي ستتم استخدامها. يتعين على الفريق تحديد ما إذا كان سيتم استخدام خوادم في مركز البيانات الخاص بالشركة أم سيتم الاعتماد على خوادم سحابية (Cloud) مثل AWS أو Azure.

#### 2. \*\*تثبيت وتكوين نظام التشغيل:\*\*

- يجب تثبيت نظام التشغيل المناسب على الخادم. سيختلف نوع نظام التشغيل بناءً على احتياجات التطبيق واختيارات التكنولوجيا. على سبيل المثال، Linux و Windows Server هما من أنظمة التشغيل الشائعة.

#### 3. \*\*تثبيت وتكوين البرمجيات الخادمة:\*\*

- بعد تثبيت نظام التشغيل، يتعين تثبيت وتكوين البرمجيات الخادمة المطلوبة مثل خوادم الويب (مثل Apache أو Nginx) وخوادم قواعد البيانات (مثل MySQL أو PostgreSQL) وخوادم التطبيقات (مثل Tomcat أو Node.js).

#### 4. \*\*ضبط الأمان:\*\*

- يجب تكوين إعدادات الأمان على الخادم لحمايته من الهجمات الإلكترونية. ذلك يتضمن تكوين جدران الحماية (Firewalls) وتثبيت برامج مكافحة الفيروسات وتكوين شهادات SSL لتأمين الاتصالات.

## 5. \*\*تكوين قواعد البيانات:\*\*

- إذا كان التطبيق يتطلب قاعدة بيانات، يتعين إنشاء وتكوين قاعدة البيانات بمعلومات الاتصال والجدول والفهارس المطلوبة. يجب أيضًا تنفيذ عمليات النسخ الاحتياطي وضبط أدونات الوصول بحيث تكون محمية وفعالة.

## 6. \*\*اختبار ومراقبة:\*\*

- بعد تكوين الخوادم وقواعد البيانات، يجب إجراء اختبارات لضمان أنها تعمل بشكل صحيح وفقًا للمتطلبات. يتم أيضًا إعداد أنظمة مراقبة لتتبع أداء الخوادم والكشف عن أي مشاكل محتملة.

## 7. \*\*توثيق:\*\*

- يجب وثائق كافية لتكوين الخوادم وقواعد البيانات بما في ذلك المعلومات الفنية وإعدادات الأمان والإجراءات الضرورية للصيانة والتحديث.

تكوين الخوادم وقواعد البيانات هو جزء أساسي من إطلاق التطبيقات والخدمات على الإنترنت. يتطلب الأمر خبرة واهتماماً بالتفاصيل لضمان أن البيئة الخادمة جاهزة لاستقبال وخدمة المستخدمين بشكل فعال وآمن.

## ● صيانة وحماية البيانات

صيانة وحماية البيانات هي عملية أساسية في مجال تكنولوجيا المعلومات وإدارة المعلومات لضمان سلامة وسرية وتوافر البيانات. يتعلق هذا الأمر بحفظ وصيانة المعلومات الحساسة والضرورية للشركات والمؤسسات بحيث تكون متاحة للوصول السريع والفعال وفي نفس الوقت محمية من فقدان والاختراق والتلف. فيما يلي شرح مفصل للمفاهيم المتعلقة بصيانة وحماية البيانات:

### 1. \*\*صيانة البيانات:\*\*

- صيانة البيانات تعني العمليات والممارسات التي تهدف إلى الحفاظ على نوعية واتساق البيانات على مدى الوقت. تشمل هذه العمليات:

- \*\*النسخ الاحتياطي (Backup):\*\* إنشاء نسخ احتياطية من البيانات بانتظام لحمايتها من فقدان أو تلف غير متوقع.

- \*\*الاستعادة (Recovery):\*\* قدرة استعادة البيانات من النسخ الاحتياطية في حالة حدوث مشكلة.

- \*\*تحديث البيانات (Data Updates):\*\* تحديث البيانات بانتظام بمعلومات جديدة أو مصادر خارجية.

- \*\*تنظيف البيانات (Data Cleaning):\*\* التحقق وإصلاح البيانات التي يمكن أن تكون غير دقيقة أو مكررة.

- \*\*مراقبة البيانات (Data Monitoring):\*\* مراقبة البيانات لاكتشاف أي تغييرات غير مرغوب فيها أو أنشطة غير معتادة.

## 2. \*\*حماية البيانات:\*\*

- حماية البيانات تشمل جملة من الإجراءات والتقنيات التي تهدف إلى منع الوصول غير المصرح به إلى البيانات وضمان سرية المعلومات. تشمل هذه الإجراءات:
  - \*\*التشفير (Encryption):\*\* استخدام تقنيات التشفير لتحويل البيانات إلى شكل غير قابل للقراءة إلا بوجود مفتاح تشفير.
  - \*\*إدارة الهوية والوصول (Identity and Access Management - IAM):\*\* تنظيم منح الوصول إلى البيانات والتحكم فيه باستخدام سياسات وأدوات.
  - \*\*الجدران النارية (Firewalls):\*\* استخدام الجدران النارية لحماية البيانات من الهجمات الخارجية وتصفية حركة المرور.
  - \*\*برامج مكافحة الفيروسات والبرامج الضارة (Antivirus and Anti-Malware Software):\*\* استخدام برامج مكافحة الفيروسات والبرامج الضارة للكشف والوقاية من البرامج الخبيثة.
  - \*\*تحديثات الأمان (Security Patching):\*\* تحديث البرامج والأنظمة بانتظام لسد الثغرات الأمنية المعروفة.
  - \*\*التدقيق والمراقبة (Auditing and Monitoring):\*\* مراقبة النشاطات على النظام لاكتشاف أنشطة غير مصرح بها.

## 3. \*\*الامتثال والتشريعات:\*\*

- يجب على المؤسسات الامتثال للقوانين واللوائح المتعلقة بحماية البيانات. على سبيل المثال، قانون الحماية العامة للبيانات (GDPR) في الاتحاد الأوروبي يفرض متطلبات صارمة على حماية البيانات الشخصية.

## 4. \*\*تدريب الموظفين:\*\*

- يجب تدريب الموظفين على أفضل الممارسات لحماية وصيانة البيانات. هم عادةً يكونون أول خط الدفاع ضد التهديدات الأمنية.

صيانة وحماية البيانات أمور حيوية لأمان الأعمال وسمعتها. تجمع بين الاهتمام بتوفير البيانات والتأكد من عدم تعرضها للمخاطر الأمنية، وهي مسؤولية تتطلب استراتيجيات وتكنولوجيا متطورة للتعامل مع تحديات العصر الرقمي.

● **الذكاء الاصطناعي وتعلم الآلة وكيفية دمجهما في التصميم**

الذكاء الاصطناعي (AI) وتعلم الآلة (Machine Learning) هما مفاهيم أساسية في مجالات تكنولوجيا المعلومات والبرمجة تلعب دورًا مهمًا في تحسين الأنظمة والتطبيقات. فيما يلي شرح مفصل لكل من هذه المفاهيم وكيفية دمجهما في التصميم:

## 1. \*\*الذكاء الاصطناعي (AI):\*\*

- الذكاء الاصطناعي هو مجموعة من التقنيات والبرامج التي تهدف إلى إعطاء الأنظمة الحاسوبية القدرة على تنفيذ مهام تتطلب تفكيرًا ذكيًا. يتضمن ذلك تقنيات مثل معالجة اللغة الطبيعية والتعرف على الصوت والرؤية الحاسوبية والتخطيط واتخاذ القرار. تستخدم الذكاء الاصطناعي لإنشاء أنظمة يمكنها تحليل البيانات واتخاذ قرارات مستنيرة بناءً على البيانات والسياق.

## 2. \*\*تعلم الآلة (Machine Learning):\*\*

- تعلم الآلة هو فرع من الذكاء الاصطناعي يركز على تطوير نماذج وخوارزميات تمكن الأنظمة الحاسوبية من تعلم من البيانات وتحسين أدائها تلقائيًا دون برمجة صريحة. يتيح تعلم الآلة للأنظمة تحليل البيانات واكتساب الخبرة منها لاتخاذ قرارات دقيقة. تشمل تقنيات تعلم الآلة الشبكات العصبية الاصطناعية والتصنيف والتجميع والتنبؤ.

## 3. \*\*دمج الذكاء الاصطناعي وتعلم الآلة في التصميم:\*\*

- دمج الذكاء الاصطناعي وتعلم الآلة في التصميم يمكن أن يعزز بشكل كبير أداء وقدرات الأنظمة والتطبيقات. إليك كيفية دمجها بنجاح:

- **\*\*تحليل البيانات الذكي:\*\*** يمكن استخدام تقنيات تعلم الآلة لتحليل البيانات بشكل ذكي واستخراج أنماط وتوجيهات مهمة. مثلاً، في تصميم تطبيق للتسوق عبر الإنترنت، يمكن استخدام تعلم الآلة لتوصية المنتجات بناءً على تفضيلات المستخدم وتاريخ مشترياته السابقة.

- **\*\*التفاعل الذكي مع المستخدمين:\*\*** يمكن استخدام الذكاء الاصطناعي لتطوير واجهات مستخدم ذكية تستجيب للتفاعلات البشرية بطريقة تجعل التجربة أكثر سلاسة وشخصية. مثلاً، مساعد صوتي مثل Siri أو Alexa يعتمد على تقنيات تعلم الآلة لفهم الأوامر الصوتية والرد عليها بشكل دقيق.

- **\*\*تحسين أمان الأنظمة:\*\*** يمكن استخدام تعلم الآلة في اكتشاف ومكافحة التهديدات الأمنية. يمكن للأنظمة المدعومة بالذكاء الاصطناعي تحليل سلوك المستخدمين واكتشاف الأنشطة غير المعتادة التي قد تشير إلى هجمات.

- **\*\*تحسين إدارة الموارد:\*\*** يمكن استخدام الذكاء الاصطناعي وتعلم الآلة لتحسين إدارة الموارد في البيئات المعقدة مثل السلاسل اللوجستية أو أنظمة توزيع الطاقة.

- \*\*توجيه السيارات الذاتية:\*\* في صناعة السيارات، يمكن استخدام الذكاء الاصطناعي وتعلم الآلة لتطوير نظم القيادة الذاتية والتنبؤ بسلوك المركبات.

في الختام، دمج الذكاء الاصطناعي وتعلم الآلة في التصميم يتيح إمكانيات كبيرة لتطوير تطبيقات وأنظمة أكثر ذكاءً وفاعلية. يتطلب الأمر فهمًا عميقًا لكلا النهجين واستخدامهما بشكل مناسب لتحقيق الأهداف المطلوبة في التصميم.

المحور الثاني: التطبيق العملي

مقدمه عن التطبيقات العمليه

والخوارزميات

## 1- مقدمة :

إن الحاسب الآلي كل متكامل عتاد وبرمجيات فبدون أي جزء لا يمكن أن يعمل ويحقق الغاية المرجوة منه ، وبالتالي فإن البرمجيات لا تقل أهمية عن العتاد وتلقى البرمجيات اهتماماً كبيراً في العالم المعلوماتي المعاصر وتشهد سوقها منافسة قوية .  
وتصميم البرمجيات ينطلق من الخوارزميات والمخططات التدفقية حيث يشكلان بداية الطريق لخلق أي برنامج ، ولننطلق نحن لإيضاح هذه المفاهيم وكيفية التعامل معها ولكن لنستعرض أولاً خطوات حل مسألة باستخدام الحاسب .

## 2- ما هي المراحل التي نمر بها لحل مسألة ؟

نمر بخمس مراحل لحل مسألة وهي :

### 1-2 تعريف وتحليل المسألة :

تعريف المسألة يعني دقة في التعبير في تطبيق المسألة بحيث تصبح مفهومة بصورة واضحة دون لبث فيه لجميع من يعمل ضمن الاختصاص الذي تتناوله المسألة ، أما تحليلها ووضع الحل فهنا لب القضية وقد نعاني صعوبات في ذلك حيث يمكن أن نستخدم كثير من القوانين والطرق الرياضية المناسبة للحل وقد نضطر لتطويعها لتتناسب مع المسألة المدروسة وفي هذه الخطوة يجب تحديد ما يلي :

1- طبيعة الخرج (OUTPUT) ( النتائج ) وتنظيمها .

2- الدخول (INPUT) ( المعطيات أو المعلومات ) وتحديد نوعها وتنظيم إدخالها .

3- طرق الحل المناسبة وتقييمها بما يتلاءم مع طريقة تنفيذها وفي ضوء ذلك نختار الأفضل.

### 2-2 وضع الحل التخطيطي ( سرد خطوات وبياني ) :

نقوم هنا بالتعبير عن الحل أو الحلول التي استنتجت سابقاً على شكل خطوات متسلسلة ومترابطة منطقياً ودقيقة الوصف للوصول إلى الحل وهي ما تدعى بالخوارزمية وقيامنا بعد ذلك بوضع هذه الخطوات ضمن مخطط بياني مستخدمين مجموعة من الأشكال الاصطلاحية



والرموز نكون قد حصلنا على المخطط التدفقي لحل المسألة ويدعى أيضاً مخطط سير العمليات أو المخطط النهجي .

## 2-3- كتابة الكود البرمجي :

ليتمكن الحاسب من فهم هذا الحل يجب تحويله إلى لغة يفهمها وبالتالي يتم تحويل الحل التخطيطي إلى كود برمجي مكتوب بإحدى لغات البرمجة المعروفة ويسمى عندئذ بالبرنامج المصدر .

## 2-4- ترجمة البرنامج المصدري :

يتم ذلك بإدخال البرنامج إلى الحاسب وترجمته إلى لغة الآلة بوساطة برنامج الترجمة الخاص بلغة البرمجة المستخدمة وذلك في حال عدم وجود أخطاء في البرنامج المصدر وتمر عملية الترجمة بالمراحل التالية :

1- مرحلة التحليل المعجمي : يتم فيها مطابقة مفردات برنامج المصدر والعلاقات والأسماء مع تلك المسموح بها في لغة البرمجة المستخدمة واكتشاف الأخطاء فيها ، إن وجدت .

2- مرحلة التحليل اللغوي والنحوي : يتم فيها مطابقة تعليمات برنامج المصدر مع القواعد اللغوية للغة المستخدمة ، واكتشاف الأخطاء فيه إن وجدت .

3- مرحلة ترجمة البرنامج إلى لغة الآلة : يتم تحويل البرنامج المصدر إلى برنامج بلغة التجميع ونحصل نحن على البرنامج الهدف الذي نستطيع تنفيذه .

## 2-5- تنفيذ البرنامج وتجربته :

يتم تجربة البرنامج الهدف الذي حصلنا عليه للتأكد من صحته منطقياً ، وذلك باستخدام عينة من البيانات الاختبارية فإذا ثبت صحتها نكون قد حصلنا على البرنامج المطلوب ، بأفضل صورة له وجاهز للتطبيق العملي على بيانات حقيقية واستثماره .

## 3- ما هو مفهوم الخوارزمية أو الألوغوريتم ؟

إن التعريف البسيط لكلمة خوارزمية ( Algorithm ) إنها طريقة أو خطة أو قاعدة للوصول اعتباراً من معطيات إلى نتائج ، ونستطيع صياغة تعريف آخر أكثر دقة كالتالي هي عبارة عن مجموعة من الخطوات المتسلسلة التي تصف بصورة مضبوطة وبدون أي غموض جميع الخطوات الرياضية والمنطقية اللازمة لحل مسألة ما ، وقد تطور هذا المفهوم وأصبح يعني طريقة أو خطة شاملة وعامة لحل مسألة ما ، و، نقوم بوصف كافة الخطوات بشكل مفصل ونقول عن طريقة حل مسألة بأنها خوارزمية إذا اتصفت بما يلي :

1- تحقق إمكانية وصف الطريقة المتبعة في حل المسألة بعدد من الخطوات التي تحوي تعاليم أو أوامر بشكل مفصل وصريح توضح فيها ما يجب عمله في كل خطوة بدون التباس فيه .

2- تحوي على عدد محدد من الخطوات توصلنا إلى النتائج انطلاقاً من المعطيات المتوفرة ، فوجود عدد لانهائي من الحلول لطريقة ما ، لا يمكننا من اعتبارها خوارزمية حل .

3- يجب أن تتضمن الخوارزمية جميع الشروط والاعتبارات في حل المسألة مهما اختلفت المعطيات التي تتناولها المسألة المطروحة .

ومما تجدر الإشارة إليه أن أي عمل نقوم به في حياتنا لإنجازه بالشكل الصحيح يخضع لخطة عمل محددة أي خوارزمية كما المسائل العلمية وللإيضاح نتناول مثالين أحدهما حياتي والآخر رياضي :

### مثال /1/ :

ما هي الخطوات ( الخوارزمية ) التي نتبعها لتناول وجبة في مطعم ؟  
الحل :

- 1- البداية .
- 2- الذهاب إلى المطعم .
- 3- اختيار مكان الجلوس .
- 4- طلب الوجبة .
- 5- تناول الوجبة .
- 6- استلام الفاتورة .
- 7- دفع الفاتورة .
- 8- مغادرة المطعم .
- 9- النهاية .

### مثال /2/ :

ما هي خوارزمية حل معادلة من الدرجة الأولى من الشكل (  $a + b x = c$  ) باعتبار  $a, b, c, x$  أعداد صحيحة وسيتم دراسة هذا المثال بشيء من التفصيل لتوضيح مفهوم الخوارزمية .

$$x = \frac{c - a}{b}$$

نلاحظ أن يمكن وضع المعادلة بالشكل :

وعلى ذلك فخطوات الحل تكون التالية:

1- ا طرح  $a$  من  $c$  وسمّ هذه القيمة  $m$

2- قسم  $m$  على  $b$  وسمّ هذه القيمة  $x$

3- اكتب قيمة  $x$

إن هذه الخوارزمية ليست دقيقة ولا تراعي كل شروط الحل وبالتالي تعطي نتائج خاطئة في حالتين :

1- عندما  $b$  تساوي الصفر : فإذا كانت  $a = c$  فإن  $x$  يمكن أن تأخذ أي قيمة وإذا كانت  $a \neq c$  أي لا توجد أية قيمة لـ  $x$  .

2- إذا لم تعط نتيجة قسمة  $m$  على  $b$  عدداً صحيحاً فعندها لا يوجد أي عدد صحيح يحقق المعادلة السابقة .

ولكي تصبح الطريقة خوارزمية دقيقة يجب أن تراعي الحالات الخاصة وتحقق شروط الخوارزمية وعلى ذلك نستطيع كتابة الحل الصحيح بالشكل التالي :

1- إذا كانت  $b = 0$  وكانت  $a = c$  فننفذ الخطوة السادسة من هذه الخوارزمية و إلا فتابع إلى الخطوة الثانية .

2- إذا كانت  $b = 0$  وكانت  $a \neq c$  فننفذ الخطوة الخامسة من هذه الخوارزمية و إلا فتابع إلى الخطوة الثالثة .

3- ا طرح  $a$  من  $c$  وسميها  $m$  .

4- قسم  $m$  على  $b$  وإذا كان هناك باقي بنتيجة القسمة فننفذ الخطوة الخامسة و إلا فاطبع النتيجة وتوقف .

5- اكتب " لا يوجد عدد صحيح يحقق المعادلة " ثم توقف .

6- اكتب " أي عدد صحيح يحقق المعادلة " ثم توقف .

وبالتالي نجد إن هذه الخوارزمية دقيقة وعامة وتحوي عدد محدد من الخطوات ومطلوب اتخاذ قرارات وإجراء مقارنات وإسناد قيم بمعنى وجود مفاهيم رياضية بسيطة يجب على قارئ الخوارزمية إدراكها إضافة لإدراك عملية الانتقال أو القفز من خطوة إلى أخرى وبالنهاية التوقف .

### 3-1- المتحولات :

المتحولات ( Variables ) عبارة عن مقادير تأخذ قيماً مختلفة وتخزن قيم المتحول في خلية من ذاكرة الحاسب وهذا يعني أننا عند تسمية متحول فإن الحاسب يخصص له خلية من ذاكرته معرفة باسمه وأن أية قيمة يأخذها هذا المتحول تخزن في خلية الذاكرة المعنونة باسمه وتزول القيمة السابقة في الخلية عند إسناد قيمة جديدة لها ونستطيع استخدام هذه القيمة بذكر

اسم المتحول ويمكننا تسمية هذه المتحولات بأي اسم نريده مثلاً حرف ( X , A , B , .... ) أو أكثر من حرف أو حتى كلمة والأفضل أن تحمل معنى مثل ( VALUE , SUM , .... ) ونصادف ثلاث أنواع من المتحولات :

1- المتحولات العددية .

2- المتحولات المحرفية .

3- المتحولات المنطقية .

تأخذ المتحولات العددية قيمة رقمية ويمكننا إجراء العمليات الحسابية من جمع وطرح وغيره عليها أما المحرفية فلا نستطيع إجراء العمليات الحسابية عليها إنما هي أسماء أو معطيات تصنيفية وحتى لو أشرنا إليها بأرقام فهي لا تخضع للعمليات الحسابية ، وبالنسبة للمتحولات المنطقية فهي تأخذ قيمتين " صح " ( TRUE ) و "خطأ" ( FALSE ) ونحتاجها في الاختيارات و الاختبارات وتطبق عليها عمليات الجبر البولي ( AND , OR , ..... ) وقد تعرضنا لفكرة عن المتحول كونه سيمر استخدامها في الخوارزميات والمخطط التدفقية وهي لا تخضع لقواعد هنا بعكس ما نجده في متحولات لغات البرمجة التي تخضع لقواعد لغة البرمجة المتعامل معها .

### 3-2- ما هي التعاليم والأوامر الرئيسية التي يتقبلها الحاسب ؟

لا يستطيع الحاسب أن يحل مسألة ما إلا إذا لقناه الحل بمنطق يتقبله ، و يعني هذا تلقينه الحل وفق تعليمات وأوامر ومحاكمات يستطيع فهمها والتعامل معها والتعاليم والأوامر الرئيسية التي يتقبلها هي :

1- قراءة عدد أو اسم وحفظه في ذاكرته وهذا يتم في خلية محددة ومعنونة من الذاكرة نستطيع أن نغير محتوى الخلية كما نريد في البرنامج ، إلا أن عنوان الخلية يبقى ثابتاً .

2- طبع عدد أو اسم موجود في خلية محددة من الذاكرة ويعني نقل المعلومات من الخلية المحددة إلى عنصر من عناصر الخرج .

3- القيام بالعمليات الحسابية على أن يتم تحديد نوع العملية ( جمع ، طرح ، ..... ) مع تخزين النتائج الانتقالية والنهائية في خلايا محددة من الذاكرة .

4- التوقف عن تنفيذ الأوامر والانتقال أو القفز من أمر إلى آخر في البرنامج .

5- القدرة على المحاكمة أي تقبل سؤال مطروح بشكل صريح بالبرنامج بشأن المعطيات أو إحدى النتائج التي توصل إليها على أن تكون الإجابة مقتصرة على ( نعم أو لا ) مع إيضاح ما يجب فعله في كل حالة من الحالتين .

### 3-3- ما هي الخوارزميات المبرمجة ؟

يمكننا أن نقول عن خوارزمية أنها مبرمجة ( قابلة للبرمجة ) إذا كنا قد وضعنا خطواتها بشكل مفصل ومنظم وفق منطق مترابط ويتقبله الحاسب وينسجم مع تكوينه أي

بمعنى آخر يجب وضع خطوات العمل في الخوارزمية المبرمجة باستخدام التعاليم والأوامر التي يتقبلها الحاسب وعند ذلك نستطيع تحويلها إلى برنامج بإحدى لغات البرمجة يلقن به الحاسب بسهولة .

#### 4- ما هي المخططات التدفقية ؟

جاءت المخططات التدفقية ( Flowcharts ) كضرورة لتسهيل عمل المبرمج عندما تتعد الخوارزمية خاصة أي تزداد خطواتها والمقارنات والمحاكمات فيها فالمخطط التدفقي هو تمثيل رسومي للخوارزمية الذي يعطينا تمثيلاً جيداً لها ويستخدم في هذه المخططات رموز وأشكال هندسية متنوعة لها دلالات محددة .

#### 4-1 كيف يتم كتابة المخططات التدفقية ؟

يتطلب كتابة مخطط تدفقي ، يحوي عمليات حسابية وشرط ومحاكمات تمريناً طويلاً وعلى المبتدئ أن يضع إمكانياته وقدراته لإيجاد الخطوات المناسبة التي تضمن خوارزمية صحيحة ومخطط تدفقي صحيح ، حقيقة الأمر أنه لا توجد طريقة عامة متبعة لكتابة خوارزمية مبرمجة و صياغة مخطط تدفقي ، ولذلك يجب دراسة كل مسألة على حدة ويمكن وضع عدة خوارزميات لمسألة واحدة نتوصل فيها لنفس النتائج ويفيد التمرين والخبرة في ذلك كثيراً .

#### 4-2 ما هي أهم فوائد استخدام المخططات التدفقية ؟

- 1- تساعد المبرمج على الإحاطة بالمسألة المراد حلها بشكل كامل والسيطرة على كل أجزائها بحيث يستفاد منها في اكتشاف الأخطاء المنطقية .
- 2- يصعب على المبرمج متابعة التفرعات الكثيرة التي تظهر في بعض البرامج بدون المخططات التدفقية .
- 3- تساعد المبرمج عند العمل على تعديل برنامج ما عند النظر إلى المخططات التدفقية فبنظرة سريعة على المخطط ندرك ماهية المسألة وإمكانية التعديل .
- 4- تعتبر المخططات التدفقية لحل مسألة معينة مرجعاً هاماً يجب الاحتفاظ به للعودة إليه عند الحاجة للتعديل أو الاستخدام في مسائل مشابهة .

#### 4-3- ما هي الرموز المستخدمة في تمثيل المخططات التدفقية ؟

سنقوم باستعراضها وإيضاحها حسب ما هو معتمد في المعهد الوطني الأمريكي ( AINSI ) :

##### 1- الإطار المستطيل المنتهي بنصفي دائرة :

يستخدم هذا الإطار للدلالة على نقطة بداية المخطط التدفقي أو نهايتها فنضع فيه إما كلمة البداية أو ( START ) في أول المخطط وكلمة النهاية أو توقف ( STOP/END ) عند نهاية المخطط ويجب أن يحوي المخطط على إطار واحد للبداية ولكنه قد يحوي على أكثر من إطار توقف في عدة أماكن من المخطط .

##### 2- الإطار المتوازي الأضلاع :

يستخدم للدلالة على قراءة المعطيات وطباعتها أي لعمليات الدخل والخرج وضمنه نعين أسماء وعناوين الخلايا التي تخزن فيها المعطيات عند تنفيذ الخوارزمية وكذلك الخلايا المطلوب طباعتها فمثلاً ( READ x,y ) ويعني أنه عند تنفيذ الأمر يجب قراءة متحولين ( عدد أو اسم ) وتخزينهما في خليتين معنوتين بـ ( x,y ) .

##### 3- الإطار المستطيل :

يستخدم لتحديد العمليات الحسابية وبيان الخلايا التي تخزن فيها نتائج العمليات الحسابية وهي ما تدعى بالأوامر أو التعاليم الحسابية في المخطط التدفقي فمثلاً (  $x = y + 5$  ) ويعني إضافة محتوى الخلية المعنونة y إلى العدد 5 وتخزين الناتج في الخلية المعنونة x .

##### 4- الإطار المعين :

يستخدم لوضع التساؤلات والاختبارات في المخطط التدفقي فعندما نكتب ضمن إطار معين  $x = y$  فهذا يعني هل محتوى x يساوي محتوى y ويجب أن يتفرع من إطار المعين مسارات قد تكون اثنان أو ثلاثة حسب حالة الاختبار ففي حالتنا السابقة مسارين لأن جواب السؤال سيكون إما نعم أو لا وثلاثة مسارات في حال كان السؤال  $x \leq y$  فكل مسار يمثل حالة.

##### ملاحظة :

إن وجود التعبير  $x = y$  في إطار مستطيل يعني احفظ أو خزن محتوى الخلية المعنونة y في الخلية المعنونة x بينما وجوده في الإطار المعين يعني هل x تساوي y .

##### 5- الإطار المستطيل المنتهي بنصفي معين :

يستخدم للتعبير عن حالات التكرار والحلقات مثلاً تكرار إعطاء قيمة لعداد I من 1 حتى n وتكتب بالشكل  $I = 1, n$  .

##### 6- الدائرة :

تستخدم لبيان وصل منطقة من المخطط التدفقي مع منطقة ثانية ويوضع داخلها رقم أو حرف ونفس الرقم أو الحرف في المنطقة الأخرى المراد القفز إليها .

جدول يبين أشكال الرموز المستخدمة في المخططات التدفقية

اسم الرمز	شكل الرمز
الإطار المستطيل المنتهي بنصفي دائرة	START
الإطار المتوازي الأضلاع	READ x,y
الإطار المستطيل	$x = y + 5$
الإطار المعين	$x = y$
الإطار المستطيل المنتهي بنصفي معين	yes                      no $I = 1, n$
الدائرة	10

#### ملاحظة :

يمكن كتابة الخوارزمية أو محتوى المخطط التدفقي باللغة العربية أو باللغة الإنكليزية .

# الفصل الاول

## الخرائط: التدفقيه



#### 4 4 ما هي الأنواع الرئيسية للمخططات التدفقية ؟

توجد أربعة أنواع رئيسية للمخططات التدفقية وهي :

1- مخططات التتابع البسيط ( Simple sequential Flowchart )

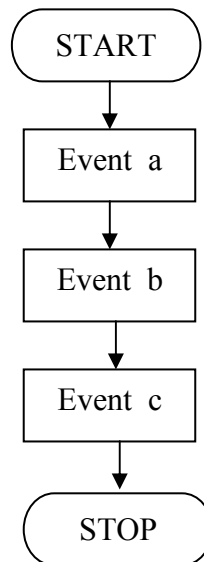
2- مخططات التفرع ( Branched Flowchart )

3- مخططات التكرار البسيط ( Loop Flowchart )

4- مخططات التكرارات المتداخلة ( Nested – loop – Flowchart )

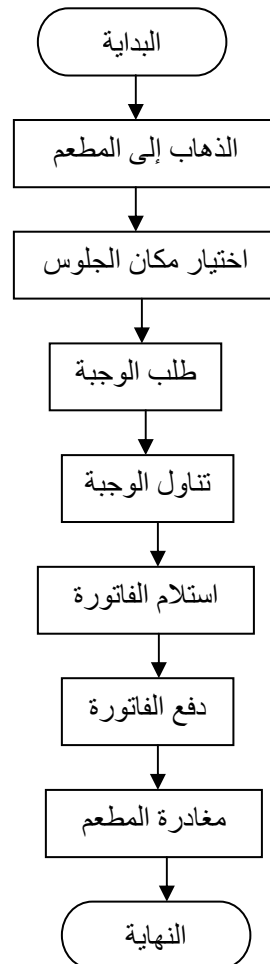
4-4-1 مخططات التتابع البسيط ( Simple sequential Flowchart ) :

يخلو هذا النوع من التفرعات والتكرارات أو القرارات وإنما مجموعة أوامر وأحداث متسلسلة وله الشكل العام :



### مثال /3/ :

المخطط التدفقي لتناول وجبة في مطعم والواردة خوارزميته في المثال /1/



## مثال /4/ :

اكتب الخوارزمية التي تمكننا من حساب محيط ومساحة دائرة نصف قطرها R وارسم المخطط التدفقي لهذه المسألة .

الخوارزمية :

1- ابدأ

2- اجعل قيمة  $PIE = 3.14$

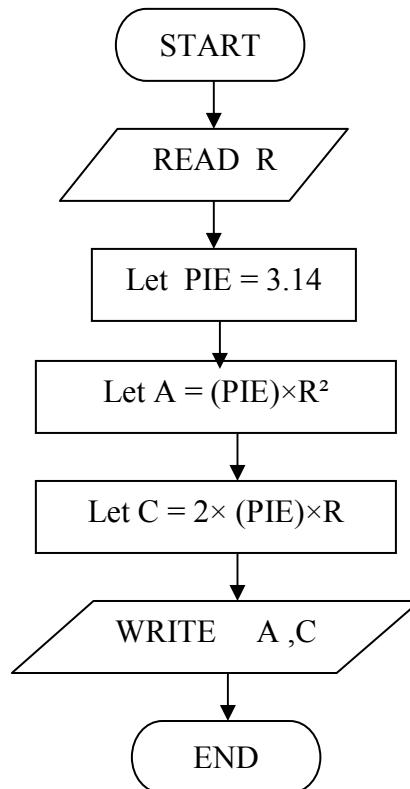
3- احسب المساحة (A) من المعادلة  $A = PIE \times R^2$

4- احسب المحيط (C) من المعادلة  $C = 2 \times PIE \times R$

5- اطبع قيم كل من المساحة والمحيط

6- توقف

المخطط التدفقي :



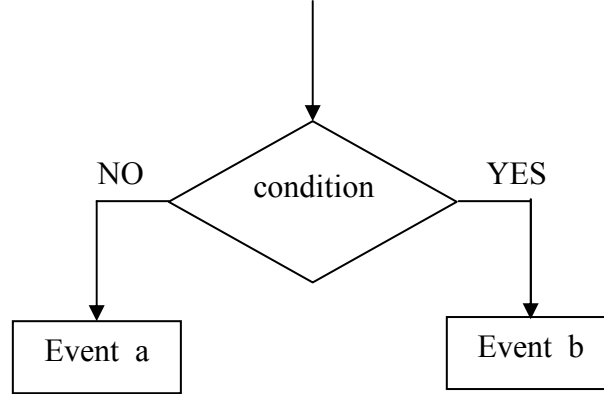
#### 4-4-2- مخططات التفرع (Branched Flowchart) :

يتضمن هذا النوع اتخاذ القرارات أو مفاضلة بين خيارين أو أكثر وهناك أسلوبان في تنفيذ القرار

1- قرار ذو تفرعين .

2- قرار ذو ثلاث تفرعات .

والشكل العام له كما يلي :



مثال /5/ :

سنناقش مثال تناول وجبة في مطعم مع إضافة أننا نريد دفع بقشيش إضافة للفاتورة وفي هذه الحالة ستتم التغيرات التالية :

الخوارزمية :

1- البداية .

2- الذهاب إلى المطعم .

3- اختيار مكان الجلوس .

4- طلب الوجبة.

5- تناول الوجبة .

6- استلام الفاتورة .

7- هل الخدمة جيدة وأعجبك ؟

8- إذا كان الجواب نعم تابع و إلا اذهب إلى الخطوة 10 .

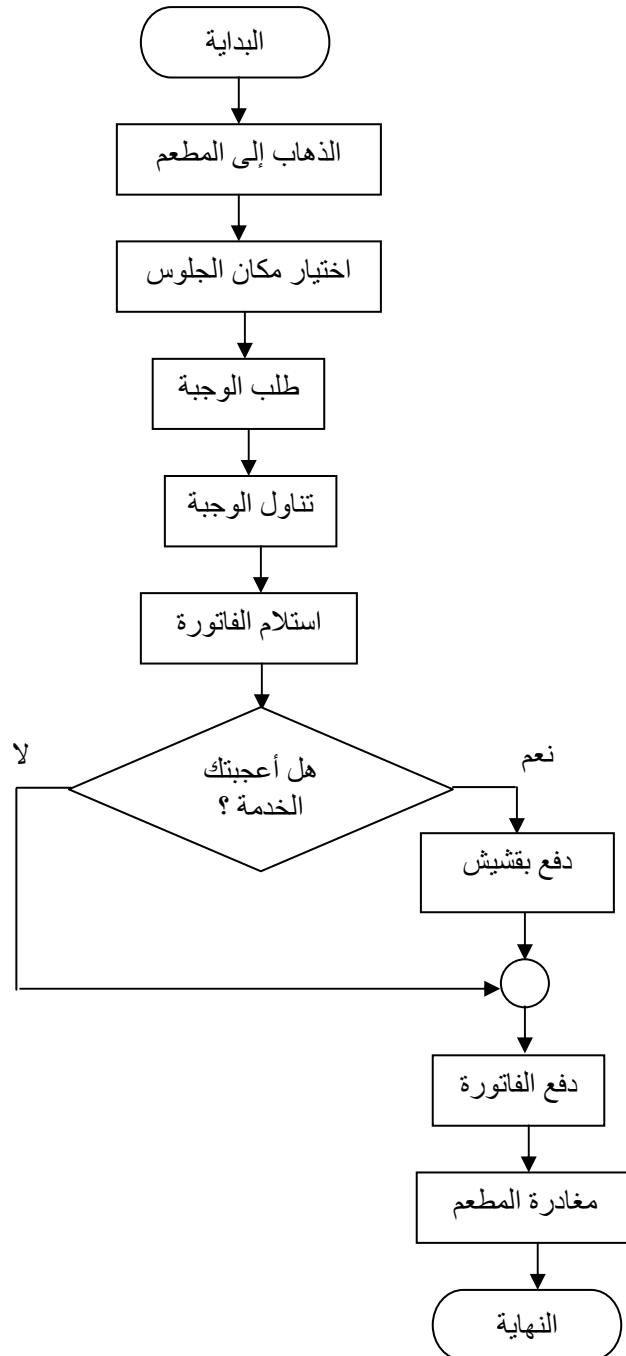
9- دفع بقشيش للعامل .

10- دفع الفاتورة .

11- مغادرة المطعم .

12- النهاية .

## المخطط التدفقي :



مثال /6/ :

اكتب الخوارزمية التي تمكننا من إيجاد القيمة الأعظمية لرقمين وفق المعادلة التالية :

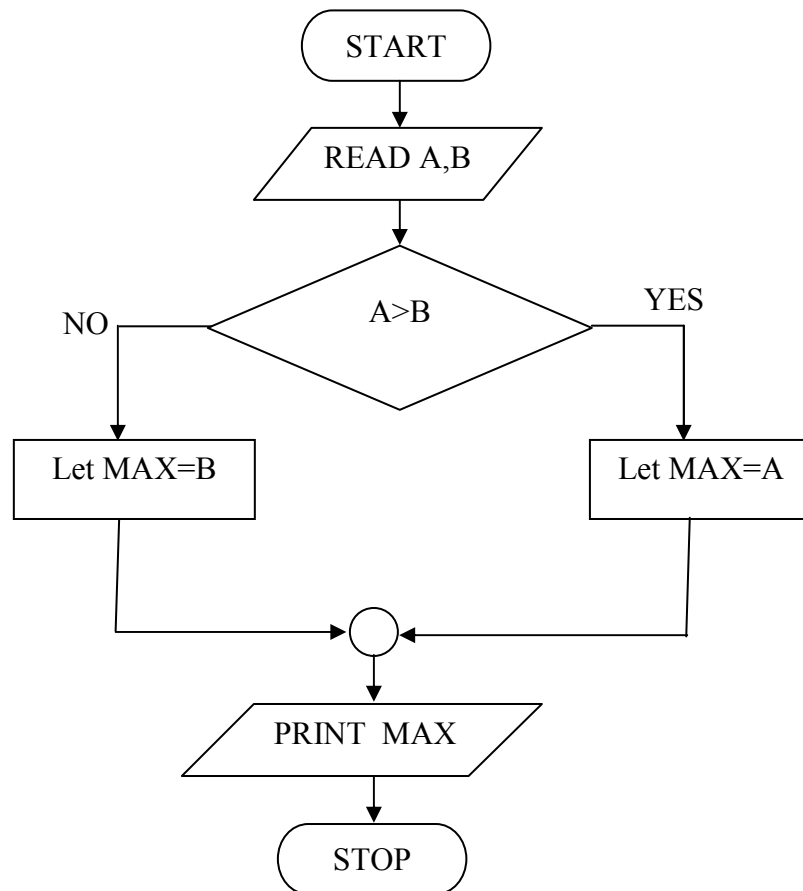
$$\text{MAX} = \max ( A , B )$$

ثم ارسم المخطط التدفقي الموافق .

الخوارزمية :

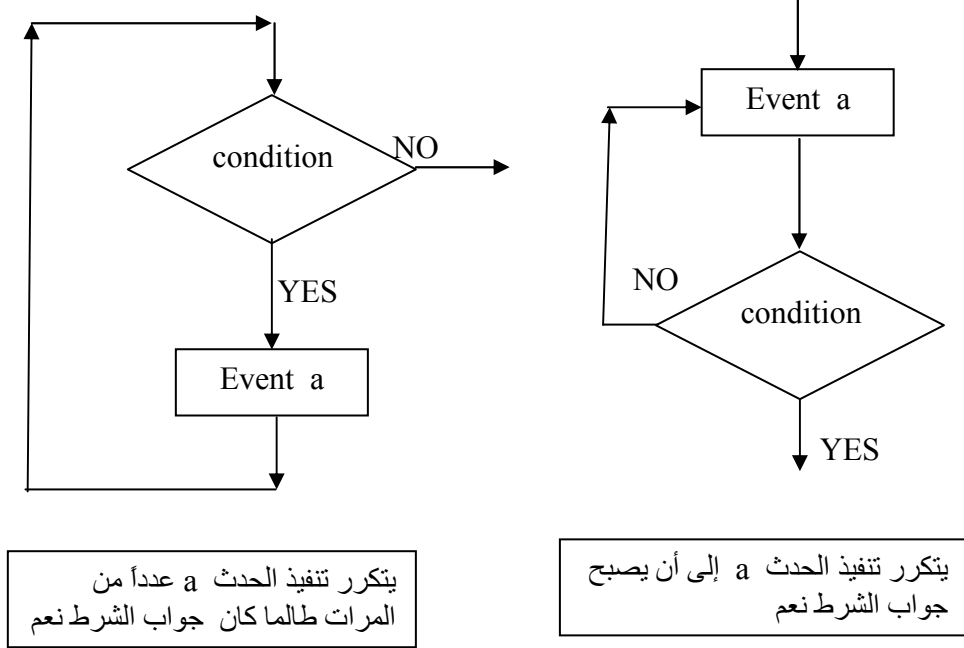
- 1- START
- 2- READ ( A , B )
- 3- IF A>B GOTO 4 ELSE GOTO 5
- 4- LET MAX = A AND GOTO 6
- 5- LET MAX = B
- 6- PRINT MAX
- 7- STOP

المخطط التدفقي :



#### 4-4-3- مخططات التكرار البسيط ( Loop Flowchart ) :

نحتاج لهذا النوع من المخططات لإعادة عملية أو مجموعة من العمليات في البرنامج عدداً محدداً أو غير محدود من المرات والشكل العام لها كما يلي :



#### مثال /7/ :

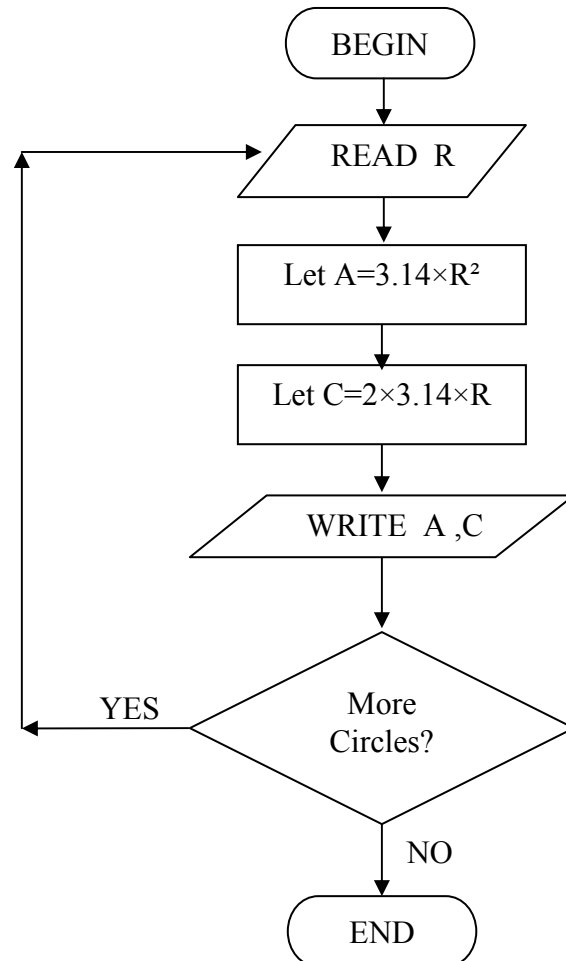
اكتب خوارزمية حساب المساحة والمحيط لمجموعة من الدوائر أنصاف أقطارها معلومة (R)

( تعديل على المثال رقم /4/ )

الخوارزمية :

- 1- Begin
- 2- Read (R)
- 3- Let  $A = 3.14 * R^2$
- 4- Let  $C = 2 * 3.14 * R$
- 5- Write ( A ,C )
- 6- More Circles ? If Yes Goto (2) Else Goto (7)
- 7- End

المخطط التدفقي :





العداد ( Counter ) :

نحتاج في البرامج الحاسوبية إلى العد في كثير من الأحيان وعملية العد للإنسان طبيعية اكتسبها مع نموه خلال حياته إلا أن الحاسب يحتاج لتلقيه خطوات معينة يتبعها ليستطيع العد ويمكن أن نحدد هذه الخطوات بما يلي :

1- اجعل العداد مساوياً للصفر

2- اجعل القيمة الجديدة للعداد تساوي القيمة القديمة له زائد واحد أي :

قيمة العداد ( الجديدة ) = قيمة العداد ( القديمة ) + 1

3- كرر الخطوات ابتداءً من الخطوة (2)

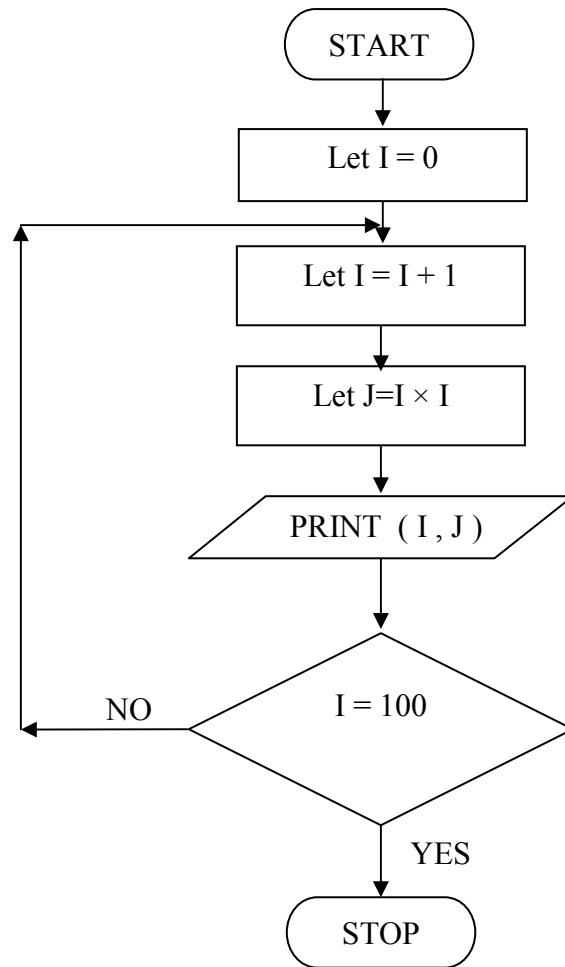
مثال /8/ :

اكتب خوارزمية طباعة الأعداد الطبيعية من 1 إلى 100 ومربعاتها وارسم المخطط التدفقي المناسب .

الخوارزمية :

- 1- START
- 2- Let I = 0
- 3- Let I = I + 1
- 4- Let J = I×I
- 5- PRINT ( I , J )
- 6- If I = 100 Goto (7) Else Goto (3)
- 7- STOP

المخطط التدفقي :



ملاحظة :

تعتمد الزيادة في قيمة العداد على المسألة المطروحة وليس بالضرورة زيادة (1) .

المجاميع الإجمالية :

نحتاج في برامج الحاسب في كثير من الأحيان إلى جمع مجموعة كبيرة من الأعداد التي تمثل ظاهرة معينة ، فمثلاً عندما نريد أن نحسب معدل علامات طالب وكذلك الأمر في هذه الحالة يجب علينا أن نرشد الحاسب للقيام بعملية الجمع ويمكننا ذلك باستخدام متغيرين اثنين أحدهما المتغير الذي نجمعه والآخر هو الجمع الإجمالي ( المجمع ) ويتم ذلك وفق الخطوات التالية :

- 1- اجعل المجمع مساوياً للصفر
- 2- ادخل قيمة واحدة للمتغير
- 3- اجعل القيمة الجديدة للمجمع تساوي القيمة القديمة له زائد القيمة المدخلة للمتغير أي  
أن قيمة المجمع الجديدة = قيمة المجمع القديمة + آخر قيمة مدخلة للمتغير
- 4- كرر ابتداءً من الخطوة الثانية

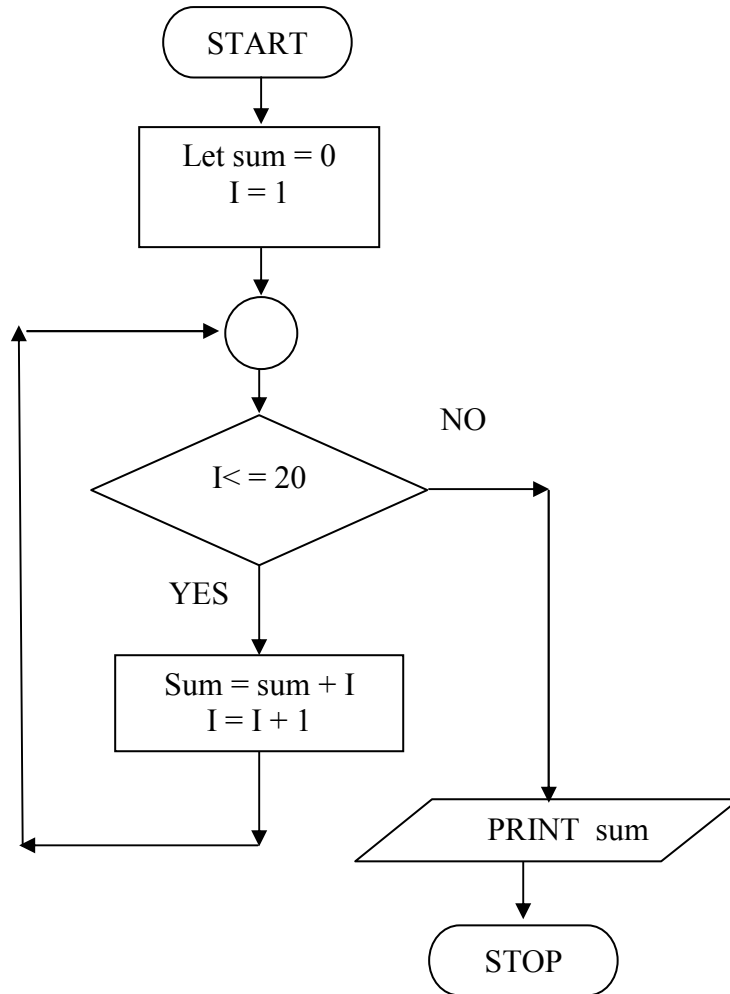
**مثال /9/ :**

اكتب خوارزمية لإيجاد مجموع الأرقام من 1 إلى 20 وارسم المخطط التدفقي المناسب للمسألة المطروحة .

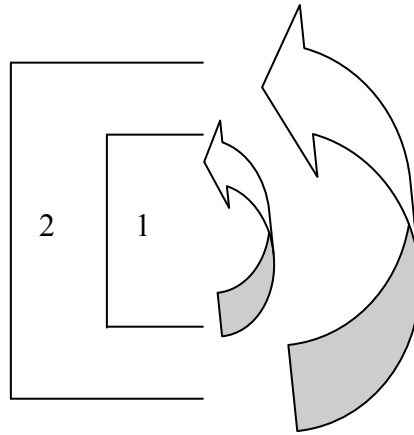
الخوارزمية :

- 1- ابدأ
- 2- ضع  $sum = 0$  و  $I = 0$
- 3- إذا كان جواب الشرط (  $I \leq 20$  ) /نعم/ اذهب إلى الخطوة 4 و إلا اذهب إلى الخطوة 6
- 4- ضع  $sum = sum + I$  و  $I = I + 1$
- 5- اذهب إلى الخطوة 3
- 6- اكتب المجموع sum
- 7- توقف

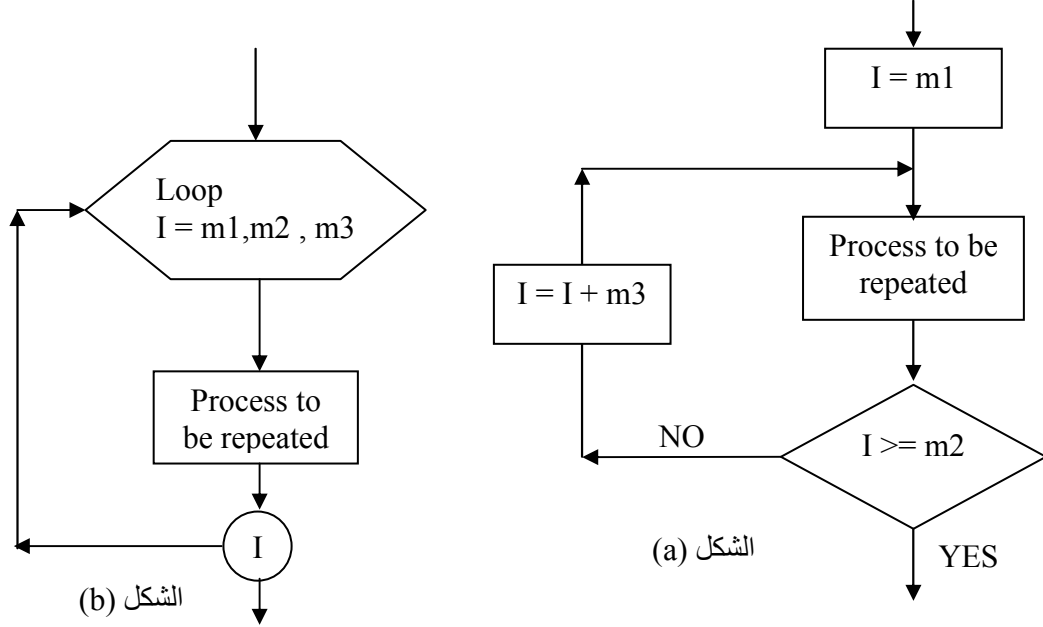
المخطط التدفقي :



4-4-4- مخططات التكرارات المتداخلة ( Nested – loop – Flowchart ) :  
تكون التكرارات متداخلة تماماً بحيث لا تتقاطع فإذا كان لدينا تكرارين من هذا النوع يسمى  
التكرار (1) تكرار داخلي بينما التكرار (2) تكرار خارجي ويتم التنسيق في عمل هذين  
التكرارين بحيث تكون أولوية التنفيذ للتكرار الداخلي كما في الشكل التالي :



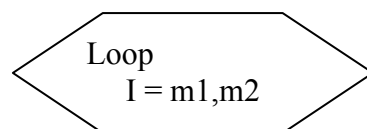
صيغة التكرار باستخدام الشكل الاصطلاحي ( الدوران ) :



نلاحظ في الشكل (a) أنه لتحقيق التكرار نحتاج لما يلي :

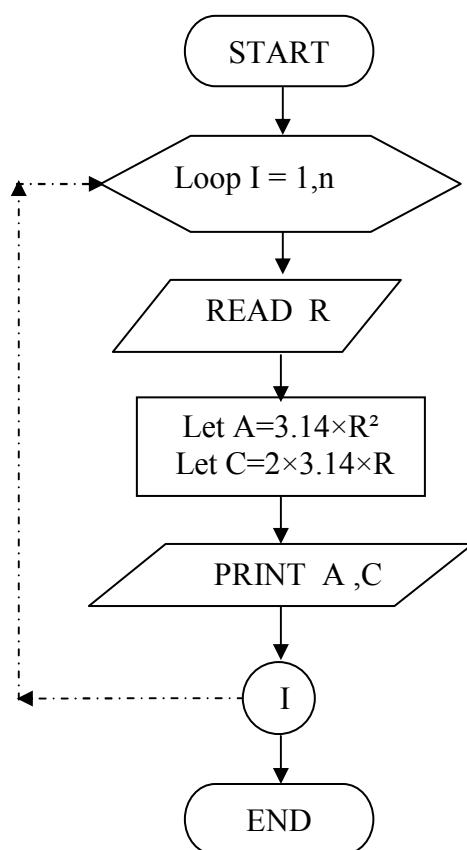
- 1- العدد ( I ) / متغير التكرار .
  - 2- القيمة الأولية للعدد وتساوي m1 .
  - 3- القيمة النهائية للعدد وتساوي m2 .
  - 4- الزيادة الدورية ( الزيادة عند نهاية كل تكرار ) وتساوي m3 .
- وتكون آلية عمل هذه العناصر كما يحددها المبرمج بما يلي :
- 1- اجعل العدد I يبدأ بقيمة أولية مقدارها m1 .
  - 2- أتم الإجراءات المطلوب إعادتها .
  - 3- إذا كانت قيمة العدد I وصلت إلى القيمة النهائية m2 اذهب إلى الخطوة التالية في البرنامج وإلا فإذهب إلى الخطوة (4)
  - 4- زد العدد I بمقدار الزيادة الدورية m3
  - 5- عد إلى الخطوة (2)
- يمكننا استبدال الخطوات (1-3-4-5) في الشكل (a) بخطوة واحدة مبينة في الشكل (b) حيث ينفذها الحاسب بشكل آلي مما يؤدي إلى تسهيل عملية البرمجة واختصار عدد التعليمات وتجنب الأخطاء .

نشير إلى أن قيمة m3 تساوي /1/ دائماً ما لم تعط قيمة أخرى غير ذلك ،وفي حال عدم ذكر m3 تكون قيمتها مساوية /1/ ضمناً وتمثل كما يلي :



**مثال /10/ :**

اعد رسم المخطط التدفقي لإيجاد مساحة ومحيط n من الدوائر الوارد في المثال / 7/ باستخدام الدوران .



## 5- ما هي فعالية الخوارزمية (درجة تعقيد الخوارزمية)؟

الجدير بالذكر أنه قد يكون هناك أكثر من خوارزمية لحل مسألة واحدة وجميع الخوارزميات تؤدي إلى نفس النتيجة ولكن بطرق مختلفة وبكفاءات متفاوتة وهنا كانت الحاجة لمعرفة فعالية الخوارزمية لاختيار الأفضل منها وتعتمد فعالية الخوارزمية على عاملين أساسيين وهما :

- 1- حجم الذاكرة اللازم لتخزين هذه المعطيات وإعطاء إمكانية استخدامها .
  - 2- الوقت اللازم لإدخال المعطيات إلى الذاكرة وكذلك الوقت المطلوب لتنفيذ الخوارزمية ويهمل العامل الأول بالقياس لأهمية العامل الثاني .
- يتعلق تقييم الزمن بعدة عوامل منها حجم الدخل ويتطلب زمن من أجل التعبير عن معطيات الدخل وبالتالي فإن زمن التنفيذ تابع لـ  $n$  أي  $T(n)$  ويتعلق كذلك بنوع التعليمات والسرعة التي تنفذ فيها الآلة وهذه العوامل تعتمد على نوع الجهاز المستخدم ، ومنه فإننا لا نستطيع تحديد  $T(n)$  بشكل دقيق في واحدة الزمن الحقيقي كالثواني فهو بالتالي عدد تقريبي لعدد العمليات المنفذة وبسبب وجود عامل آخر يؤثر على الوقت وهو نوعية الشيفرة المنتجة في الحاسب نفسه ، كذلك الأمر لا يمكن تحديد  $T(n)$  لعدد العمليات المنفذة وإنما يحدد  $(n)$  عدد المرات التي يتم تنفيذ الخوارزمية فيه .

### مثال /11/ :

لنكتب خوارزمية حساب متوسط مجموعة قيم ونستنتج درجة تعقيدها .

رقم الخطوة	الخوارزمية	عدد مرات التنفيذ
1	اقرأ عدد الأرقام ( n )	1
2	اجعل المجموع يساوي الصفر ( sum = 0 )	1
3	اجعل العداد يساوي الصفر ( I=0 )	1
4	طالما $I \leq n$ نفذ	$n+1$
5	اقرأ الرقم	$n$
6	أضف الرقم إلى المجموع	$n$
7	قم بزيادة I بمقدار 1	$n$
8	احسب المتوسط = المجموع ÷ عدد الأرقام ( sum/n )	1
المجموع		$4n+5$



ومنه ينتج أن الزمن اللازم لهذه الخوارزمية يعطى بالعلاقة :

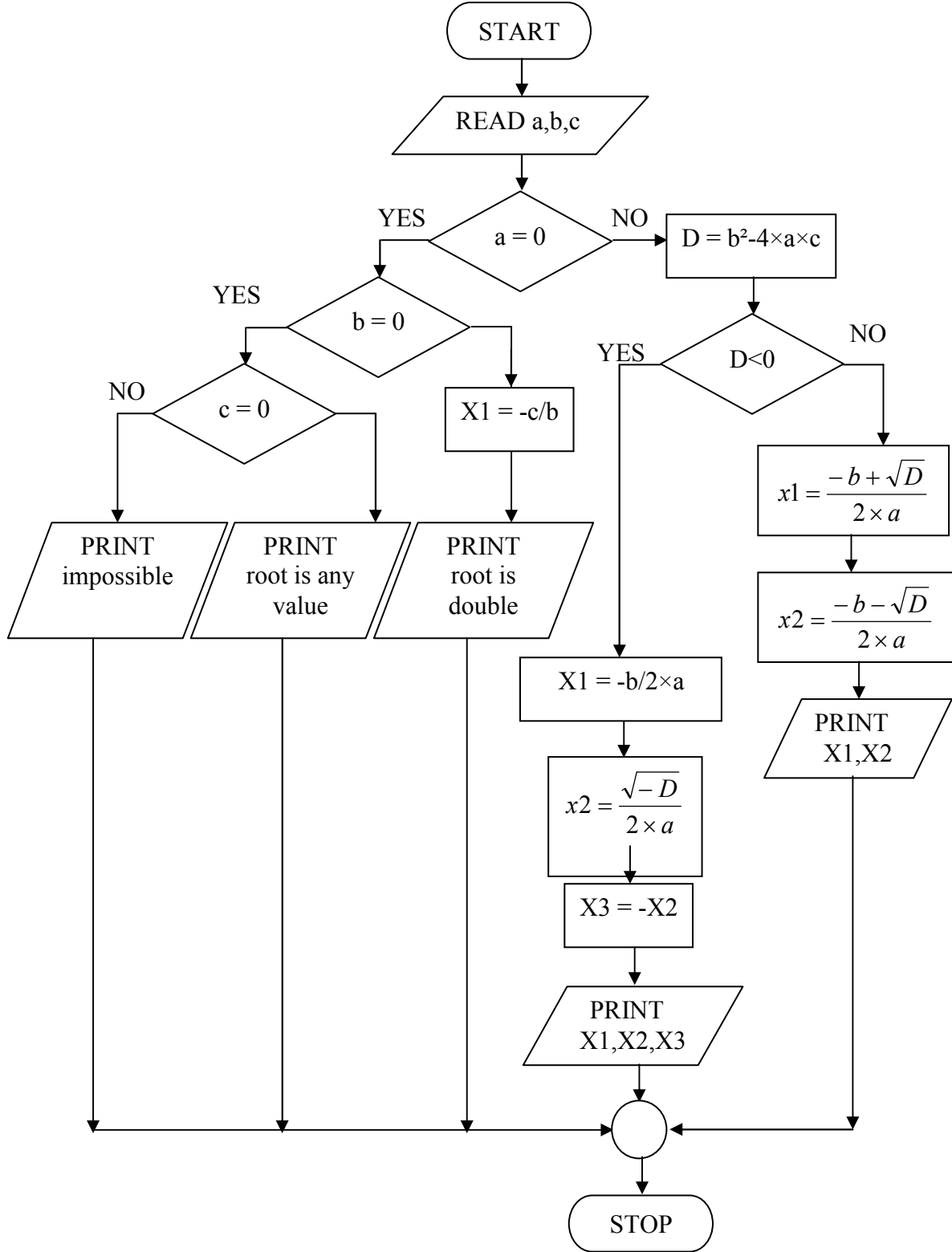
$$T(n) = 4n + 5$$

حيث  $n$  عدد قيم الدخل

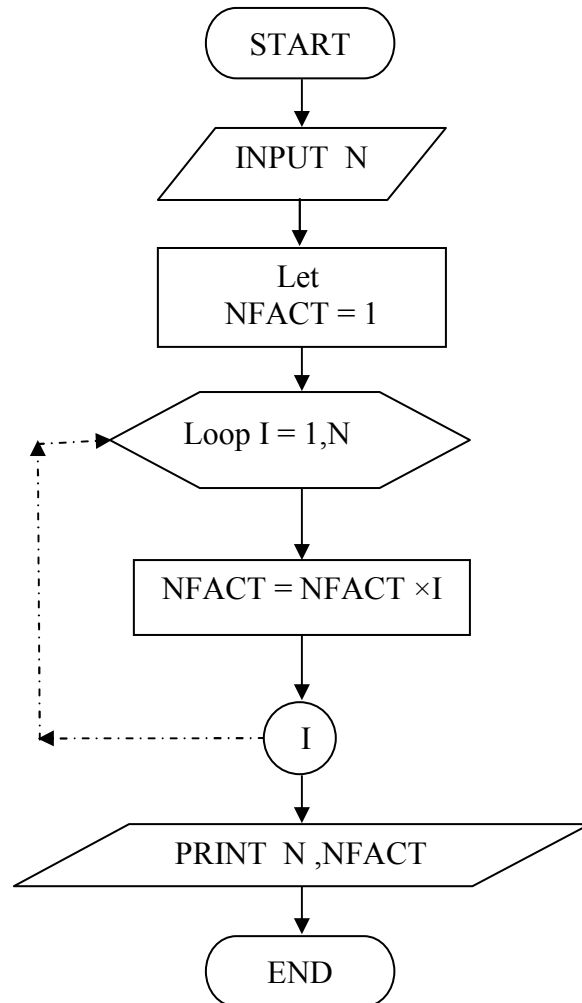
نلاحظ أن العلاقة تابع درجة أولى بالنسبة لـ  $n$  إلا أنه قد تكون الخوارزمية أكثر تعقيداً وينتج تابع يحتتمل أي شكل للمعادلة كالدرجة الثانية أو الثالثة أو حتى تابع لوغاريتمي أو أسّي .

## 6- تطبيقات على الخوارزميات والمخططات التدفقية :

1- ارسم المخطط التدفقي لحل معادلة من الدرجة الثانية من الشكل :  $ax^2 + bx + c = 0$



2- ارسم المخطط التدفقي لإيجاد العايلي ( n عاملي ) والتي تعطى بالعلاقة :  
$$N! = n(n-1)(n-2)(n-3) \dots 2 \times 1$$



3- قامت إحدى الشركات التجارية في مدينة اللاذقية بعرض كمية من السجاد وتقدم عدد من المناقصين للشراء حسب المناقصة المعلنة وبفرض الأسعار بالليرة السورية هي التالية :

$X_1, X_2, X_3, \dots$

وإذا اعتبرنا  $n$  عدد المتقدمين فاكتمل الخوارزمية التي يتبعها الحاسب لاختيار أكبر مبلغ مقدم لشراء الصنف وارسم المخطط التدفقي للحل .

الحل :

نلاحظ أنه لدينا مجموعة من عروض الأسعار لنعتبرها مرتبة في نسق  $X$  على الشكل التالي :

$X(1), X(2), X(3), \dots, X(n)$

ونفترض أن  $X_{MAX}$  يمثل أعلى عروض الأسعار فيكون :

الخوارزمية :

1- ابدأ

2- ادخل العدد الكلي للمتقدمين ( $n$ )

3- ادخل عروض الأسعار  $X(1), X(2), X(3), \dots, X(n)$

4- اجعل  $X_{MAX} = X(1)$

5- اجعل العداد  $I = 2$

6- إذا كان  $X_{MAX} \geq X(I)$  اذهب إلى الخطوة 8 وإلا اذهب إلى الخطوة 7 (مقارنة)

7- اجعل  $X_{MAX} = X(I)$

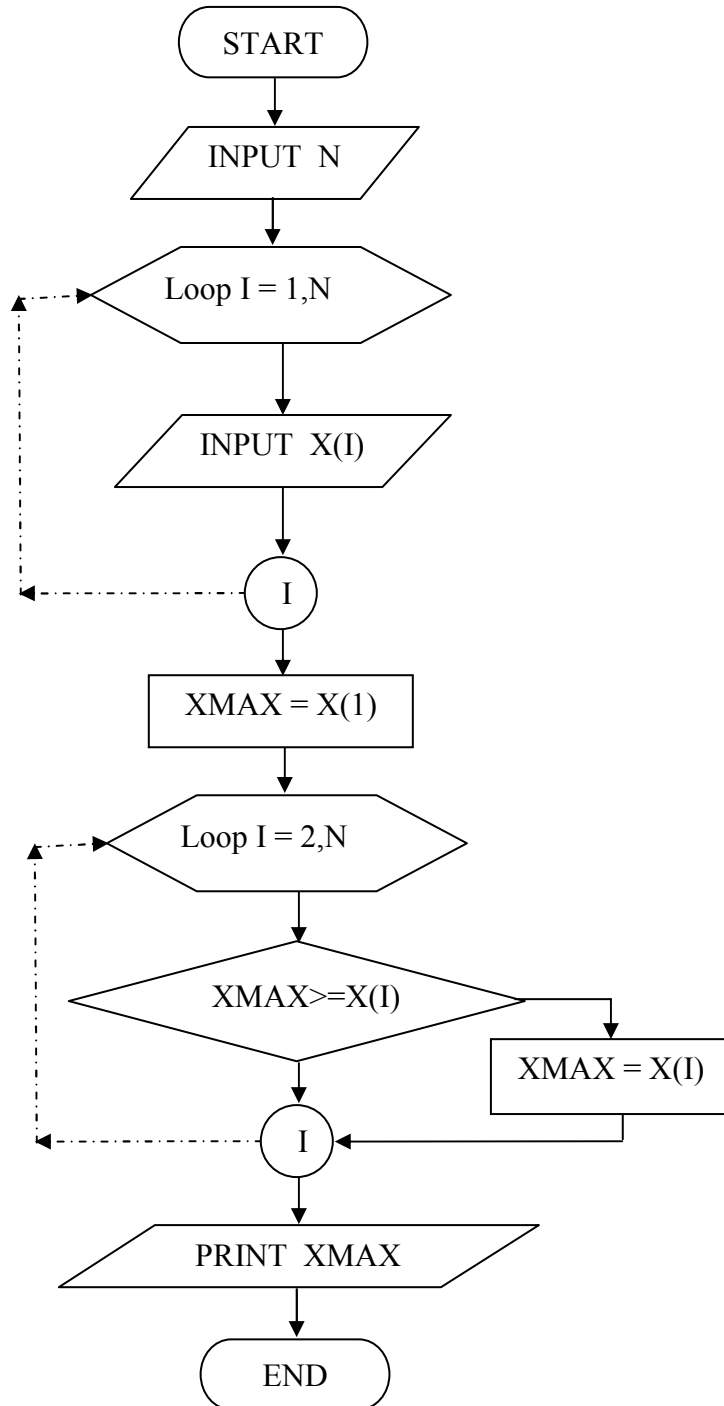
8- اجعل  $I = I + 1$

9- إذا كان  $I > n$  اذهب إلى الخطوة 10 وإلا عد إلى الخطوة 6

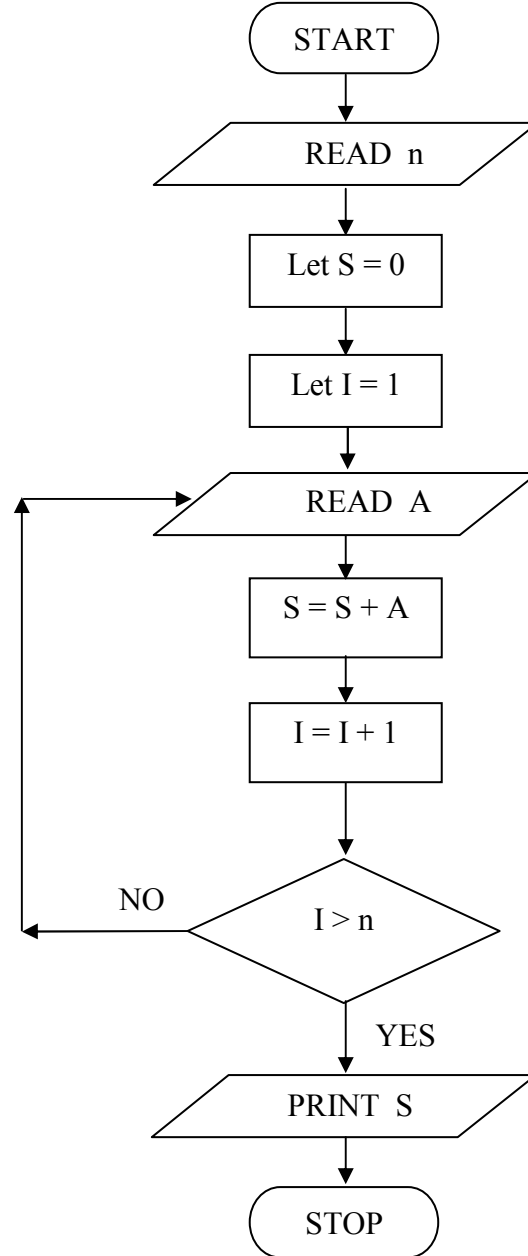
10- اطبع  $X_{MAX}$

11- النهاية

المخطط التدفقي :



4 ارسم المخطط التدفقي لحساب مجموع  $n$  عدد .



**ملاحظة :**

قمنا بكتابة الخوارزميات والمخططات التدفقية باللغة العربية تارة والإنكليزية تارة أخرى واستعمال كلمات متنوعة كلها مستخدمة في هذا المجال ليصار إلى التآلف مع هذه المفردات .

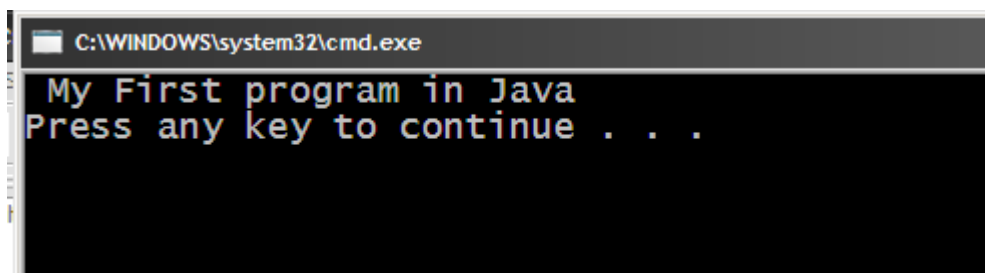
\*\*\*\*\*

## مميزات لغة الجافا:

- إنها لغة قوية تحتوي على أدوات كثيرة تساعد في كتابة البرامج.
  - لكون جافا لغة حديثة مكنها من تلافي عيوب كثير من اللغات قبلها، من أهم هذه العيوب إمكانية الوصول المباشر لمواقع الذاكرة الخاصة بالبرنامج والذي يؤدي إلى ضعف سرية المعلومات وسهولة تدميرها.
  - إن البرنامج المكتوب بلغة جافا يمكن نقله وتشغيله على جهاز حاسوب آخر يحتوي على نظام تشغيل يختلف عن الحاسوب الأول (مثلاً يحتوي Windows, Linux وغيرهما) بدون مشاكل.
  - تعتبر لغة جافا لغة برمجة بالكائنات (Object Oriented Programming Language) ، ويعتبر هذا الصنف من لغات البرمجة من أوسعها انتشاراً وأكثرها استخداماً اليوم.
- لغة جافا كغيرها من لغات البرمجة لاتخلو من العيوب، ويكمن اعتبار لغة جافا بطيئة نسبياً. إن السرعة ميزة مهمة، ولكن يجب التضحية ببعض المميزات لاكتساب مميزات أهم.

وهذا اول برنامج لنتعرف على محتويات برنامج جافا

```
1 class first
2 {
3     public static void main(String args[])
4     {
5         System.out.println(" My First program in Java ");
6
7         } // end of main
8     } // end of class
```



يبدأ برنامج جافا بالكلمة المحجوزة class يليها اسم البرنامج الذي اختاره المبرمج وهنا first ويجب حفظ الملف بنفس الاسم ويحتوي ال class على الدالة (public static void main(String args[])) ويبدأ تنفيذ البرنامج من هذه الجملة

### انواع البيانات في الجافا:

#### الاعداد الصحيحة:

النوع	طوله في الذاكرة
Byte	1 byte
short	2 byte
int	4 byte
long	8 byte

#### الاعداد الحقيقية:

النوع	طوله في الذاكرة
Float	4 byte
Double	8 byte

#### النوع المنطقي:

Boolean ويشمل القيم true او false

#### النوع String:

هذا النوع شائع الاستخدام على الرغم من انه من انواع البيانات غير الاساسية ويستخدم لتعريف النصوص

#### المتغيرات:

#### تعريف المتغيرات:

```
int number1;  
int number1 , number2;  
double num;  
boolean test;  
char ch;  
String text;
```



وضع قيمة للمتغير:

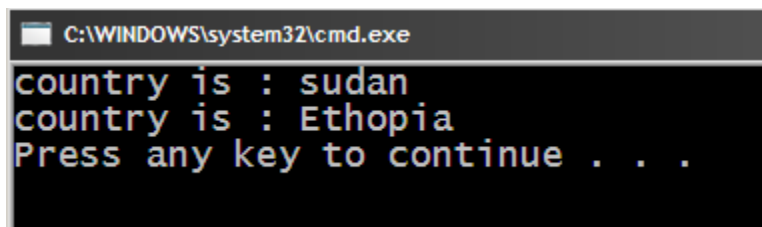
```
number1 = 6;  
num = 6.8;  
text = "Sudan";  
test = true;  
ch = 'a';
```

وهذا يعني ان قيمة المتغير number1 هي 6 و قيمة المتغير test هي true وهكذا لبقية المتغيرات

مثال:

```
1  class country  
2  {  
3      public static void main(String args[])  
4      {  
5          String count;  
6          count = "sudan";  
7          System.out.println("country is : " + count);  
8          count = "Ethopia";  
9          System.out.println("country is : " + count);  
10         }  
11     }
```

الخرج من البرنامج



```
C:\WINDOWS\system32\cmd.exe  
country is : sudan  
country is : Ethopia  
Press any key to continue . . .
```

ربط السلاسل نصية:

لربط السلاسل النصية نستخدم المعامل (+)

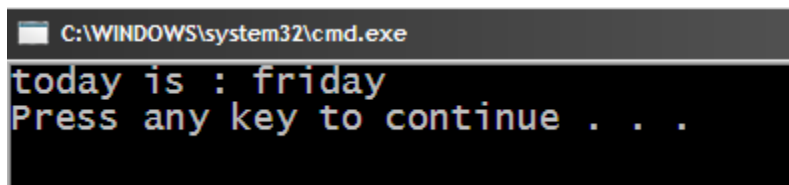
والمثال التالي يوضح ذلك

```

1  class today
2  {
3      public static void main(String args[])
4      {
5          -----
6              String text = "today is : ";
7              String day = "friday";
8              String output = text + day;
9              System.out.println(output);
10
11      }
12  }

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
today is : friday
Press any key to continue . . .

```

الدالة `print` و `println` تقوم هذه الدوال بعملية الطباعة على الشاشة ولكن الدالة `println` بعد الفراغ من الطباعة تنتقل الى سطر جديد

### تعريف الثوابت:

الثابت هو متغير لا يمكن تغيير قيمته في البرنامج ولكننا نقوم بتعريفه ووضع قيمة ابتدائية له لحظة التعريف، وتظل هذه القيمة ثابتة طوال البرنامج. تعريف الثوابت لا يختلف عن المتغيرات إلا في الكلمة المحجوزة **final** والتي نكتبها أمام التعريف لنستدل بها على أنه ثابت.

```

final char plus = '+';
final double pi = 3.14;

```

## العمليات الرياضية في الجافا:

العملية	العلامة	طريق الكتابة
الجمع Addition	+	$a + b$
الطرح Subtraction	-	$a - b$
الضرب Multiplication	*	$a * b$
القسمة Division	/	$a / b$
باقي القسمة Modulus	%	$a \% b$

## العبارات المنطقية:

العملية	معناها	قيمتها
&&	x and y	صواب فقط إذا كان كل من x و y صواب
	x or y	خطأ فقط إذا كان كل من x و y خطأ
!	not z	خطأ إذا كان z صواب، وصواب إذا كان z خطأ

مثال :

```

1  class math
2  {
3      public static void main(String args[])
4      {
5          int num1 = 3, num2 = 4;
6          int sum = num1 + num2;
7          System.out.println("Sum is : " + sum);
8      }
9  }
10 }
```

الخرج من البرنامج

```
C:\WINDOWS\system32\cmd.exe
Sum is : 7
Press any key to continue . . .
```

قراءة البيانات من المستخدم:

لقراءة البيانات من المستخدم نستخدم الكائن `BufferedReader` الموجود بالحزمة `java.io` والبرنامج التالي يوضح ذلك

```
1 import java.io.*;
2 // to use BufferedReader
3 class input
4 {
5     public static void main(String args[]) throws IOException
6     {
7         String text;
8         char ch;
9         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
10        System.out.print(" Enter Text : ");
11        text = in.readLine();
12        System.out.print(" Enter Character : ");
13        ch = (char)in.read();
14        System.out.println(text + " - " + ch);
15    }
16 }
```

الخرج من البرنامج

```
C:\WINDOWS\system32\cmd.exe
Enter Text :

C:\WINDOWS\system32\cmd.exe
Enter Text : Mohammed
```

```
C:\WINDOWS\system32\cmd.exe
Enter Text : Mohammed
Enter Character : M

C:\WINDOWS\system32\cmd.exe
Enter Text : Mohammed
Enter Character : M
Mohammed - M
Press any key to continue . . .
```

طريقة اخرى لقراءة البيانات من المستخدم:

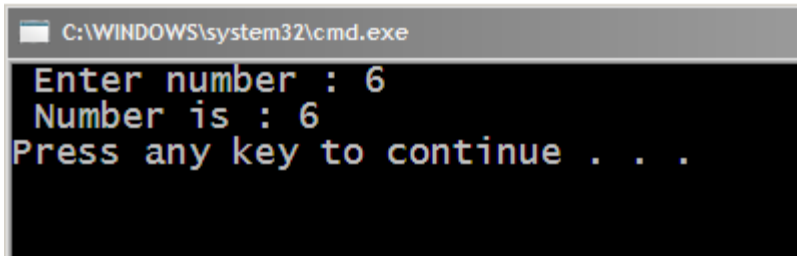
نستخدم الكائن in الموجود في الحزمة System ولعمل ذلك نستخدم الفئة Scanner الموجودة في

import java.util.scanner

والمثال التالي يوضح ذلك

```
1  import java.util.Scanner;
2  // to import scanner class
3  class input
4  {
5      public static void main(String args[])
6      {
7          Scanner in = new Scanner(System.in);
8          System.out.print(" Enter number : ");
9          int m = in.nextInt();
10         System.out.println(" Number is : " + m);
11     }
12 }
```

الخرج من البرنامج

A screenshot of a Windows command prompt window. The title bar at the top shows the path 'C:\WINDOWS\system32\cmd.exe'. The command prompt itself has a black background with white text. It displays three lines of text: 'Enter number : 6', 'Number is : 6', and 'Press any key to continue . . .'.

```
C:\WINDOWS\system32\cmd.exe
Enter number : 6
Number is : 6
Press any key to continue . . .
```

ونلاحظ ان هذه الطريقة اسهل في الادخال

لادخال قيمة من النوع char نستخدم `in.nextChar()` و `in.nextString()` للسلاسل

## جمل التحكم والحلقات التكرارية

---

### عبارات المقارنة:

>	أكبر من
>=	أكبر من أو يساوي
<	أصغر من
<=	أصغر من أو يساوي
==	يساوي
!=	لا يساوي

### جمل الشرط:

#### عبارة الشرط **if**:

الصيغة العامة لعبارة if

```
if(condition)
{
    statement ;
}
```

والمثال التالي يوضح ذلك

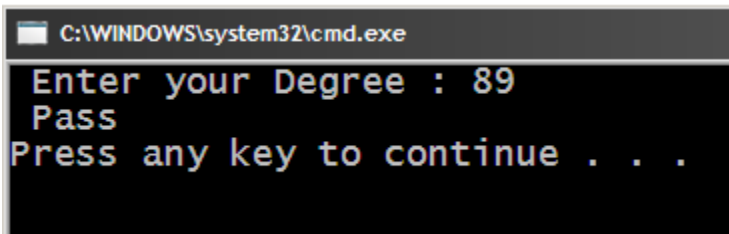
```

1 // if statement program
2 import java.util.*;
3
4 class IfStatement
5 {
6     public static void main(String args[])
7     {
8         int degree;
9         Scanner input = new Scanner(System.in);
10        System.out.print(" Enter your Degree : ");
11        degree = input.nextInt();
12
13        if(degree >= 50)
14        {
15            System.out.println(" Pass ");
16        }
17    }
18 }
19 }

```

الخرج من البرنامج

- القيمة المدخلة اكبر من 50

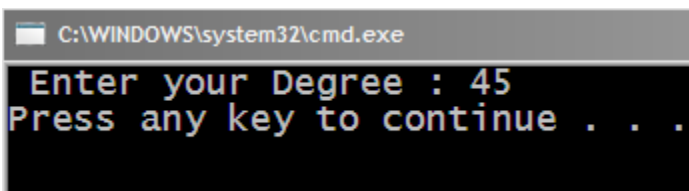


```

C:\WINDOWS\system32\cmd.exe
Enter your Degree : 89
Pass
Press any key to continue . . .

```

- القيمة المدخلة اقل من 50



```

C:\WINDOWS\system32\cmd.exe
Enter your Degree : 45
Press any key to continue . . .

```

العبارة if else :



```
if(condition)
    statement1 ;
else
    statement2 ;
```

عندما تكون قيمة الشرط condition صواباً، يتم تنفيذ statement1 وتجاهل else والعبارة التي تليها. وعندما يكون الشرط condition خطأ يتم تجاهل العبارة statement1 وتنفيذ العبارة statement2 . وكما في عبارة if ، إذا كان المطلوب تنفيذ أكثر من أمر واحد في حالة قيمة الشرط خطأ، توضع الأوامر بين قوسين بداية ونهاية.

الآن سنقوم بتعديل المثال السابق

```
1 // if statement program
2 import java.util.*;
3
4 class IfStatement
5 {
6     public static void main(String args[])
7     {
8         int degree;
9         Scanner input = new Scanner(System.in);
10        System.out.print(" Enter your Degree : ");
11        degree = input.nextInt();
12
13        if(degree >= 50)
14            System.out.println(" Pass ");
15        else
16            System.out.println(" Fail ");
17    }
18 }
```

الخرج من البرنامج عندما يدخل المستخدم قيمة اكبر من 50

```
C:\WINDOWS\system32\cmd.exe
Enter your Degree : 78
Pass
Press any key to continue . . .
```

وعندما يدخل قيمة اقل من 50

```
C:\WINDOWS\system32\cmd.exe
Enter your Degree : 47
Fail
Press any key to continue . . .
```

مثال:

برنامج يحدد اذا كان العدد يقبل القسمة على 6

```
1 // program to known number is devisable by six
2 import java.util.*;
3 class test1
4 {
5     public static void main(String args[])
6     {
7         Scanner in = new Scanner(System.in);
8         int number1;
9         System.out.print(" Enter number : ");
10        number1 = in.nextInt();
11        if(((number1%2)== 0)&&((number1%3)==0))
12            System.out.println(" number is devisable by 6 ");
13        else
14            System.out.println(" number is not devisable by 6 ");
15    }
16 }
```

الخرج من البرنامج عندما ادخل المستخدم الرقم 24

```
C:\WINDOWS\system32\cmd.exe
Enter number : 24
number is devisable by 6
Press any key to continue . . .
```

عندما ادخل المستخدم الرقم 56

```
C:\WINDOWS\system32\cmd.exe
Enter number : 56
number is not devisable by 6
Press any key to continue . . .
```

ويمكن التعبير عن ال if else بـ :

**max = (number1 < numner2)?number1:number2;**

المثال التالي يوضح ذلك

```
1  import java.util.*;
2  class test2
3  {
4      public static void main(String args[])
5      {
6          Scanner in = new Scanner(System.in);
7          int number1 , number2;
8          System.out.print(" Enter number 1 : ");
9          number1 = in.nextInt();
10         System.out.print(" Enter number 2 : ");
11         number2 = in.nextInt();
12         int max = (number1 > number2) ? number1 : number2;
13         System.out.println(" Maximum number is " + max );
14     }
15 }
```

الخروج من البرنامج

```
C:\WINDOWS\system32\cmd.exe
Enter number 1 : 78
Enter number 2 : 90
Maximum number is 90
Press any key to continue . . .
```

العبارة : if else if else..... :

```

if(condition)
    statement;
else if(condition)
    statment;
else
    statement;

```

العبارة **switch**:

الصيغة العامة للعبارة switch

```

switch(variable)
{
    case value1:
        statement;
    break;
    case value2:
        statement;
    break;
    default:
        statement;
}

```

حيث variable هو اسم المتغير المطلوب إجراء الاختبارات على قيمته، ويشترط فيه أن يكون من النوع **int** أو **char** . value2, value1 عبارة عن قيم يمكن أن يأخذها المتغير.

عند إجراء الاختبار على المتغير variable ، إذا ساوت قيمته أيّاً من القيم الموجودة بعد كلمة **case** ، يتم تنفيذ العبارة أو العبارات التالية حتى الوصول إلى نهاية **switch** أو العثور على الكلمة **break** ، والتي تقوم بإيقاف تنفيذ عبارات **case** التالية لعبارة **case** التي تم تنفيذها. أما إذا احتوى المتغير على قيمة غير موجودة ضمن عبارات **case** ، عندئذ يتم تنفيذ العبارات التالية للكلمة المحجوزة **default** .

مثال:

```

1  import java.util.*;
2  class switchstatement
3  {
4      public static void main(String args[])
5      {
6          Scanner in = new Scanner(System.in);
7          int digit;
8          String number;
9          System.out.print(" Enter number  : ");
10         digit = in.nextInt();
11         switch(digit)
12         {
13             case 1:
14                 number = "one";
15                 break;
16             case 2:
17                 number = "two";
18                 break;
19             case 3:
20                 number = "three";
21             break;
22             default:
23                 number = " number is " + digit;
24         }
25         System.out.println(number);
26     }
27 }

```

الخرج من البرنامج

```

C:\WINDOWS\system32\cmd.exe
Enter number  : 1
one
Press any key to continue . . .

```

```

C:\WINDOWS\system32\cmd.exe
Enter number  : 4
number is 4
Press any key to continue . . .

```

## الحلقات التكرارية:

في كثير من البرامج، نحتاج لتكرار تنفيذ جزئية معينة من البرنامج لعدد من المرات، مثلاً إذا كان البرنامج يقوم بقراءة أسماء 50 موظفاً، ليس من المنطقي أن نكتب 50 عبارة قراءة مختلفة. أو إذا كان البرنامج يطبع الأعداد من 1 إلى 1000، فلا يمكن تصور برنامج يحتوي على 1000 عبارة طباعة، لأنه سيكون طويلاً جداً، وفي نفس الوقت يحتوي على مجموعة من العمليات المتشابهة، وهي عملية الطباعة.

من المتوقع أن نحتاج إلى تكرار تنفيذ العبارات في أغلب البرامج، وخاصة البرامج الكبيرة والأنظمة لأنها تتعامل مع مجموعات من البيانات. ففي نظام للمرتبات، يتم حساب المرتب لكل موظف على حده. أي تكرار عملية حساب المرتب بعدد الموظفين. وفي نظام بنكي، للبحث عن اسم عميل بواسطة رقم حسابه، يتم المرور على جميع عملاء البنك واختبار أرقام الحساب إلى أن نجده أو ينتهي العملاء. ولذلك نجد أن للتكرار أهمية كبرى يكاد لا يستغني عنها أي نظام.

توفر لغة **Java** ثلاث عبارات مختلفة للتكرار. سنتناولها بالتفصيل.

### الحلقة **while**:

تقوم الحلقة **while** بتكرار العبارات بداخلها مادامت قيمة الشرط **condition** هي **true**

الصيغة العامة للعبارة **while**

```
while(condition)
{
    Statement;
}
```

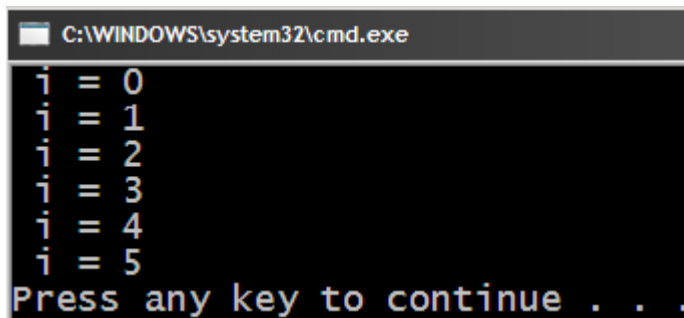
مثال:

```

1  // use while statement
2  class whileloop
3  {
4      public static void main(String args[])
5      {
6          int i = 0;
7          while(i <= 5)
8          {
9              System.out.println(" i = " + i);
10             i++;
11         }
12     }
13 }

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
Press any key to continue . . .

```

الحلقة **do while**:

هي شبيهة بحلقة **while** إلا أنه يتم اختبار شرطها في نهاية الحلقة. أي أنها تقوم بتنفيذ العبارات الموجودة بداخلها ثم اختبار قيمة الشرط لتحديد استمرارية تكرار عباراتها أو توقفها.

الصيغة العامة للحلقة **do while**

```

do
{
    statement;
}while(condition);

```

مثال:

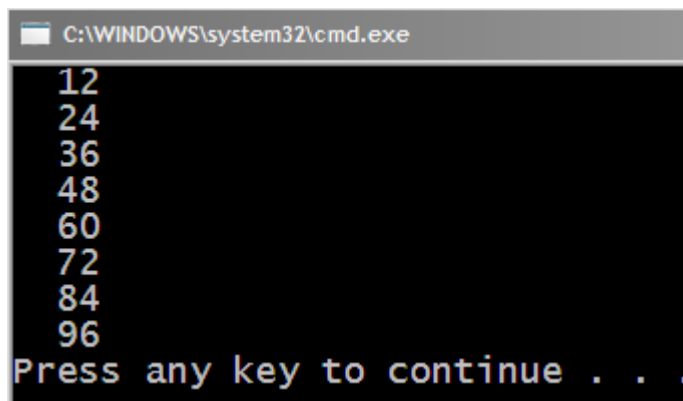
برنامج يقوم بطباعة مضاعفات العدد 12 ال 100

```

1 // use do while statement
2 class dowhileloop
3 {
4     public static void main(String args[])
5     {
6         int i = 12;
7         do
8         {
9             System.out.println(" " + i);
10            i += 12;
11        }while(i <= 100);
12    }
13 }

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
12
24
36
48
60
72
84
96
Press any key to continue . . .

```

### الحلقة for:

عبارة أو حلقة **for** تقوم بتكرار تنفيذ التعليمة statement لعدد معلوم من المرات. هذا العدد المعلوم عبارة عن عدد القيم التي يأخذها عداد الحلقة counter. يأخذ العداد القيمة الابتدائية initialValue ويتم تنفيذ العبارة statement ، وبعد كل تنفيذ تزداد قيمة المتغير counter حسب ما هو معرف في incrementExpression حتى يصل إلى القيمة النهائية finalValue ، وعندها يتوقف التكرار.

الصيغ العامة للحلقة for

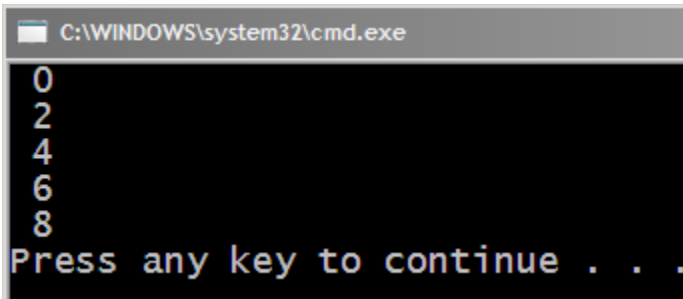


```
for(start;end;frequency)
    statement ;
```

مثال:

```
1 // use for statement
2 class forloop
3 {
4     public static void main(String args[])
5     {
6         for(int i = 0;i < 10; i+=2)
7             System.out.println(" " + i);
8     }
9 }
```

الخروج من البرنامج



```
C:\WINDOWS\system32\cmd.exe
0
2
4
6
8
Press any key to continue . . .
```

الحلقة for المتداخلة:

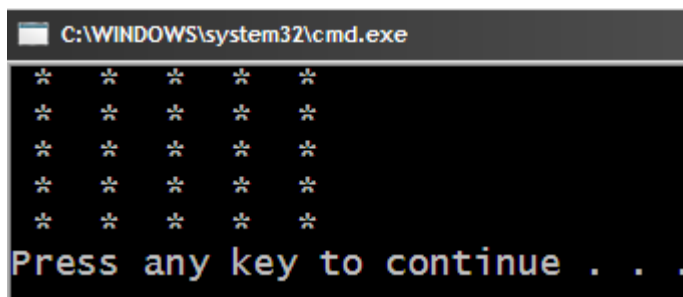
في هذا المثال نستخدم حلقة for داخل حلقة for اخرى

```

1 // use nested for statement
2 class forloop
3 {
4     public static void main(String args[])
5     {
6         for(int i = 0; i < 5; i++)
7         {
8             for(int j = 0; j < 5; j++)
9                 System.out.print(" * ");
10                System.out.println();
11            }
12        }
13    }

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
Press any key to continue . . .

```

### ملاحظات حول الحلقات التكرارية :

- عندما نعلم سلفاً عدد التكرارات التي ستنفذها الحلقة، الأفضل استخدام حلقة **for**.
- إذا كنا لا نعلم عدد التكرارات تحديداً، وخصوصاً إذا كان التكرار يعتمد على قيمة يقوم بإدخالها المستخدم، في هذه الحلقة يفضل استخدام **while** أو **while do**.
- إذا كنا نحتاج لعدد لمعرفة رقم التكرار أو استخدام قيمته في البرنامج يمكن استخدام حلقة **for** للاستفادة من عددها، حيث أن قيمته تبين رقم التكرار.
- إذا كان من الممكن ألا يتم تنفيذ الحلقة أصلاً، فالأصح استخدام حلقة **while**، أما إن كان تنفيذ الحلقة يكتمل للمرة الأولى في كل الأحوال، يتساوى حينها استخدام **while** و **while do**.
- عموماً عند استخدام لغة **Java** يمكن أن نعبر عن أي فكرة بها تكرار بأي من العبارات التكرارية الثلاث التي توفرها اللغة، وبصورة سليمة وصحيحة، ولكننا دائماً نختار الحلقة الأمثل والأفضل والتي تجعل كتابة

البرنامج أسهل وأقل تعقيداً، وتؤدي المطلوب بصورة أكفأ وذلك حسب خواص الحلقة وطبيعة البرنامج المطلوب.

# الفصل الثاني: المصفوفات

## المصفوفات

---

### المصفوفات Arrays:

المصفوفة عبارة عن صف من البيانات ذات علاقة ببعضها من نفس نوع البيانات، يكون للمصفوفة اسم واحد وعدد من الحجرات توضع بها البيانات.

### المصفوفات احادية البعد:

الصيغة العامة لتعريف المصفوفة احادية البعد

```
DataType[] VariableName = new DataType[Number];
```

او

```
DataType VariableName[] = new DataType[Number];
```

والمثال التالي يوضح تعريف مصفوفة من النوع int

```
int array[] = new int[5];
```

وضع قيم ابتدائية لعناصر المصفوفة:

الشكل التالي يوضح كيفية وضع قيم اولية للمصفوفة array

```
array[0] = 1;  
array[1] = 2;  
array[2] = 3;  
array[3] = 4;  
array[4] = 5;
```

تعريف المصفوفة واعطاءها قيم اولية

```
int array[] = new int[]{1,2,3,4,5};
```

طباعة عنصر من المصفوفة

```
System.out.println(array[3]);
```

لطباعة كل عناصر المصفوفة نستخدم حلقة for

```
for(int i = 0; i < array.length; i++)  
    System.out.print(" " + array[i]);
```

مثال على المصفوفات احادية البعد

```
1 import java.util.Scanner;  
2 class student  
3 {  
4     public static void main(String args[])  
5     {  
6         Scanner in = new Scanner(System.in);  
7         String name[] = new String[3];  
8         int degree[] = new int[3];  
9         for(int i = 0; i < 3; i++)  
10        {  
11            System.out.print(" Enter Name : ");  
12            name[i] = in.next();  
13            System.out.print(" Enter Degree : ");  
            degree[i] = in.nextInt();  
        }  
        System.out.println(" Name " + " *** " + " Degree ");  
        for(int j = 0; j < name.length; j++)  
        {  
            System.out.println(name[j] + " *** " + degree[j]);  
        }  
    }  
}
```

الخروج من البرنامج

```
C:\WINDOWS\system32\cmd.exe
Enter Name : md
Enter Degree : 78
Enter Name : ea
Enter Degree : 88
Enter Name : mn
Enter Degree : 77
Name *** Degree
md *** 78
ea *** 88
mn *** 77
Press any key to continue . . .
```

### المصفوفات متعددة البعد:

الصيغة العامة لتعريف مصفوفة متعددة البعد

```
int arr2[][] = new int[2][3];
```

تعريف ووضع قيم أولية لمصفوفة متعددة الأبعاد

```
int arr2[][] = new int[][] {{1,2,3},{4,5,6}};
```

طباعة عنصر محدد من المصفوفة متعددة الأبعاد

```
System.out.print(arr2[0][1]);
```

لطباعة كل عناصر المصفوفة نستخدم حلقات for متداخلة

```

for(int i = 0;i < arr2.length;i++)
{
    for(int j = 0;j < arr2[i].length;j++)
    {
        System.out.print(arr2[i][j] + " ");
    }
    System.out.println();
}

```

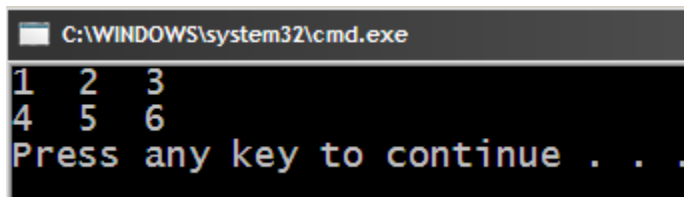
مثال على المصفوفة متعددة البعد

```

1  import java.util.*;
2
3  class arraytest1
4  {
5      public static void main(String args[])
6      {
7          int arr2[][] = new int[][] {{1,2,3},{4,5,6}};
8          for(int i = 0;i < arr2.length;i++)
9          {
10             for(int j = 0;j < arr2[i].length;j++)
11             {
12                 System.out.print(arr2[i][j] + " ");
13             }
14             System.out.println();
15         }
16     }
17 }

```

الخروج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
1 2 3
4 5 6
Press any key to continue . . .

```

## الدوال في الجافا

---

### الدوال في الجافا:

الدالة هي مجموعة من التعليمات التي تؤدي وظيفة معينة، يتم نداؤها خلال البرنامج عند الحاجة بواسطة اسمها. في لغة **Java** تسمى الدوال `methods`. وهناك الكثير من الدوال الموجودة والمعروفة أصلاً في لغة **Java** والتي يمكن أن نستخدمها عندما نريد. من أكثر دوال لغة **Java** التي استخدمناها خلال الأمثلة الدالة `print` والدالة `println`، وهما دالتان الغرض منهما الطباعة على الشاشة.

تحتوي لغة **Java** على عدد هائل جداً من الدوال ذات الوظائف المختلفة في شتى المجالات، ولا يتسع المجال لذكرها، بل ولا يمكن حصر جميع الدوال في متناول اليد، ولكن يبحث المبرمج عن الدوال التي يحتاجها بناءً على مجالها. ولاستخدام هذه الدوال لا بد من معرفة طريقة كتابتها ونوع وعدد الوسائط التي تأخذها.

### الدوال الجاهزة:

دوال ال `math class`:



الطريقة	وصف الطريقة	مثال
<code>abs (x)</code>	القيمة المطلقة لـ $x$ .	<code>Math.abs (6.2) → 6.2</code> <code>Math.abs (-2.4) → 2.4</code>
<code>ceil (x)</code>	تقرب $x$ إلى أقل عدد صحيح ليس أقل من $x$ .	<code>Math.ceil (5.1) → 6</code> <code>Math.ceil (-5.1) → -5</code>
<code>floor (x)</code>	تقرب $x$ إلى أكبر عدد صحيح ليس أكبر من $x$ .	<code>Math.floor (5.1) → 5</code> <code>Math.floor (-5.1) → -6</code>
<code>max (x, y)</code>	أكبر قيمة من $x$ و $y$ .	<code>Math.max (7, 6) → 7</code>
<code>min (x, y)</code>	أقل قيمة من $x$ و $y$ .	<code>Math.min (-7, -8) → -8</code>
<code>pow (x, y)</code>	$x$ مرفوعة للأس $y$ .	<code>Math.pow (6, 2) → 6<sup>2</sup> → 36</code>
<code>sqrt (x)</code>	الجذر التربيعي لـ $x$ .	<code>Math.sqrt (9) → <math>\sqrt{9}</math> → 3</code>
<code>random ()</code>	تكوّن رقم عشوائي بين الصفر والواحد.	<code>Math.random () → 0.23121</code>

هذه بعض الدوال الموجودة في الفئة `math`

الدوال الخاصة بالسلاسل:

## إيجاد طول السلسلة الرمزية:

ترجع الطريقة <code>length()</code> طول السلسلة الرمزية <code>s</code> .	<code>s.length()</code>
عمليات المقارنة بين سلسلتين رمزيتين (ملاحظة: لا تستخدم <code>==</code> و <code>!=</code> ).	
تقوم الطريقة بمقارنة السلسلة الرمزية <code>s</code> مع السلسلة الرمزية <code>t</code> وتعيد رقم سالب إذا كانت <code>s</code> أقل من <code>t</code> وتعيد صفر إذا كانت <code>s</code> تساوي <code>t</code> وتعيد رقم موجب إذا كانت <code>s</code> أكبر من <code>t</code> .	<code>s.compareTo(t)</code>
تعمل هذه الطريقة بنفس عمل الطريقة <code>compareTo()</code> ولكن مع إهمال حالة الحروف (صغيرة أم كبيرة).	<code>s.compareToIgnoreCase(t)</code>
تعيد <code>true</code> إذا كان <code>s</code> يساوي <code>t</code> .	<code>s.equals(t)</code>
تعمل هذه الطريقة بنفس عمل الطريقة <code>equals()</code> ولكن مع إهمال حالة الحروف (صغيرة أم كبيرة).	<code>s.equalsIgnoreCase(t)</code>
تعيد <code>true</code> إذا كان <code>s</code> يبدأ بالسلسلة الرمزية <code>t</code> .	<code>s.startsWith(t)</code>
تعيد <code>true</code> إذا كانت السلسلة الرمزية <code>t</code> موجودة في <code>s</code> بدءاً من الموقع <code>i</code> .	<code>s.startsWith(t, i)</code>
تعيد <code>true</code> إذا كان <code>s</code> تنتهي بـ <code>t</code> .	<code>s.endsWith(t)</code>

### عمليات البحث:

كل طرق `indexOf()` تقوم بإرجاع -1 إذا كان العنصر المراد البحث عنه غير موجود، ويمكن للعنصر المراد البحث عنه أن يكون حرف أو سلسلة رمزية.

ترجع موقع أول مكان توجد فيه <code>t</code> داخل السلسلة الرمزية <code>s</code> .	<code>s.indexOf(t)</code>
ترجع موقع أول مكان توجد فيه <code>t</code> داخل السلسلة الرمزية <code>s</code> بعد الموقع <code>i</code> .	<code>s.indexOf(t, i)</code>
ترجع موقع أول مكان يوجد فيه الحرف المخزن في المتغير <code>c</code> داخل السلسلة الرمزية <code>s</code> .	<code>s.indexOf(c)</code>
ترجع موقع أول مكان يوجد فيه الحرف المخزن في المتغير <code>c</code> داخل السلسلة الرمزية <code>s</code> بعد الموقع <code>i</code> .	<code>s.indexOf(c, i)</code>
ترجع موقع آخر مكان يوجد فيه الحرف المخزن في المتغير <code>c</code> داخل السلسلة الرمزية <code>s</code> .	<code>s.lastIndexOf(c)</code>
ترجع موقع آخر مكان توجد فيه السلسلة الرمزية <code>t</code> داخل السلسلة الرمزية <code>s</code> .	<code>s.lastIndexOf(t)</code>

عمليات أخذ جزء من السلسلة الرمزية `string`.

ترجع الحرف الموجود في الموقع <code>i</code> داخل السلسلة الرمزية <code>s</code> .	<code>s.charAt(i)</code>
ترجع جزء من السلسلة الرمزية <code>s</code> بدءاً من الموقع <code>i</code> وحتى النهاية.	<code>s.substring(i)</code>
ترجع جزء من السلسلة الرمزية <code>s</code> بدءاً من الموقع <code>i</code> وحتى الموقع <code>j-1</code> .	<code>s.substring(i, j)</code>

عمليات التعديل على السلسلة الرمزية `string` وإنشاء سلسلة رمزية جديدة.

إنشاء سلسلة رمزية جديدة تحتوي كل ما في السلسلة الرمزية <code>s</code> بعد تحويل كل الحروف إلى حروف صغيرة.	<code>s.toLowerCase()</code>
إنشاء سلسلة رمزية جديدة تحتوي كل ما في السلسلة الرمزية <code>s</code> بعد تحويل كل الحروف إلى حروف كبيرة.	<code>s.toUpperCase()</code>
إنشاء سلسلة رمزية جديدة من السلسلة الرمزية <code>s</code> بعد الفارغ من البداية والنهاية.	<code>s.trim()</code>
إنشاء سلسلة رمزية جديدة من السلسلة الرمزية <code>s</code> بعد تبديل كل <code>c1</code> بـ <code>c2</code> ، وهما من نوع <code>char</code> .	<code>s.replace(c1, c2)</code>

عمليات أخرى على السلاسل الرمزية string.	
<code>s.matches(regexStr)</code>	ترجع هذه الطريقة <code>true</code> إذا كانت السلسلة الرمزية <code>regexStr</code> تطابق السلسلة الرمزية <code>s</code> كاملة.
<code>s.replaceAll(regexStr, t)</code>	إنشاء سلسلة رمزية <code>string</code> جديدة بعد تبديل كل <code>regexStr</code> بـ <code>t</code> .
<code>s.replaceFirst(regexStr, t)</code>	إنشاء سلسلة رمزية <code>string</code> جديدة بعد تبديل أول <code>regexStr</code> بـ <code>t</code> .
<code>s.split(regexStr)</code>	إنشاء مصفوفة تحتوي على أجزاء من السلسلة الرمزية <code>s</code> مقسمة حسب ظهور <code>regexStr</code> .
<code>s.split(regexStr, count)</code>	كما في الطريقة <code>split(regexStr)</code> لكن مع تحديد عدد مرات التقسيم.

الدوال المعرفة بواسطة المستخدم :

الشكل العام لتعريف الدالة

```

access static return_type method_name(parameters)
{
    statement1;
    statement2;
    .
    .
    .
}

```

Access محدد الوصول ويكون `public` او `private` او `protected`  
`Static` تستخدم لتعريف الدالة ليتم استخدامها داخل الصنف الذي عرفت فيه فقط.

Return\_type يحدد نوع القيم التي تعيدها الدالة

Method\_name اسم الدالة

Parameters هي المعاملات . وعند تعريف الدالة تسمى هذه المعاملات بالمعاملات الشكلية ( Formal

Parameters) وعند استدعاء الدالة تسمى بالمعاملات الفعلية (Actual Parameters)

مثال :

```
public static void university()  
{  
    System.out.println(" Alzaim Alazhari University ");  
}
```

دالة لا تعيد قيمة ولا تحمل وسائط هذه الدالة تقوم بطباعة النص Alzaim Alazhari University

اشكال الدوال:

- دالة لا تأخذ وسائط ولا تعيد قيمة
- دالة تأخذ وسائط ولا تعيد قيمة
- دالة تأخذ وسائط وتعيد قيمة
- دالة لا تأخذ وسائط وتعيد قيمة

مثال يوضح اشكال الدوال :

```

public static void method1 ()
{
    System.out.println("method 1");
}
public static void method2 (String method2)
{
    method2 = "method 2";
    System.out.println(method2);
}
public static int method3 (int x)
{
    return x * x;
}
public static int method4 ()
{
    int x, pi = 3.14;
    return x * pi;
}

```

**استدعاء الدوال:**

يتم استدعاء الدالة باسمها كما في الشكل التالي

```

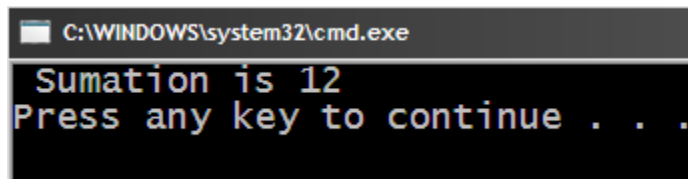
university ();

```

مثال:

```
1  class method
2  {
3      static int sum(int a, int b)
4      {
5          return a * b ;
6      }
7      public static void main(String[] args)
8      {
9
10         System.out.println(" Sumation is " + sum(3,4)) ;
11     }
12 }
```

الخرج من البرنامج



C:\WINDOWS\system32\cmd.exe

Sumation is 12  
Press any key to continue . . .

النداء الذاتي:

هو ان تقوم الدالة باستدعاء نفسها بنفسها . البرنامج التالي يقوم بايجاد مضروب العدد n باستخدام النداء الذاتي

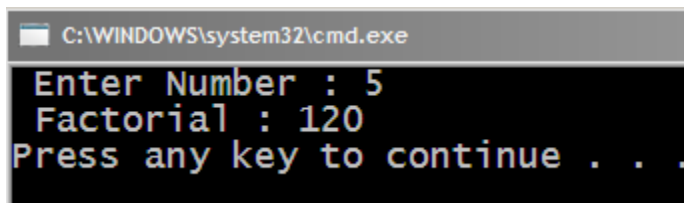


```

1  import java.util.*;
2  class recursion
3  {
4      static int fact(int a)
5      {
6          if(a == 1 || a == 0)
7              return 1;
8          else
9              return a * fact(a - 1);
10     }
11     public static void main(String args[])
12     {
13         Scanner in = new Scanner(System.in);
14         int num;
15         System.out.print(" Enter Number : ");
16         num = in.nextInt();
17         System.out.println(" Factorial : " + fact(num));
18     }
19 }
20

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
Enter Number : 5
Factorial : 120
Press any key to continue . . .

```

ملاحظات:

- عند استخدام النداء الذاتي يجب الانتباه إلى ضرورة وجود شرط معين لإيقاف النداء الذاتي، وإلا ستتواصل النداءات لعدد لانهائي من المرات، وعندها لا يتوقف البرنامج عن التنفيذ .
- عند استخدام النداء الذاتي يجب الاحتراس والتأكد من وجود شرط توقف النداءات. لكن الأفضل استبداله بالحلقات لأن تنفيذ البرنامج بالنداء الذاتي يستغرق زمناً أطول في التنفيذ ويستهلك ذاكرة أكبر من تنفيذ نفس البرنامج باستخدام الحلقات.

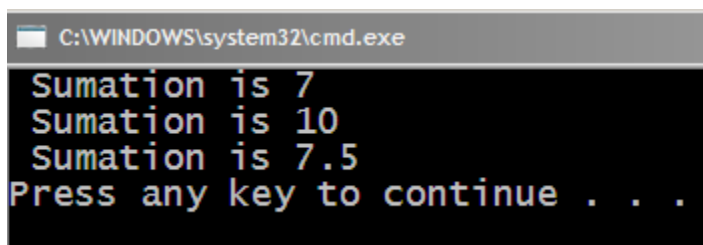
## تحميل الدوال بشكل زائد :

تتم عملية تحميل الدوال بشكل زائد عندما تكون هناك اكثر من دالة تحمل نفس الاسم في نفس الفئة ويتم التمييز بين هذه الدوال من خلال عدد المعاملات التي تحملها وانواعها

مثال :

```
1  class method
2  {
3      static int sum(int a, int b)
4      {
5          return a + b ;
6      }
7      static int sum(int a,int b,int c)
8      {
9          return a + b + c;
10     }
11     static double sum(double a,double b)
12     {
13         return a + b;
14     }
15     public static void main(String[] args)
16     {
17
18         System.out.println(" Sumation is " + sum(3,4));
19         System.out.println(" Sumation is " + sum(3,4,3));
20         System.out.println(" Sumation is " + sum(3.2,4.3));
21     }
22 }
```

الخرج من البرنامج



```
C:\WINDOWS\system32\cmd.exe
Sumation is 7
Sumation is 10
Sumation is 7.5
Press any key to continue . . .
```

# الفصل الثالث: البرمجيات بالكائنات

## البرمجة بالكائنات:

تعتبر لغة جافا من أشهر لغات البرمجة بالكائنات Object Oriented Programming Languages تقوم البرمجة بالكائنات على مبدأ أن كل فكرة أو موضوع في النظام المعين هو عبارة عن كائن object له صفات properties وسلوك behavior يظهر بها في النظام ويتفاعل عن طريقها مع الكائنات الأخرى في النظام. وتقوم على مبدأ أن النظام هو مجموعة من الكائنات التي تحتوي على صفاتها الخاصة والتي لا تسمح الكائنات الأخرى بالوصول إليها. وهذه الكائنات تتفاعل مع بعضها البعض بواسطة طرق محددة سلفاً وهي الدوال الخاصة بالكائن.

الصف class هو عبارة عن هيكل برمجي يحتوي على بيانات attributes ودوال methods تصف معا الشكل الذي ستكون عليه الكائنات عند تشغيل البرنامج. هذا يعني أن الـ class يمثل قالباً تصنع منه الكائنات في البرنامج، أي أنه يتم تصميم الصف مرة واحدة ثم اشتقاق أي عدد من الكائنات من هذا الصف. يمكن أن يتم إنشاء الكائنات داخل الدالة الرئيسية main أو بداخل كائن آخر إذا كان يتعامل معه. ولأن لغة جافا لا تسمح بتنفيذ أي برنامج إلا إذا احتوى على class ، كان من اللازم تعريف class لكل برنامج قمنا بتنفيذه من قبل رغم أننا لم نستخدمه بالطريقة القياسية، والأمر كذلك بالنسبة إلى الدالة الرئيسية main والتي يبدأ منها التنفيذ. وفيما يلي مثال لصف وكيفية استخدامه لاشتقاق عدد من الكائنات والتعامل معها.

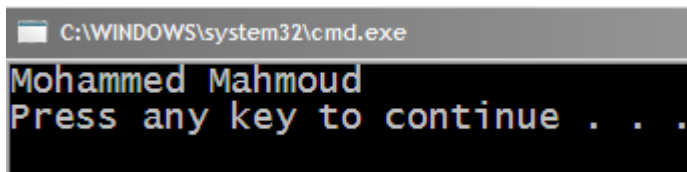
**مثال:**

```

1  class student
2  {
3      public String name;
4      public void printname()
5      {
6          System.out.println(name);
7      }
8  }
9  class classtest
10 {
11     public static void main(String args[])
12     {
13         student std = new student();
14         std.name = "Mohammed Mahmoud";
15         std.printname();
16     }
17 }

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
Mohammed Mahmoud
Press any key to continue . . .

```

يعرف هذا البرنامج الفئة student والتي تحتوى على عضو بياني واحد من النوع String ودالة لطباعة هذا الاسم

في الدالة main في السطر رقم 13 قمنا بانشاء كائن من الفئة student يحمل الاسم std تقوم الكلمة المحجوزة new بحجز موقع جديد بالذاكرة بالحجم الذي يحتاجه الكائن لتخزين بياناته ودواله، ويشار لهذا الموقع في الذاكرة باسم الكائن لنستطيع التعامل مع هذا الموقع فيما بعد. باستخدام اسم الكائن متبوعاً بنقطة نستطيع الوصول إلى محتويات الكائن من بيانات ودوال، في السطر رقم 14 وضعنا قيمة في المتغير name وقمنا بتنفيذ الدالة printName للكائن std .

### محددات الوصول:

- Public عندما يتم الاعلان عن محدد الوصول عام فان هذا العضو يمكن الوصول اليه من جميع الفئات الاخرى
- Private عندما يعن عن محدد الوصول خاص فان هذا العضو يستخدم داخل الفئة فقط ولا تستطيع الفئات الاخرى استخدامه
- Protected عندما يكون محدد الوصول محمي فان هذا العضو يستخدم داخل الفئة والفئات المشتقة فقط

### المشيدات : Constructor

1. هي دالة تحمل نفس اسم الفئة
2. يتم تنفيذها تلقائياً عند انشاء كائن من الفئة
3. تستخدم هذه الدالة لإجراء العمليات التي نرغب في تنفيذها ابتداءً لحظة إنشاء الكائن وقبل تعامل أي جهة مع هذا الكائن
4. يمكن للمشيدات ان تحمل وسائط لكنها لا ترجع اي قيمة حتى void .  
الصيغة العامة لتعريف مشيد

```
access classname(parameters)
{
    statement ;
}
```

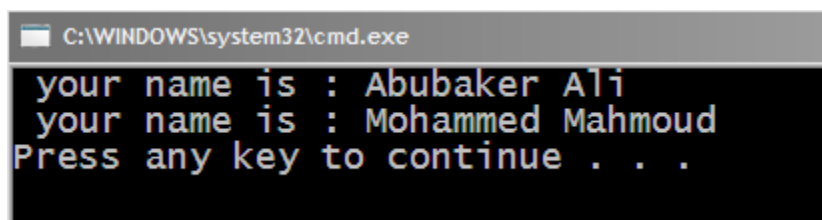
مثال:

```

1  class student
2  {
3      public String name;
4      public student()
5      {
6          name = "Abubaker Ali";
7      }
8      public void printname()
9      {
10         System.out.println(" your name is : " + name);
11     }
12 }
13 class Constuctor
14 {
15     public static void main(String args[])
16     {
17         student std = new student();
18         std.printname();
19         std.name = "Mohammed Mahmoud";
20         std.printname();
21     }
22 }
23

```

الخرج من البرنامج



```

C:\WINDOWS\system32\cmd.exe
your name is : Abubaker Ali
your name is : Mohammed Mahmoud
Press any key to continue . . .

```

السطور من 1 الى 12 تم تعريف الفئة student والذي يحتوى على الخاصية name و الدالة printname() والمشييد student() والذي يتم فيه اعطاء قيمه اولية للخاصية name .  
 في السطر 17 تم انشاء كائن من الفئة student بالاسم std .  
 ويمكن ان تحتوى الفئة على اكثر من مشيد تختلف في عدد ونوع بيانات الوسائط وفي هذه الحالة يعرف بالتحميل الزائد للمشييدات constructor overloading .

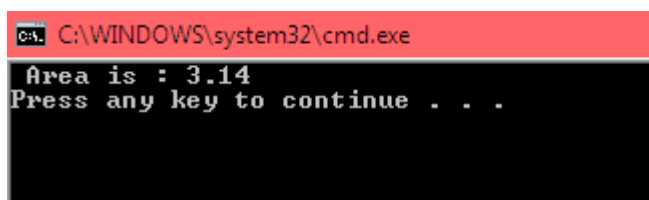
## المؤشر this:

يستطيع كل class أن يشير إلى محتوياته من متغيرات ودوال باستخدام المؤشر this

والمثال التالي يوضح كيف استخدام المؤشر this

```
1 public class Circle
2 {
3     int radius ;
4     public Circle()
5     {
6         radius = 1;
7     }
8     public Circle(int radius)
9     {
10        this.radius = radius;
11    }
12    public double getArea()
13    {
14        return 3.14 * radius * radius ;
15    }
16    public static void main(String args[]) // Main of program
17    {
18        Circle c = new Circle();
19        System.out.println(" Area is : " + c.getArea());
20    }
21 }
```

وهذا الخرج من البرنامج:



```
C:\WINDOWS\system32\cmd.exe
Area is : 3.14
Press any key to continue . . .
```

في السطر رقم 10 العبارة this.radius تشير الي العضو البياني radius في الفئة Circle وليس الوسيط radius الممرر للمثيد . radius



## الوراثة:

تعتبر الوراثة واحدة من أهم المميزات التي توفرها لغات البرمجة بالكائنات، وتسمح بالاستفادة من خصائص ودوال الفئات مسبقا لتعريف فئات جديدة، بحيث لا يضطر المبرمج إلى إعادة كتابة تلك الخصائص لمرة ثانية، ولعمل علاقات جديدة تربط بين الكائنات. عندما ترث فئة معينة خصائص فئة أخرى تسمى الفئة الوارثة بالابن subclass وتسمى الفئة الموروثة بالأب superclass. يمكن أن تكون الفئة الأب أي فئة معرفة سابقا وتأخذ شكل الفئات التي تعرضنا لها سابقا. ولكي ترث فئة معينة فئة معرفة سلفا يتم تحديد هذه العلاقة بالكلمة المحجوزة extends التي تظهر بعد اسم الفئة مباشرة يليها اسم الفئة التي سترثها هذه الفئة،

الشكل العام لعملية الوراثة :

```
class superclass
{
    attributes ;
    .
    .
    method ;
    .
    .
}
class subclass extends superclass
{
    attributes ;
    .
    .
    method ;
    .
    .
}
```

الشكل اعلاه يوضح عملية الوراثة

محددات الوصول في الوراثة:

1. public : يمكن الوصول للمتغيرات والدوال المعرفة public من داخل الفئة المعرفة بها ومن داخل الفئات التي ترث تلك الفئة وباستخدام أسماء الكائنات المعرفة من نوع تلك الفئة.
2. private : يمكن الوصول للمتغيرات والدوال المعرفة private من داخل الفئة المعرفة بها ، ولكن لا يمكن الوصول إليها من داخل الفئات التي ترث تلك الفئة ولا باستخدام أسماء الكائنات المعرفة من نوع الفئة ، أي أنها خاصة بالفئة فقط.
3. protected : يمكن الوصول للمتغيرات والدوال المعرفة protected من داخل الفئة المعرفة بها ومن داخل الفئات التي ترث تلك الفئة، ولكن لا يمكن الوصول إليها باستخدام أسماء الكائنات المعرفة من نوع الفئة، أي أنها خاصة بالفئة والفئات التي ترث منها.

**مثال:**

```

1  import java.util.*;
2  class person
3  {
4      protected String name ;
5      protected int age ;
6      public void set()
7      {
8          Scanner sc = new Scanner(System.in);
9          System.out.print(" Enter Name : ");
10         name = sc.next();
11         System.out.print(" Enter age : ");
12         age = sc.nextInt();
13     }
14 }
15 class student extends person
16 {
17     double degree = 0.0;
18     public void display()
19     {
20         System.out.println(" Name " + "\t age" + "\t degree");
21         System.out.println(name + "\t" + age + "\t" + degree);
22     }
23 }
24 public class Inher
25 {
26     public static void main(String args[])
27     {
28         student std = new student();
29         std.set();
30         std.degree = 97;
31         std.display();
32     }
33 }

```

الخرج من البرنامج

```

C:\WINDOWS\system32\cmd.exe
Enter Name : kojo
Enter age : 20
Name      age      degree
kojo      20       97.0
Press any key to continue . . .

```

في المثال السابق الفئة student ترث الفئة person ونلاحظ على الرغم من ان الدالة set() معرفة في الفئة person الا اننا استطعنا استخدامها بواسطة الكائن std التابع للفئة student وذلك لان الفئة student ورثت متغيرات ودوال الفئة person .

## التجريد :Abstraction

الفئة مجردة هي فئة اب super class لا يمكن انشاء كائن من الفئة المجردة لان الفئة المجردة تحتوى على الاقل على دالة واحدة غير مكتملة incomplete . ولعمل فئة مجردة نستخدم الكلمة المحجوزة abstract تحتوى على الاقل على دالة واحدة مجردة abstract ويتم عمل تحميل override عليها . وتعني override امكانية تعديل الدالة في الفئات التي ترث الفئة المجردة .

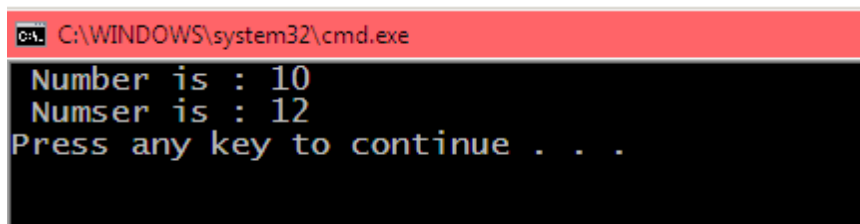
المثال التالي يوضح مفهوم التجريد :-

```

1  abstract class B // abstract class
2  {
3      protected int num ;
4      public abstract void show(); // abstract function
5  }
6  class A extends B
7  {
8      public void show()
9      {
10         num = 10;
11         System.out.println(" Number is : " + num);
12     }
13 }
14 class C extends B
15 {
16     public void show()
17     {
18         num = 12;
19         System.out.println(" Numser is : " + num);
20     }
21 }
22 public class Abstract
23 {
24     public static void main(String args[])
25     {
26         A a = new A();
27         a.show();
28         C b = new C();
29         b.show();
30     }
31 }

```

الخرج من البرنامج:



```

C:\WINDOWS\system32\cmd.exe
Number is : 10
Numser is : 12
Press any key to continue . . .

```