مقدمة في برمجيات الحاسوب

البـــــــــــاب الأول

مقـــدمة:

الحاسب، Computer، هو عبارة عن جهاز إلكتروني يقوم باستقبال البيانات، ثم معالجتها، وتخزينها، وإظهارها للمستخدم بالصورة المناسبة (ويطلق عليه باللغة العربية الحاسوب).

وبالطبع، لابد للحاسب، حتى يقوم بتلك الوظائف، من أجهزة خاصة تساعده، فهناك أجهزة خاصة للإدخال وأخرى للمعالجة وثالثة للتخزين، الخ.

وإذا نظرنا للحاسوب نظرة شاملة، نجد أنه لا يقوم فقط باستقبال البيانات، ومن ثم معالجتها حسب رغبتنا، وإخراج نتائج عمليات المعالجة وتخزينها، بل يمكنه أيضا نقلها إلى جهاز حاسوب آخر، أى تبادل المعلومات بين الحواسب وبعضها، عند ربطها معاً فيما يسمى بالشبكات.

والآن نأتي إلى بعض المصطلحات والمفاهيم الخاصة بالحاسوب:

البيانات (data):

هى أية معلومات مكتوبة بطريقة تمكن الحاسوب من أن يتعامل معها، فالمعلومات التى لا يستطيع الحاسوب التعامل معها لا تعتبر بيانات بالنسبة له.

- **المعالجة (processing):**

هى عملية تحويل البيانات من شكل إلى آخر.

- **إخراج البيانات (data output):**

هى عملية إظهار أو استرجاع البيانات إلى شكل يتمكن مستخدم الحاسوب من فهمها، واستخدامها.

- **التخزين (storage):**

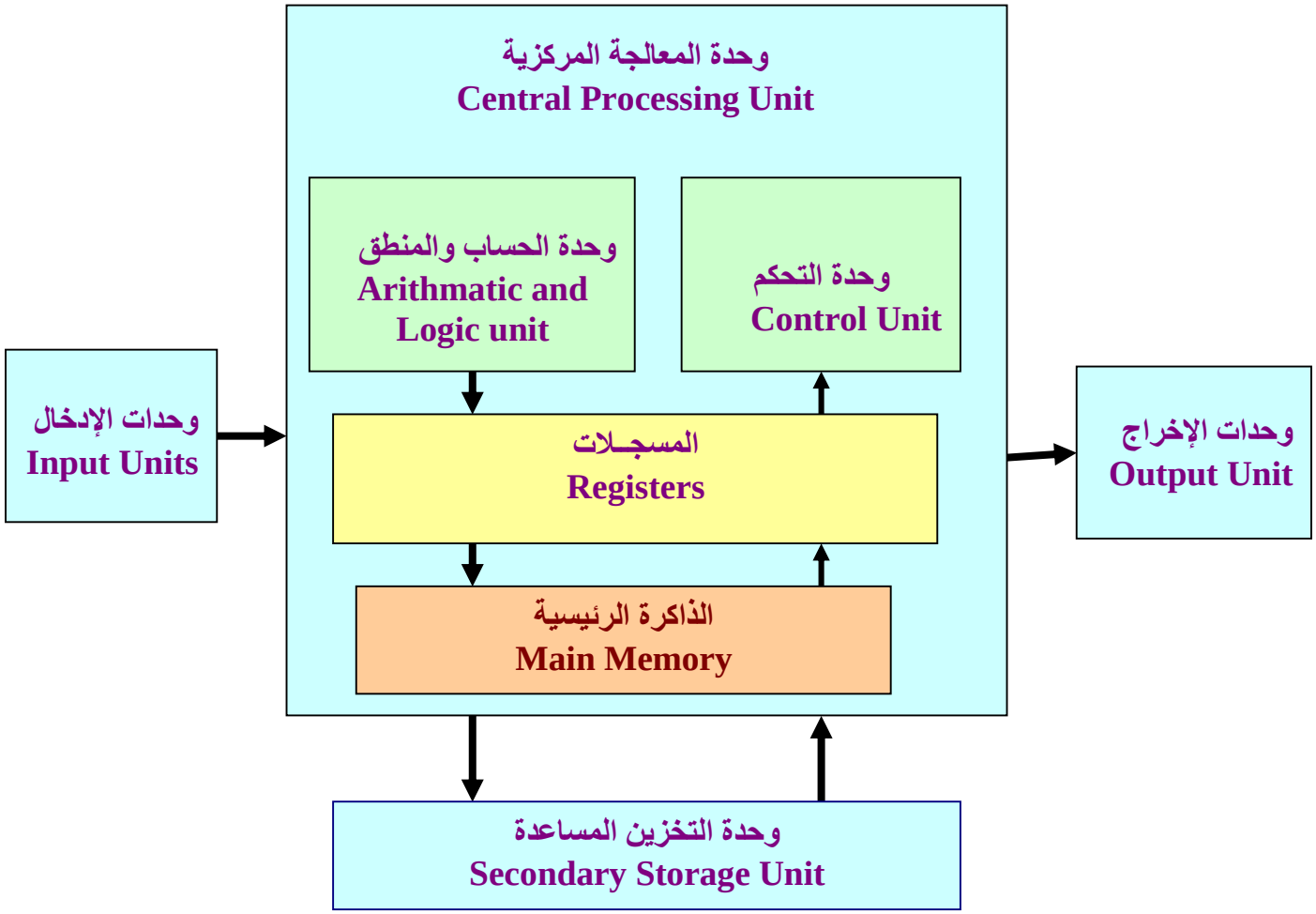
هى عملية الاحتفاظ بالبيانات لاسترجاعها لاحقا.

- **الشبكات (networks):**

هى مجموعة من أجهزة الحاسـوب مرتبطة مع بعضــها البعض، لتتمكن من تبادل البيانات مع بعضها البعض.

1-1) نظم الحاسب Computer Systems :

يتكون نظام الحاسوب من المكونات الأساسية الآتية:



شكل (1-1)

(1) المكونات المادية، أو الأجهزة (Hardware):

وهى المكونات المادية الملموسة في الحاسوب الآلي، مثل لوحة المفاتيح، الشاشة، الفأرة، القرص الصلب، الخ.

(2) البرامج (Software):

وهى مجموعة التعليمات والأوامر التي تستخدم للحصول على النتائج المطلوبة من الحاسوب الآلي.

2-1) المكونات الصلبة (المادية) للحاسب (Hardware):

1- وحدات الإدخال (Input units):

تستخدم لإدخال البيانات إلى جهاز الحاسوب، ومن وحدات الإدخال المعروفة ما يأتي:

- لوحة المفاتيح (Keyboard)
- الفأرة (Mouse)
- شاشة اللمس (Touch Screen)

1. وحدات إدخال الصوت:
2. وحدات إدخال الصور:

أ. الكاميرا الرقمية

ب. الماسحات الضوئية

الفأرة

لوحة مفاتيح

الماسح الضوئي

لوحة اللمس

ميكروفون

كاميرا ويب

عصا التوجيه

شكل (1-2) بعض وحدات الإدخال

2- وحدات الإخراج (Output units):

وتستخدم هذه الوحدات لعرض النتائج على الشاشة أو طباعتها على الورق ومن وحدات الإخراج المعروفة ما يأتي:

- الشاشة (screen)
- الطابعات (printers):

أ. طابعة المصفوفة النقطية (dot matrix printer)

ب. طابعة نفث الحبر (ink jet printer)

حـ. طابعة الليزر (laser printer)

- الراسمات (scanners)
- وحدات إخراج الصوت (speakers)

طابعات

السماعات

الشاشة

شكل (1-3) بعض وحدات الإخراج

3- وحدة المعالجة المركزية (Central processing units):

تتم المعالجة الفعلية للبيانات في وحدة المعالجة المركزية، (CPU)، التي تتكون في الحواسب الصغيرة من رقيقة معالج دقيق (ميكروي) واحد (Microprocessor). أما في أجهزة الحاسوب الكبيرة، فتتكون من أنواع مختلفة من الرقائق والدوائر. تعتمد سرعة الحاسوب ونوع البرمجيات التي يمكن تشغيلها على نوع المعالج

الدقيق الميكروي الموجود فيه. وفي كلتا الحالتين: هناك ثلاثة مكونات رئيسية لوحدة المعالجة المركزية هى:

شكل (1-4) المعالج الدقيق

أ- وحدة الحساب والمنطق (Aarithmetic and Logical Unit) (ALU)

وهى الوحدة التي تقوم بمعالجة البيانات، من ترتيب وتصنيف وفرز، ومعالجة العمليات الحسابية الأربع: الضرب والقسمة والجمع والطرح، وكذلك العمليات المنطقية، مثل عمليات المقارنة بين قيمتين (يساوى، أكبر من، أصغر من، الخ).

ب- وحدة التحكم (Control Unit)

هى مجموعة من الدوائر الإلكترونية، تقوم بتوجيه وتوظيف جميع مكونات نظام الحاسوب. وبالاعتماد على تعليمات البرامج الموجودة في الذاكرة الرئيسية، تعمل على نقل البيانات من وإلى وحدة الحساب والمنطق(ALU) والمسجلات والذاكرة الرئيسية وأجهزة الإدخال والإخراج، كما تخبر وحدة الحساب والمنطق (ALU) بالعمليات التي يجب أن تنفذها. إذن تقوم وحدة التحكم بالوظائف التالية:

- قراءة تعليمات البرنامج وتفسيرها.
- توجيه العمليات داخل وحدة المعالجة المركزية(CPU).
- التحكم فى تدفق البرامج من وإلى الذاكرة الرئيسية وأجهزة الإدخال والإخراج.

ج- المسجلات (Registers)

هى مواقع تخزين مؤقتة، "ذاكرة مؤقتة"، لإجراء بعض العمليات الوسيطة بسرعة عالية.

4- الذاكرة الرئيسية (Main Memory)

الذاكرة الرئيسية هى المكان الذي يتم فيه تخزين البيانات وتعليمات البرنامج ونتائج الحسابات والمخرجات بعد معالجتها. وفي الحاسبات الشخصية تتكون الذاكرة من رقائق مثبتة على لوحات صغيرة يتم تركيبها على اللوحة الأم في أماكن معينة. ومن العوامل الأساسية التي يمكن أن تتميز بها الذاكرة: سعتها، وكلما زادت سعتها كانت أفضل وأسرع فى معالجة البيانات.

يعتمد الحاسوب في بنائه على النظام الثنائي، والرقم الثنائي يختصر بالرمز بت (bit)، وهو أصغر جزء من البيانات التي يتعامل معها الحاسوب. والرقم الثنائي إما أن يكون صفراً أو واحداً ويمثل كل حرف أو رمز خاص أو رقم داخل الحاسوب بسلسلة من الأرقام الثنائية وعددها ثمانية تعرف بالبايت (Byte) وهى أصغر وحدة لقياس سعة الذاكرة وتتكون من **8** بت.وتتكون وحدة الذاكرة الرئيسية من:

1-ذاكرة القراءة فقط (الذاكرة الثابتة Rom):

وهى ذاكرة لا تفقد محتوياتها عند قطع التيار الكهربى عنها، وتبرمج بواسطة الشركة المنتجة.

2-ذاكرة الوصول العشوائي (الذاكرة المؤقتة Ram):

وهى تفقد محتوياتها بمجرد قطع التيار الكهربى عنها ويوجد منها عدة أشكال.

شكل (1-5)

5- وحـدات التخزين الثـانويـة (Auxiliary Memory)

و تعتبر الخزان الكبير الذي توضــع فيه كل البيانات والبرامج، وهى على اتصــال مباشــر بالذاكرة الرئيســية (العشــوائية) وذاكرة التخزين نوع دائم المغنطة تحتفظ بالمخزون بها من معلومات حتى بعد إغلاق الجهاز ومنها:

أ– القرص المرن (Floppy Disk):

ويوجد بمقاسات مختلفة و سعات مختلفة، منها:

3.5 بوصة وهو الأكثر استخداماً وسعته التخزينية 1.44 ميجابيت (1.44 مليون بايت)، (1.44M.B) ونلاحظ أنه قل استخدامه والحاجة إليه.

شكل (1-6) القرص المرن ومشغل القرص المرن Floppy Drive

ب- القرص الصلب (Hard Disk):

هناك نوعان من الأقراص الصلبة؛ أحدهما يمكن فصله عن الحاسوب ويوصل بالجهاز عند الاستخدام فقط، والنوع الآخر لا ينفصل عن الحاسوب. ويمتاز القرص الصلب عن القرص المرن بكثافة التسجيل وسعته التخزينية الكبيرة جداً.

شكل (1-7) القرص الصلب HD

ج- وحدة الأقراص المدمجة (CD ROM):

ويتم الكتابة على هذه الوحدة عن طريق شعاع من الليزر، ويمكنها تخزين سعات عالية (حتى 700 ميجا بايت)، ويوجد منها أنواع تسمى CD RW يمكن التسجيل عليها، كما يوجد أنواع أحدث تقوم كذلك بتخزين سعات أعلى من سابقتها (حتى 4.2 و8.4 جيجا بايت، أى4200 إلى 8400 ميجابايت) تسمى أقراص DVD، ويوجد منها أيضاً ما يمكن التسجيل عليه.

شكل

مشغل أقراص الليزر

شكل (1-8) مشغل CD

د- ذاكرة من نوع الفلاش (Flash Memory)

وهى إحدى التقنيات الحديثة للتخزين، ولها سعات مختلفة وتمتاز بصغر حجمها وسهولة حملها، ويسر استخدامها لنقل البيانات من وإلى الحاسوب بمجرد توصيلها به من الخارج.

شكل (1-9) بعض أنواع الذاكرات من نوع الفلاش

3-1) برامج الحاسوب (SOFTWARE)

البرمجيات (Software)

معدات الحاسوب وحدها لا تفي بالغرض، فهى تحتاج إلى برامج لتشغيلها. والبرنامج (Program) عبارة عن مجموعة من التعليمات المتسلسلة التي تخبر الحاسوب ماذا يفعل. أما البرمجيات (Software) فهى مصطلح عام يطلق على أى برنامج منفرد، أو مجموعة من البرامج والبيانات والمعلومات المخزنة. وبمقارنة البرمجيات بالمعدات التي تتكون من مواد صلبة كالمعادن والبلاستيك، فإن البرمجيات تبنى من المعرفة والتخطيط والفحص. ويسمى الشخص الذي يكتب أو يؤلف البرنامج "المبرمج (Programmer)". ويستخدم المبرمجون معرفتهم بكيفية عمل الحاسوب من أجل وضع مجموعة التعليمات التي تنجز وظائف مفيدة.

ويمكن تعريف البرمجيات: بأنها مجموعة من البرامج تستخدم لتشغيل نظام الحاسوب بأقصى طاقة ممكنة. وتضم البرمجيات ما يأتي:

- برامج نظم التشغيل (Operating System).
- حزم البرامج الجاهزة (Software Packages).
- برامج التطبيقات (General Purpose Software).

1) نظم التشغيل (Operating Systems)

نظام التشغيل هو الوسيط الذي يمكنك من خلاله أن تتعامل مع الحاسوب، فهو ينسق العمل بين مكونات ا لحاسوب المختلفة، ولنظم التشغيل أثر كبير على سرعة ودقة أداء الحاسوب. وبينما تستثمر أنظمة التشغيل عالية الكفاءة إمكانيات الحاسوب بالكامل، تعجز النظم الأقل كفاءة عن تحقيق ذلك.

وهناك نوعان من نظم التشغيل:

- **الأول** يعتمد على النصوص Text Based Software

- **والآخر** يعتمد على الصور والمرئيات Graphic Based Software

وتتعدد أنواع نظم التشغيل، فمنها ما هو مخصص للأجهزة الشخصية، مثل نظام النوافذ (Windows XP)، ومنها ما هو مخصص لأجهزة الحواسب الرئيسة فى شبكات الحواسب، أو الخادمة (Servers)، مثل (Windows Servers) ومنها ما هو مخصص لهما معا.

أمثلة لأنظمة التشغيل:

(1) نظام التشغيل "دوس"

(Microsoft disk operating system, MS-DOS)

نظام "دوس" قديم حيث ظهر فى أوائل الثمانيات، أنتجته شركة مايكروسوفت، وكان نقلة هامة في وقتها، مع أننا لم نعد نتعامل معه الآن في التطبيقات العادية، وهو يعتمد على النصوص؛ أى أنك مضطر لأن تكتب أوامر لكي ينفذ ا لحاسوب ما تريده، ولا يتم استخدام الفأرة (mouse) كما هو متبع اليوم عند استخدام نظام النوافذ (Windows).

(2) نظام النوافذ (Microsoft Windows)

نظام التشغيل نوافذ (Windows) من شركة مايكروسوفت، تم تسميته بالنوافذ لأنه يستخدم نوافذ تظهر أمامك، بالإضافة إلى أنك يمكنك تنفيذ الأوامر بطرق مختلفة خلال تلك النوافذ، ويعتبر هذا هو أشهر نظم التشغيل في العالم الآن بسبب سهولته الفائقة، فهو يعتمد على الرسومات التى تظهر على الشاشة أمام المستخدم للحاسوب، التى تعرف "بواجهة الاستخدام التصويرية"

(GUI Graphic User Interface)

وليس على النصوص، وتتمثل الأوامر على الواجهة برسومات، تعرف بالأيقونات (icons)، التى يؤدى النقر بالفأرة على إحداها إلى قيام الحاسوب بتنفيذ الأوامر التى ترمز

الأيقونة لها. كما توجد على الواجهة أيضـــاً قوائم لأوامر مختلفة، يمكن تنفيذها بمجرد الإشارة إليها بالفأرة مع النقر على أحد أزرارها.

2) برامج التطبيقات (Application Programs)

هى البرامج التي تقوم بمعالجة، أو تشـــغيل البيانات بهدف الوصـــول إلى المعلومات التي تلبي احتياج المستخدم أو مجموعة العمل أو الهيئة، أو أى قطاع معين، لذا فهى توفر نافذة على البيانات يستطيع المستخدم من خلالها الاطلاع عليها والإضافة إليها و الحذف منها و تعديلها.

وبرامج التطبيقات: يتم إعدادها و تصـميمها باسـتخدام أى لغة من لغات البرمجة، مثل لغة بيســـك المرئية (Visual Basic)، أو لغة ســـي المرئية (Visual C++)، أو لغة ســـى شارب(C#)، أو لغة جافا (Java).

إن البرنامج التطبيقي: هو نوع من البرامج يمكنك اســـتخدامه بعد تحميل نظام التشـــغيل، ومن أمثلتها برامج معالجة الكلمات مثل برنامج "word" وبرامج الجداول الإلكترونية، مثل برنامج "Excel" وبرامج قواعد البيانات "Data Base" وبرامج التصميمات المختلفة، مثل برنامج "AUTO CAD" للرسم الهندسي.

3) لغات الحاسب - البرمجة (Programming Languages):

كما أن اللغة وسيلة التخاطب بين الناس، فإن الحاسوب بحاجة إلى وسيلة للتخاطب والتفاهم بينه وبين المســـتخدمين تمكنهم من إعطائه الأوامر التى يقوم هو بتنفيذها. وتعد لغات البرمجة وسيلة التخاطب بين الإنسان والحاسوب. وتتكون لغة البرمجة من مجموعة من الرموز والأرقام تستخدم لكتابة التعليمات المعطاة للحاسوب وفق قواعد معينة تختلف من لغة لأخرى.

وقد تطورت لغات البرمجة مع تطور الحاسوب، واستمر الإنسان في تحسين وتسهيل لغات البرمجة، وتقريبها من لغة الإنسان العادية. ويمكن تصنيف لغات البرمجة على النحو الآتي:

- **لغات البرمجة ذات المستوى المنخفض (Low Level Languages).**

سميت بهذا لأنها بعيدة عن لغة الإنسان، وتحتاج إلى مترجمات خاصة لتحويل هذه اللغات إلى لغة الآلة.

- **لغات البرمجة ذات المستوى العالي High Level Language**

سميت باللغات ذات المستوى العالي نظراً لقربها من لغة الإنسان، وقد صممت للتغلب على بعض المساوئ والصعوبات، التي صاحبت استخدام لغات البرمجة ذات المستوى المنخفض.

أجيال لغات البرمجة:

تنقسم اللغات، من حيث مراحل تطورها ومدى اقترابها من لغة الحاسوب، إلى ما يعرف بالأجيال، فتعتبر لغة بسيطة أو برامج منخفضة المستوى كلما اقتربت من شكل لغة الآلة أو الحاسوب، التى يعبر عنها "بالأرقام الثنائية"،[0,1]، التي يستخدمها الحاسوب. وتكون الأجيال أو اللغات عالية المستوى كلما اقتربت من اللغة التي يستخدمها الإنسان "اللغة الطبيعية". وتبعاً لذلك تتدرج اللغات من حيث أجيالها أو مستواها، كما يلى:

- جيل لغة الآلة.
- جيل لغة التجميع.
- جيل اللغات عالية المستوى الإجرائية.
- جيل اللغات عالية المستوى الغير إجرائية.
- اللغات الطبيعية.

1- الجيل الأول (لغة الآلة):

هى مجموعة من الرموز التي يتم بموجبها كتابة أوامر في صفحة الآلة القابلة للتنفيذ دون الحاجة إلى ترجمة. وهى لغة يصعب التعامل معها وكتابتها. مثل الأمر التالى:

1101101000 1100110010 11100110 11011101 1001001

2- الجيل الثاني (لغة التجمع):

استخدمت لغة التجميع لتسهيل البرمجة، فهى لغة الحروف المجمعة. وهى لغة تختصر بعض العبارات والرموز المستخدمة. ففيها يتم استبدال الرموز الرقمية في لغة الآلة بمجموعة من الكلمات الرمزية "المختصرة " باستخدام اللغة الإنجليزية. إذ من السهل نوعا التعامل معها، مثل:

L *for* Load, A *for* Add, B *for* Brave, *and* C *for* comp

3- الجيل الثالث (اللغات الإجرائية عالية المستوى):

هى لغة سهلة مثل اللغة التي يتعامل معها الإنسان بشكل يومي، وهى مثل قراءة الكتاب وكتابة المعادلات الرياضية. وهى مصممة للكتابة على أعداد وأنواع مختلفة من الحواسب دون الحاجة إلى تغيير أو بتغيير بسيط.

سميت باللغة الإجرائية لأنها تستخدم القواعد والخطوات في كتابتها.

ومن أشهر هذه اللغات:

- الكوبول، COBOL
- الفورتران، FORTRAN
- البيسك، BASIC
- الباسكال، PASCAL
- السى، والسى بلس، والسى بلس بلس، C, C+, C++

الأسباب التي دعت إلى استخدام اللغات عالية المستوى:

- تحرر المبرمج من التعقيدات، والخطوات والإجراءات الطويلة في كتابة البرنامج بلغة الآلة ولغة التجميع.

تقدم للمستخدم لغة يمكن استخدامها في أكثر من حاسوب مع تعديلات طفيفة.

- تمنح المبرمج فرصة ووقت أكبر للتركيز على احتياجات المستخدم، وبالتالي يصمم برنامج يتوافق مع هذه الاحتياجات.
- سـمیت باللغة الإجرائية لأن تعليمات البرمجة تتألف من مجموعة من الخطوات أو الإجراءات التي تُعْلم أو تخبر الحاسـوب، ليس فقط ماذا يفعل بالمعطيات، بل كيف يفعل بها.

مميزات و عيوب اللغة الإجرائية:

المميزات:

- سهولة تذكر الأوامر وقواعد اللغة.
- لها القدرة على وضع البيانات والأوامر في ذاكرة الحاسوب نيابة عن المبرمج. أى تخصيص الأماكن بالذاكرة الرئيسية للحاسب.
- غير معتمدة على نوع الجهاز المستخدم.
- سهولة تتبع البرامج لتعديلها، أو رصد الأخطاء وتصحيحها

العيوب:

- تحتاج إلى مترجم لتحويلها إلى لغة الآلة.

لماذا نحتاج إلى مترجم:

- دائما نحتاج إلى مترجم للغة، لأن الحاسوب يستطيع أن يفهم ويعمل بلغة الآلة فقط، والمستخدم يريد مترجم للغات أخرى.
- مترجم اللغة نحتاجه لترجمة أو تحويل اللغة عالية المستوى "لغة المصدر" إلى لغة الآلة "لغة الهدف" حتى يعمل البرنامج على جهاز الحاسوب.
- كمستخدم تحتاج إلى استخدام معالج اللغة عندما تقوم ببرمجة برنامج باللغة عالية المستوى.

المبرمجون يستخدمون نوعين من معالجات اللغة:

- مؤلف (Compiler): المجمع أو المفسر أو المحول
- مترجم (Interpreter): المصنف أو المؤلف

المترجم، المؤلف، المُجمع (Compiler):

- هو برنامج يترجم لغة التجميع إلى لغة الآلة. هو برنامج معد خصـــيصـــا للقيام بعملية الترجمة والتصنيف والتأليف لبرنامج آخر في لغة المصدر، وتحويله إلى الصيغة القابلة للتنفيذ بلغة الآلة.
- هو برنامج يقوم بعمليات المعالجة التحويلية التي تترجم كامل البرنامج المكتوب باللغة عالية المستوى إلى لغة الآلة بخطوة واحدة.

المفسر (Interpreter):

- هو البرنامج الذي يقوم بفك الشـفرة الخاصـة بالبيانات، والحصـول على المعنى لهذه الصيغة، لاستخدامها في الغرض الذي أعدت من أجله، وهو يقوم بالترجمة إلى لغة الآلة جملة تلو الأخرى أثناء التنفيذ.

شكل (1-10)

4- الجيل الرابع جيل اللغات عالية المستوى (الغير إجرائية):

- الجيل الرابع، سـهل الاسـتخدام، أكثر سـهولة من الجيل الثالث وهو لغة غير خطواتية هى لا تتطلب خطوات لإجرائها.
- المبرمج يخبر الحاسـوب النتيجة المطلوبة تحقيقها بدلا من كيف يمكنه تحقيقها. ولسهولتها أصبح للمبرمج القدرة على تطوير البرامج.

أنواعها:

- لغات الجداول الإلكترونية.
- قواعد البيانـــات.
- منتج التطبيقـات.
- منتج التقـــارير.
- اللغة الاستعلامية

الجيل الخامس (اللغات الطبيعية):

- جيل اللغات الطبيعية مشابه للغات الاستعلامية، ولكن لا يحتاج المبرمج بها إلى معرفة كلمات (مصطلحات) معينة أو قواعد أو عبارات.
- هذا الجيل يقترب من الحواسب الخبيرة، وهى مجموعة من المعارف جمعت من عدد من الخبراء في مجال معين ومحدد لحل مشكلة محددة.

لغة الهدف القيم أو المتألق (Object-oriented language)

- عندما يشرع أغلب المبرمجون البدء في مهمة البرمجة فإنهم يفكرون في جزأين مهمين

- رموز البيانات التي يجب أن تعالج.
- خطوات معالجة تلك البيانات.

هاتان الخطوتان كانتا مهمتان ومنفصـــلتان في الماضـــي، ولكن الآن باســـتخدام اللغة الطبيعية يقوم أغلب المبرمجون بدمجهما في عنصر واحد يسمى الهدف (Object).

4-1) طريقة حل المشاكل بواسطة الحاسب:

المشاكل المقصودة هنا هى المسائل البرمجية، بمعنى أكثر تفصيلاً هى كيفية صياغة المشكلة، وترتيبها بطريقة منظمة، حتى يمكننا إدخالها للحاسوب بطريقة منطقية.

خطوات حل المشاكل:

تتلخص خطوات حل المشاكل في ست خطوات نوضحها فيما يلي:

أولاً: تحليل عناصر المشاكل:

المقصود بتحليل عناصر المشكلة هو أن نعرف ماذا نريد بالضبط من البرنامج.

ولكي نقوم بعملية تحليل عناصر المشكلة يجب علينا تحديد العناصر الأساسية للمشكلة وهى:

أ- **تحديد مخرجات** البرنامج، ويقصد بها النتائج والمعلومات.

ب- **تحديد مدخلات** البرنامج، ويقصد بها المدخلات أو البيانات (المعطيات).

ج- **تحديد عمليات** المعالجة ويقصـــد بها العمليات الحســـابية والخطوات المنطقية.

مثال:

نفرض أننا نريد حساب مساحة المستطيل بمعلومية الطول والعرض.

قم بتحليل عناصر المشكلة إذا علمت أن:

مساحة المستطيل = الطول × العرض.

الحل:

لتحليل عناصر المسألة نقوم بما يلي:

- **تحديد المخرجات**: وهى هنا مساحة المستطيل (Area)
- **تحديد المدخلات:** الطول (X) والعرض (Y)
- **تحديد عملية المعالجة**: وهى قانون مساحة المستطيل:

Area=(X * Y)

ثانيا- كتابة الخطوات الخوارزمية

ويمكن تعريف الخوارزميات بأنها مجموعة من القواعد والعمليات المعرفة جيداً لحل المشكلة في عدد محدد من الخطوات.

الطرق الأساسية لكتابة الخوارزم:

أ- أن تكون كل خطوة معرفة جيداً.

ب-أن تتوقف العمليات بعد عدد محدد من الخطوات.

ج- أن تؤدي العمليات إلى الحل الصحيح للمسألة.

مثال:

أكتب الخطوات الخوارزمية لحساب مساحة المستطيل بمعلومية الطول والعرض، إذا علمت أن مساحة المستطيل = الطول × العرض.

الحل:

1) أدخل الطول (X)، والعرض (Y).
2) احسب مساحة المستطيل (Area=X * Y).
3) أطبع المساحة (Area).
4) النهاية.

ثالثاً: رسم مخططات الانسياب (خرائط التدفق):

وهى عبارة عن تمثيل بياني، أو رسم للخطوات الخوارزمية.

فوائد مخططات الانسياب:

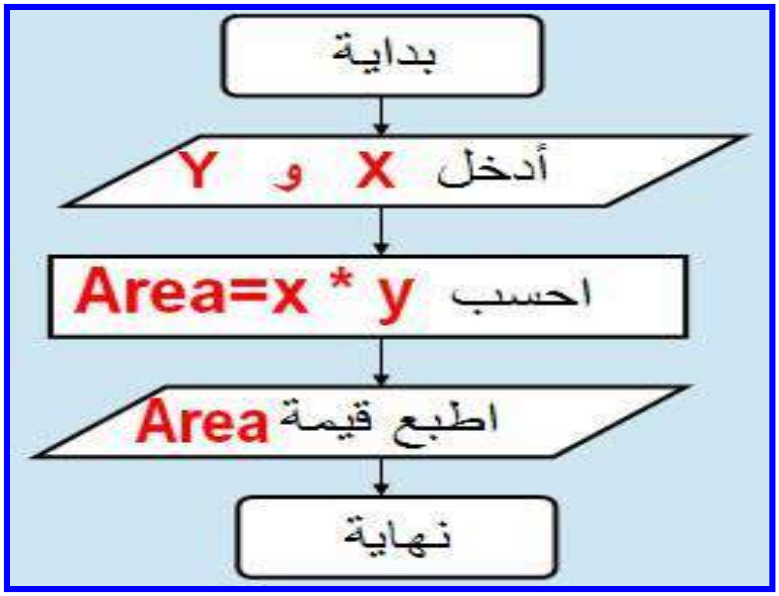
أ) توضيح طريقة سير البرنامج.

ب)توثيق منطق البرنامج للرجوع إليه عند الحاجة.

الرموز والأشكال الهندسية المستخدمة في رسم مخططات الانسياب		
الشكل	الاسم	المعنى
	بداية / نهاية	يمثل بداية أو نهاية البرنامج
	إدخال / إخراج	يمثل إدخال البيانات أثناء البرنامج أو اخراجها
	عملية	يمثل عملية معالجة البيانات
	قرار	يمثل اتخاذ قرار أو تعبير منطقي
	خط انسياب	يمثل اتجاه الانسياب المنطقي للبرنامج

شكل (1-11)

ويمكننا رسم مخطط الانسياب للخطوات الخوارزمية للمثال السابق (مساحة المستطيل)، كما يلي:



شكل (1-12)

رابعاً: كتابة البرنامج باستخدام إحدى لغات البرمجة

خامساً: ترجمة البرنامج إلى لغة الآلة

سادساً: اختبار البرنامج وإصلاح الأخطاء

وسوف نتحدث بالتفصيل عن بقية الخطوات (رابعا وخامسا وسادسا) في الفصول القادمة من خلال دراستنا للغة C++.

أسئـــــــــلة البـاب الأول

1-ماهى مكونات نظام الحاسوب مع رسم شكل توضيحي ؟

2-عرف الآتي:

(أ) البيانات "Data"

(ب) عملية المعالجة "processing"

(حـ) الشبكات "Networks"

(د) عملية تخزين البيانات "Storage"

3- عرف، مع ذكر أمثلة:

a) Software

b) Hardware

4- ما المقصود بنظام التشغيل؟ اذكر أمثلة

5-كيف يمكن تصنيف لغات البرمجة؟

7-أذكر الأجيال المختلفة للغات البرمجة.

8- بين كيف يمكن حل المشاكل بواسطة الحاسوب؟ مع ذكر الخطوات اللازمة بالتفصيل.

9- ما المقصود بمخطط الانسياب "FLOWCHART"؟ وماهى استخداماته؟

10-ما المقصود بالخوارزميات؟ اذكر الطرق الأساسية لكتابة الخوارزم

==========

الباب الثانى

البرمجة باستخدام لغة C++.

2-1 هيكل البرنامج.

2-2 الثوابت والمتغيرات.

2-3 معاملات (C++ operators).

2-4 جمل الإدخال والإخراج.

2-5 تطبيقات على جمل الإدخال والإخراج.

2-6 الدوال الأساسية.

2-7 تطبيقات.

==========

الباب الثانى

البرمجة باستخدام لغة C++:

2-1) هيكل البرنامج:

تعتبر لغة C++ من اللغات القوية فى البرمجة لحل المشكلات وبناء المشروعات ويستخدمها المتخصصون فى بناء برامج قوية.

مميزات لغة C ++:

1- لغة عامة:

أى أنها تصلح لعمل برامج قواعد البيانات والرسومات والحسابات ونظم التشغيل.

2- لغة تركيبية:

يتألف البرنامج المكتوب بلغة C++ من دالة رئيسية وبداخلها مجموعة من الدوال الإجرائية، وكل دالة من هذه الدوال عبارة عن مجموعة من الأوامر.

3- لغة متنقلة:

يمكن للبرنامج المكتوب بهذه اللغة أن يعمل مع أكثر من جهاز وأنظمة تشغيل مختلفة.

4- لغة قياسية:

معظم مترجمات اللغة تتوافق مع مترجمات اللغات القياسية الأخرى.

5- لغة سريعة:

لأن أدوات اللغة تتعامل مع الآلة مباشرة مما يختصر وقت التنفيذ.

==========

6- تتعامل على مستوى البت Bit:

حيث تستطيع أن تقرأ وتكتب وتغير وتقوم بكل العمليات التي على مستوى البت Bit، وكما نعرف فأن البت هو اصغر وحدة لقياس المعلومات داخل الكمبيوتر وأصغر وحدة تخزين في الذاكرة، وهو جزء من ثمانية أجزاء من البايت Byte.

* وحتى يمكننا استخدام لغة C++ فى بناء برامج بسيطة ومعقدة لابد من التعرض لمجموعة عناصر هامة هى:

1- دراسة المفاهيم الأساسية للغة C++.

2- التعرف والتعامل مع واجهة التطبيق ومحرر ومترجم لغة C++.

3-كيفية بناء هيكل البرنامج.

4- ترجمة البرنامج وتصحيح أخطائه واستخدام التعليقات.

5- التعرف والتعامل مع أوامر اللغة (مفاهيم إدخال وإخراج البيانات)

6- تصنيف أوامر اللغة.

وكتابة البرنامج تبعاً للمشكلة أو المهمة المطلوب حلها أو تنفيذها.

الشكل العام للبرنامج المكتوب بلغة C++:

الصيغة الرئيسية لبرنامج مكتوب بلغة C++ يجب أن يتبع الشروط التالية:-

```
#include<library.h>
main()
{
.....;
.....;
return ;
}
```

==========

1- يبدأ البرنامج بالعبارة #include< > والفراغ الموجود داخل العلامتين < > لتحديد نوع المكتبة المستخدمة في البرنامج، مثل:-

#include<iostream.h> -- #include<vector.h> -- #include<conio.h>

2- يتكون البرنامج من دالة رئيسية main() ثم يبدأ البرنامج بقوس البداية { وينتهى بقوس النهاية }.

3- جميع الدوال والكائنات تكتب بالحروف الصغيرة مثل main، cin، cout، وغيرها ولا يجوز كتابة الحروف الكبيرة فالمثال التالي خطأ MAIN أو COUT وهكذا.

4- تنتهى كل عبارة بفاصلة منقوطة.

5- جسم البرنامج كله يكون محصورا بين القوسين { سطور البرنامج }.

عناصر بناء لغة C++:

1. الأوامر Commands والتعبيرات Expressions والعبارات Statements.
2. البيانات (ثوابت ومتغيرات) Data.
3. موجهات ما قبل المعالجة preprocessor directives
4. الدوال Functions.
5. المؤشرات والمصفوفات Arrays.
6. التعليقات Comments.

رموز لغة C++:

الرموز المستخدمة فى لغة C++ هى:

أ- الحروف الإنجليزية الكبيرة A,B,C,...

ب-الحروف الإنجليزية الصغيرة a,b,c,...

ج- الأرقام العربية 1,2,3,...

==========

د-الرموز الخاصة (.....,”,;;,=!,[],!,||,(),%,#,$,<,>,//,/,-,+) وتعد هذه الرموز بأنواعها المادة الخام التى تتكون منها مفردات لغة C++.

الأدوات المستخدمة فى لغة C++:

أ- الأدوات(المعاملات) الحسابية Arithmetic Operators.

ب-الأدوات العلاقية والمنطقية Relational and Logical.

ج- الأدوات الدقيقة Bowties Operators.

د – أدوات أخرى(الشرطية – العنونة).

2-2) الثوابت والمتغيرات Constants & variables:

أولا الثوابت Constants:

وهى عبارة عن قيم ثابتة يراد الاحتفاظ بها طوال البرنامج لا تتغير قيمتها أبدا0

وتنقسم الثوابت في لغة C++ إلى:-

1- ثوابت عددية Numeric Constants

2- ثوابت رمزية Non-numeric Constants

1- الثوابت العددية:

يمكن تمثيل الثوابت العددية في لغة C++ كالآتي:-

(أ)-الثابت العددي الصحيح integer :

- هو عبارة عن عدد مكون من الأرقام من (0 إلى 9).
- لا يحتوي على فاصلة عشرية.
- يمكن أن يحوى الإشارة (+ أو -).

ومن أمثلة الثابت العددي الصحيح (0، 12، 1000، 20-،.....)

==========

كما يمكن تصنيف الأعداد الصحيحة في لغة C++ حسب طولها والسعة التخزينية لها في الذاكرة مثلا:-

- الثوابت الصحيحة (19679، 40000) تسمى ثوابت صحيحة طويلة long int
- الثوابت (16-، 09 ، 55) تسمى ثوابت صحيحة قصيرة short int
- الثوابت (20000 ، 967) تسمى ثوابت صحيحة بدون إشارة unsigned int

والفرق بين الثوابت الطويلة والقصيرة هو في عدد الوحدات التخزينية المطلوبة لكل نوع في الذاكرة، فالطويلة تأخذ حيزا أكبر، والقصيرة توفر عدد الوحدات التخزينية المستعملة، أما الثوابت الصحيحة بدون إشارة فإن استعمالها يوفر وحدة تخزينية تستعمل للإشارة.

(ب)-الثابت العددي الحقيقي Floating Constant

- هو عدد مكون من الأرقام من (0 إلى 9)
- يجب أن يحتوي على فاصلة عشرية
- يمكن أن يحوي الإشارة (+، -)

ومن أمثلة الثوابت العددية الحقيقية

(421.5، 10.55، 67.99-، 000000)

ويمكن الإعلان عن الثوابت العددية كالتالي:

```
Const int number=20;
```

2- الثوابت الرمزية Non-Numeric:

وهى عبارة عن رموز اللغة وتتكون من الحروف والأرقام وتكون بين علامتى تنصيص أو اقتباس.

ومن الأمثلة على الثوابت الرمزية ما يلي:-

("name" - "Khaled" – "12345" – "30+40 ")

==========

ونلاحظ أن هناك ثوابت رمزية تحتوي على أرقام وهى في الحقيقة قيم حسابية لا يمكن إجراء أى عملية حسابية عليها بل يتم طبع الأرقام أو الرموز كما هى.

==========

ثانيا المتغيرات Variables:

تعريف المتغير:

هو رمز أو اسم فريد يعطى لتخزين نوع من أنواع البيانات المعرفة مسبقاً داخل برنامج C++ أو التى تقوم أنت بإنشائها.
وهو عبارة عن أسماء تحجز مواقع فى الذاكرة حتى يتمكن البرنامج من تخزين البيانات فيها.

وتنقسم المتغيرات بدورها إلى نوعين:-

1- متغيرات رمزية (حرفية).

2- متغيرات عددية .

1) المتغيرات الحرفية Char:

وتتضمن الحروف بكافة أشكالها والرموز والفراغات (مسافة فارغة) مثل:

```
Char a,b;
A= “a”;    Char var1;
B=””;      Var1=” “;
```

2) المتغيرات العددية Numeric Variables:

أ) المتغيرات العددية الصحيحة Integer:

تتضمن قيم عددية صحيحة يمكن أن تأخذ قيمة تصل إلى 32767 وتكتب على الشكل التالى:

```
Int a;
A=100;
```

==========

ب) المتغيرات العددية الحقيقية Floating Point:

تتضمن جميع الأعداد الحقيقية وتكتب على الشكل التالى:

```
Float x;
X=5.2;
```

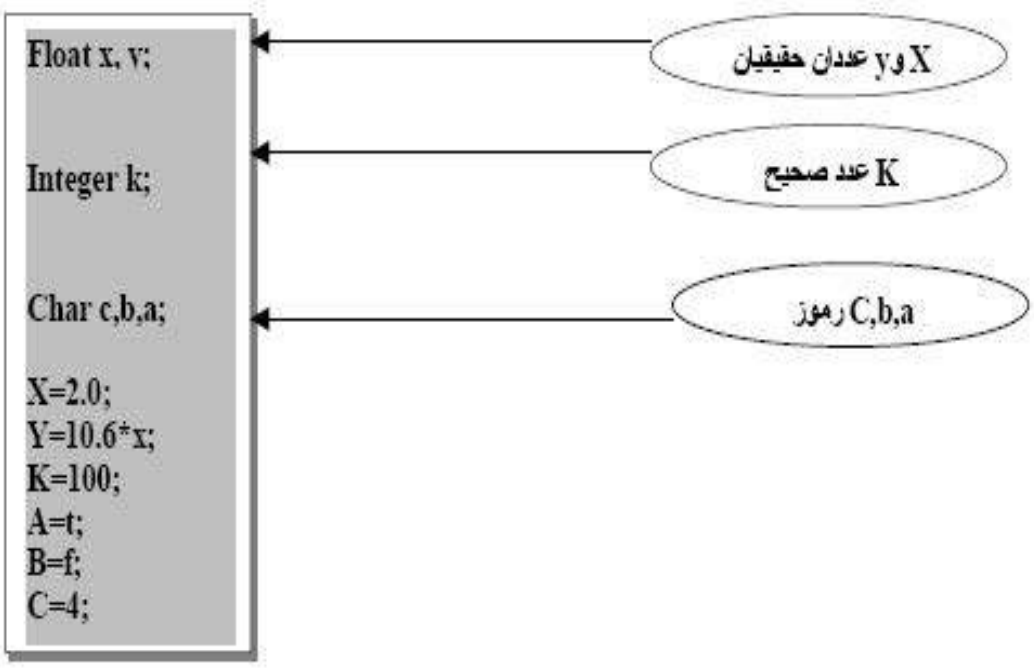
ج) المتغيرات العددية الحقيقية الطويلة Double:

هى نفس المتغيرات العددية الحقيقية ولكن يمكن تمثيلها فى خمسة عشرة خانة وتكتب على الشكل التإلى:

```
Double x;
```

ويجب الإعلان عن المتغيرات في بداية البرنامج

أمثلة للمتغيرات:-



شكل (2-1)

ونلاحظ في السطر الأول الإعلان عن متغيران x , y من النوع الحقيقي float ونلاحظ أنه تم الفصل بين المتغيرين بفاصلة.

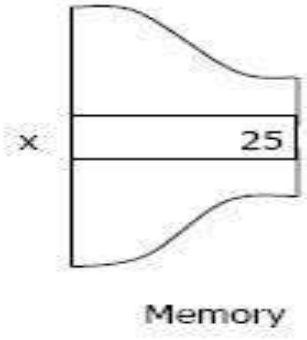
==========

وفي السطر الثاني تم الإعلان عن متغير واحد هو k من النوع الصحيح int

أما في السطر الثالث فلقد تم الإعلان عن ثلاث متغيرات من نوع الرموز char

ولفهم كيفية تخزين البيانات في الذاكرة نجري المثال التالي:

** إذا أردنا مثلا وضع القيمة 25 في الذاكرة في المخزن X أى X=25 أى أن نجعل المخزن المسمى X في الذاكرة يحتوي على القيمة 25 ويمكن فهم ذلك من الرسم التالي:



شكل (2-2)

الفرق بين الثابت والمتغير:

المتغير قابل للعنونة لأنه قابل للتغيير فلابد من معرفة عنوانه - أما الثابت لا يتم تغييره لذا لا تحتاج إلى معرفة عنوانه رغم أن العنوان موجود.

تسمية المتغير:

يمكن تسمية المتغير باستخدام خليط من الحروف والأرقام والشرطة التحتية على أن يبدأ اسم المتغير بحرف أو شرطة تحتية ولا يمكن أن يبدأ برقم كما يجب ألا يسمى المتغير بأى كلمة من الكلمات المحجوزة داخل لغة C++ وبيانها كما بالجدول التالى ولا يشترط طول معين لاسم المتغير.

==========

قائمة بالكلمات المحجوزة:

Return	delete	Asm	If
inline	Auto	Try	Do
Double	Typedef	Short	Break
Case	Signed	Int	Union
Unsigned	Long	Else	Sizeot
Static	Enum	Catch	New
Operator	Char	Virtual	Extern
Float	Void	Struct	Class
Const	Switch	Private	Volatile
While	Protected	For	Template
This	Friend	Continue	Public
throw	goto	default	Register

سمات أسماء المتغيرات:

1- يكتب اسم المتغير بحروف صغيرة Small Letters.

2- يجب أن يكون اسم المتغير معبراً عن الوظيفة التى يستخدم من أجلها مثل المتغير Salary فى برنامج المرتبات.

3- عند استخدام اسم متغير مكون من كلمتين نضع بينهما شرطة تحتية أو اجعل الحرف الأول من كل كلمة تابعة حرف كبير مثل is empty أو IS Empty.

==========

3-2) المعاملات ـ والمعاملات العلائقية:

تستخدم المعاملات فى العمليات الحسابية والمنطقية وهى كالتإلى:

المعامل	الوظيفة	العمليات المسموح بها	نوع البيانات المستخدمة فيها
+	الجمع	أدوات حسابية ومنطقية	عددية ومنطقية
-	الطرح	أدوات حسابية	عددية
/	القسمة	أدوات حسابية ومنطقية	عددية
%	باقى القسم	أدوات حسابية	عددية
*	الضرب	أدوات حسابية	عددية
++	الزيادة بمقدار واحد	أدوات حسابية	عددية
--	النقصأن بمقدار واحد	أدوات حسابية	عددية
=	التخصيص	أدوات حسابية ومنطقية	عددية ومنطقية
>	الأكبر (اكبر من)	أدوات علاقية	عددية ومنطقية
<	الأصغر (أصغر من)	أدوات علاقية	عددية ومنطقية
<=	أصغر من او يساوى	أدوات علاقية	عددية ومنطقية
==	أكبر من أو يساوى	أدوات علاقية	عددية ومنطقية
!=	إذا كان لا يساوى	أدوات علاقية	عددية ومنطقية
&&	and	أدوات منطقية	منطقية
!!	or	أدوات منطقية	منطقية
!	Not	أدوات منطقية	منطقية
~	Not	ادوات دقيقة	تتعامل مع بت bit
>>	إزاحة إلى اليسار		وتستعمل هذه الأدوات مع المخرجات والمدخلات
<<	إزاحة إلى اليمين		المعطيات int ،char
^	xor		
&	And		تستخدم مع البيانات المنطقية
!	or		ولا تستخدم مع غيرها

أولويات العمليات الحسابية Arithmetic Operations :

في السي بلس بلس توجد خمس عمليات حسابية:

1- عملية الجمع: (+)

==========

2- عملية الطرح: (-)
3- عملية الضرب: (*)
4- عملية القسمة: (/)
5- عملية باقي القسمة(%)

جميع هذه العمليات الحسابية بإمكانك القيام بها على المتغيرات العددية، وسيأتي الوقت الذي تصل فيه إلى تطبيقها ، بالنسبة إلى العملية الخامسة فلا يمكنك القيام بها إلا على أعداد من النوع int وليس غيره.

1- الزيادة والنقصان عندما تأتى قبل العدد - - ,++.
2- الأقواس ( ) .
3- إشارة السالب - .
4- القسمة وباقى القسمة والضرب /, % , *.
5- الجمع والطرح + , - .
6- التساوى (المساواة) = .
7- الزيادة والنقصان المتأخرة بعد العدد ++ , -- .

ملاحظة هامة:

قسمة عدد صحيح على عدد صحيح يكون الناتج صحيح	قسمة عدد حقيقى على عدد حقيقى يكون الناتج حقيقى
قسمة عدد حقيقى على عدد صحيح يكون الناتج حقيقى	لا توجدعملية قسمة عدد صحيح على عدد حقيقى

عمليات المقارنة أو العلائقية Relation Operator

في السي بلس بلس توجد عمليات المقارنة حيث بإمكانك مقارنة أعداد مع بعضها البعض أو مقارنة أحرف من النوع char وهذه هي عمليات المقارنة في++C == >= > <= <

معاملات الإزادة والإنقاص Increment and Decrement operators

هما من العمليات الغريبة علينا:

++ معامل الإزادة - - معامل الإنقاص

سنتعرف الآن على عملية غريبة علينا وهذه العمليتين هي عملية الإزادة ++وعملية الإنقاص--

ليس ذلك فحسب بل طريقة كتابة هذه العمليتين قد تختلف ، وهي صيغتين إما أن تكون إحدى هذه العمليتين على يمين المتغير وإما على يساره وتختلف في كلا الحالتين ، حتى تفهم ما أعنيه لنفرض أن لدي متغيران الأول هو a والثاني هو b انظر إلى هذه الأسطر

```
a = ++b ;
```

إن هذا السطر يخبر المترجم بالقول يا أيها المترجم زد قيمة المتغير b رقماً واحداً أي العدد بالعدد 1 ثم أسند قيمة المتغير b إلى المتغيرa فلو افترضنا أن قيمة المتغير b هي 6 ، فحينما يقوم البرنامج بتنفيذ السطر السابق فإنه يقوم أولاً بزيادة المتغير b زيادة واحدة أي تصبح قيمته 7 ثم يسند القيمة إلى المتغير a أي ستصبح قيمة المتغير a أيضاً 7 ؛ الآن لو افترضنا أننا قمنا بكتابة صيغة أخرى وهي هكذا:

==========

```
a = b ++ ;
```

ستختلف العملية هنا ، والآن قم بالتركيز فيما سيكتب ، أولاً سيأخذ المترجم قيمة المتغير b بدون أي تغيير ويقوم بإسنادها إلى المتغير a ثم بعد ذلك يقوم بزيادة المتغير b زيادة واحدة أي أن هذه الصيغة عكس الصيغة السابقة فلو فرضنا أن قيمة المتغير b هي 6 فأولاً سيأخذ المتغير هذه القيمة ويسندها إلى المتغير a وبالتالي تصبح قيمة المتغير a هي 6 ثم بعد ذلك يقوم المترجم بزيادة المتغير b أي أنها ستصبح 7 .
نتمنى أن تكون الصيغتان مفهومتان ، أيضاً نفس الشرح السابق يطبق على عملية الإنقاص --، مع اختلاف العمل الذي تقومان به طبعاً.

4-2) جمل الإدخال والإخراج:

لا تحتوى لغة C++ على وظائف الإدخال والإخراج ولكن يتم تخزين هذه الوظائف داخل مكتبة باسم iostream الموجودة بالملف الرئيسى iostream.h وتحتوى هذه المكتبة على مجموعة من العمليات المستخدمة لتنفيذ القراءة أو الكتابة إلى أنواع البيانات المعرفة داخل اللغة. ويتم إجراء عمليات الإدخال والإخراج من خلال ثلاث تصنيفات هى:

- **تصنيف الإدخال istream**
- **تصنيف الإخراج ostream**
- **تصنيف الإدخال والإخراج iostream**

ويتم إخراج البيانات باستخدام معامل الإزاحة الأيسر >> ويسمى معامل الإخراج.

أما فى حالة إدخال البيانات يستخدم معامل الإزاحة الأيمن<< ويسمى معامل الإدخال.

والجدول التالى يوضح معاملات الإدخال والإخراج فى لغة C++.

المعامل	الاستخدام	يتعامل مع
cin	معامل الإدخال الرئيسى	لوحة المفاتيح
Cout	معامل الإخراج الرئيسى	الشاشة
Cerr	معامل إخراج الخطأ	الشاشة
clog	معامل إخراج الخطأ	الشاشة

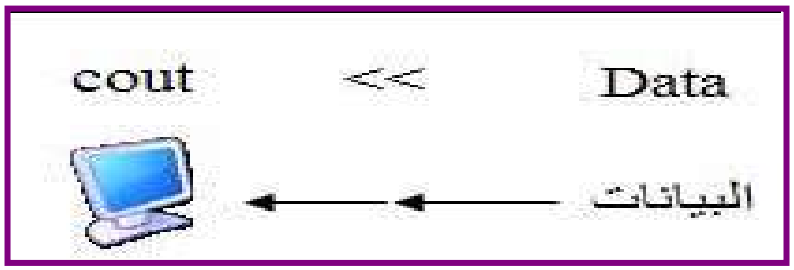
تستخدم هذه المعاملات فى البرنامج بشرط أن يحتوى البرنامج على الملف الرئيسى **iostream.h**

==========

جمل الإدخال والإخراج:

أولا جملة الإخراج:

يمكنك أن تطلب من البرنامج طبع أى قيمة على الشاشة بالكائن **cout** ويمكنك طباعة أى قيمة من خلال معامل الإخراج >>.



شكل (2-3)

وبإمكان البرنامج طباعة المتغيرات أو الجمل التى تريد أنت إظهارها ولكى تظهر جملة على الشاشة يجب كتابتها بين علامتى تنصيص " " كما هو موضح بالمثال التالى:

```
cout <<"Hello C++";
```

أما إذا أردت إظهار قيم أحد المتغيرات فعليك كتابة اسمه دون علامتى تنصيص كالتالى:

```
cout << a;
```

حيث الرمز a عبارة عن المتغير

وبإمكانك طباعة أكثر من متغير أو جملة دفعة واحدة كما هو مبين فى المثال التالى:

```
cout <<"Hellow" << endl <<"World";
```

سيكون خرج البرنامج على الشاشة كالتالى:

```
Hellow
world
```

==========

ونلاحظ هنا أن كلمة (endl) قد تسببت في طباعة الكلمتين في سطرين متتالين وهى اختصار كلمة end line ويمكن الاستعاضة عن هذه الكلمة بوضع الرمز n كالاتي:

```
Cout<<"hellow\n"<<"world;
```

```
Hellow
World
```

أما إذا استبدلنا الحرف (n) بالحرف (t) فإنه يتم ترك ثماني مسافات بين الكلمتين كالتالي:

```
Cout<<"hellow\t"<<"world";
```

فإنه يتم الطباعة على الشاشة كالتالي

```
Hellow    wold
```

لاحظ المسافة بين الكلمتين

والجدول التالى يبين بعض الخصائص للكائن cout وتسمى سلاسل الإفلات:

الخاصية	معناها
\t	جدولة افقية تترك ثماني فراغات
\n	الانتقال إلى سطر جديد
\r	اعادة المؤشر إلى بداية السطر
\a	يقوم بإصدار صوت تنبيه
\b	الحذف الخلفى (backspace)

وتكتب هذه السلاسل ضمن الجمل أى بين علامتى تنصيص " "

فمثلا لإيجاد حاصل ضرب العددين 5*7 يمكن إجراء عملية حسابية بسيطة كالتالي:

```
Cout<<5*7
```

فيتم طباعة الناتج 35

==========

ثانيا جملة الإدخال cin:

بالنسبة للإدخال فى لغة C++ فبواسطة الكائن cin , وهذا الكائن يستخدم فقط مع المتغيرات.

وعملية الإدخال هى عكس عملية الإخراج حيث أننا نستخدم معامل الإدخال <<cin.

شكل (2-4)

شكل تخيلي..يبين كيفية إدخال البيانات من المستخدم وإخراجها إلى الشاشة

فمثلا لإدخال عدد صحيح من خلال لوحة المفاتيح

Cin>>x;

هنا ينتظر منك البرنامج إدخال قيمة عددية واحدة وحفظها في المتغير x

أما إذا أردنا إدخال أكثر من قيمة فيمكن أن يكون المثال كالتالي

Cin>>x>>y>>z;

وهنا يمكن إدخال ثلاث قيم

التعليقات:

عند كتابة برنامج بأى لغة برمجة وعندما يصبح البرنامج كبير للغاية فيستحب استخدام التعليقات , ولا تستخدم التعليقات فى جميع سطور البرنامج بل فقط فى المواضع التى تعتقد أن هناك صعوبة فى فهمها ممن سيقرأ البرنامج غيرك أو حينما تأتى أنت بعد مضى مدة طويلة لتقرأ تلك الأكواد والمبرمج الذكى

==========

الذى يحرص على كتابة ما يمكنه من التعليقات على برنامجه ليسهل عليه تصحيحه أو استخدام بعض أجزائه عند الحاجة.

وتسمح لغة C++ بكتابة التعليقات بطريقتين تسهلان على المبرمج وضع ما يشاء من التعليقات على البرنامج.

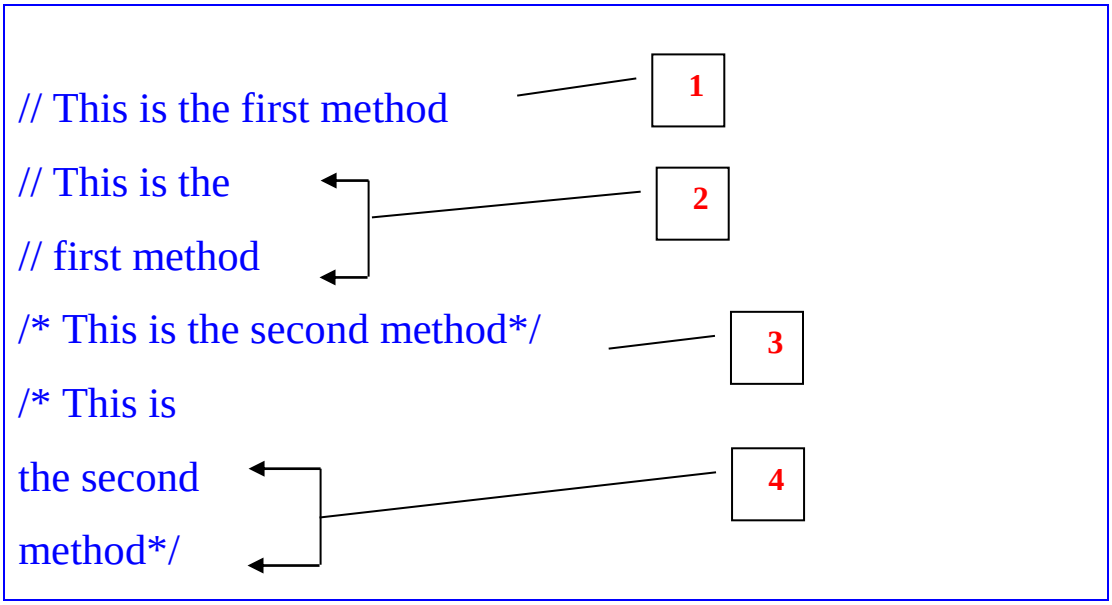
الطريقة الأولى: هى كتابة التعليق بعد العلامة (//) حيث يتجاهل المترجم السطر الذي يلي هذه العلامة

```
Int a=0 // this is a;
```

ولكن لو تجاوز التعليق السطر لزم إضافة المزيد من الرموز (//) أمام كل سطر من التعليقات

الطريقة الثانية: وهى كتابة التعليق بين العلامتين (/*) بداية التعليق و(*/) نهاية التعليق مهما كان عدد أسطر التعليق فلن يلتفت المترجم إلى هذه السطور.

والمثال التالي يوضح كيفية استخدام الطريقتين:



في المرة الأولى استخدمنا الطريقة الأولى ولم يتجاوز التعليق السطر فلم نستخدم سوى علامة تعليق واحدة0 أما في المرة الثانية تجاوز التعليق السطر فلزم علينا استخدام علامة تعليق ثانية.

==========

وفي المرة الأخيرة استخدمنا الطريقة الثانية لكتابة التعليقات ومع أن التعليق تجاوز السطر فلم نستخدم علامة تعليق جديدة لكل سطر بل اكتفينا بوجود العلامة (/*) في بداية التعليق والعلامة (*/) في نهاية التعليق.

5-2) تطبيقات:

تطبيق (1):

المطلوب كتابة برنامج يقوم بطباعة قيمه عددية وليكن العدد 2009

```
1-//هذا هو البرنامج الأول
2- //وهو برنامج طباعة العدد 2009 على الشاشة
3- #include<iostream.h>
4- main()
5- {
6- cout<<"2009";
7- return 0;
8- }
```

فعند تنفيذ البرنامج يقوم البرنامج بطباعة العدد 2009

ونلاحظ في هذا المثال أننا قمنا بترقيم الأسطر وذلك لتبسيط شرح أسطر البرنامج ولا يتم الترقيم عند كتابة هذا البرنامج في بيئة Visual C++.

- السطر الأول والثاني أسطر ملاحظات لا يلتفت لها المترجم
- السطر الثالث #include<iostream.h> يتم فيه إخبار المترجم بأننا سوف نستخدم أوامر الإدخال cin او الإخراج cout
- السطر الرابع main() وتسمى الدالة الأساسية وكل برنامج يجب أن يحتوى على هذه الدالة ويمكن أن يتكون البرنامج من عدة دوال كما سندرس لاحقا.
- السطر الخامس { وهو عبارة عن قوس البداية للبرنامج.

==========

- السطر السادس cout<<2009 وهو أمر طباعة العدد 2009
- السطر السابع return 0 وفيها نخبر الكمبيوتر أن الدالة لا ترجع بقيم وسيتم دراسة ذلك لاحقا.
- السطر الثامن { قوس النهاية وبه ينتهى البرنامج.

تطبيق (2):

المطلوب كتابة برنامج لطباعة العبارة الآتية:

"This is my second program"

```
//هذا هو البرنامج الثاني
#include<iostream.h>
main()
{
cout<<"This is my second program";
return 0;
}
```

وعند التنفيذ تظهر الرسالة

```
This is my second program
```

تطبيق (3):

البرنامج التالي يبين عملية تنسيق الخرج على الشاشة

```
#include<iostream.h>
main()
{
cout<<"One";
```

==========

```
cout<<"Two\n";
cout<<"Three\n\n";
cout<<"Four";
return 0;
}
```

وعند تنفيذ البرنامج

```
OneTwo
Three
Four
```

تطبيق(4):

طباعة العدد100 والعدد 5.36

```
#include<iostream.h>
main()
{
int a=100;
float b=5.36;
cout<<a<<b;
return 0;
}
```

لاحظ الإعلان عن العدد الصحيح int a=100 والعدد الحقيقي float b=5.36

==========

تطبيق (5):

المطلوب كتابة برنامج لجمع عددين صحيحين

```
#include<iostream.h>
main()
{
int a,b,c;
cin>>a>>b;
c=a+b;
cout c;
return 0;
}
```

وعند تنفيذ هذا البرنامج يطلب منك البرنامج إدخال عددين ويضعهما في المخزنين a, b

ثم يقوم بجمع الرقمين ووضعهما في المخزن c

ثم يقوم بطباعة الرقم الموجود في المخزن c

ملاحظات على البرنامج:

يمكن إضافة بعض الرسائل التي تود أن تظهر عند تشغيل البرنامج ولذلك يمكن تطوير البرنامج السابق كالتالي:

```
#include<iostream.h>
main()
{
Int a,b,c;
cout<<"please enter two number\n";
cin>>a>>b;
c=a+b;
cout<<"the result is "<<c;
return 0;
}
```

==========

وعند تنفيذ البرنامج تظهر الرسالة الآتية للسؤال عن الرقمين المراد جمعهما فعليك إدخال أى رقمين بجوار رسالة السؤال وليكن العددين 25 , 4 ثم اضغط مفتاح الإدخال enter يظهر سطر النتيجة بجواره نتيجة الجمع وهو العدد 29 هكذا

Please enter two numbers25 4

the result is 29

بعد إدخال العدد الأول أما أن نترك مسافة وندخل العدد الثاني أو نضغط على مفتاح الإدخال وندخل الرقم الثاني.

تطبيق (6):

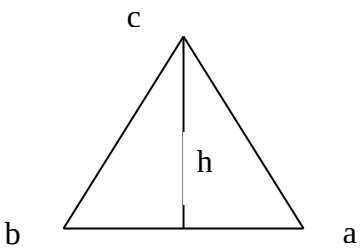
المطلوب حساب مساحة دائرة

```
#include<iostreame.h>
main()
{
int r;
float area,pi=3.14;
cout<<"please enter radius\n";
cin>>r;
area=r*r*pi;
cout<<"\t the area is:"<<area;
return 0;
}
```

==========

تطبيق (7):

المطلوب حساب مساحة مثلث




```
#include<iostream.h>
main()
{
int base,height;
float area;
cout<<"enter the base of triangle\n";
cin>>base;
cout<<"enter the height of triangle\n";
cin>>height;
cout<<"the area is: "<<0.5*base*height;
return 0;
}
```

==========

6-2) الدوال الأساسية

أنواع الدوال:

أ - دوال رئيسية.

ب - الدوال المبنية(المعرفة) سيتم دراستها في الفصول القادمة:

أ - الدوال الرئيسية:

هى دوال مستقلة بذاتها حيث لا تنتمى لأى فصيلة مثل الدالة ( )main ولها اسمها ويليه قوسين ( ) توضع بينهما معاملات الدالة إن وجدت – ويليها بلوك من الأوامر يبدأ وينتهى بالقوسين"{ " , "} "- والدالة الرئيسية هى أول ما ينفذه الكمبيوتر عند تنفيذ البرنامج – ويمكن أن تسبقها كلمة من الكلمات المحجوزة void لتوضح أن الدالة ليس لها قيمة ترجع بها.

ومن فوائد الدوال ما يلي:-

1. تساعد الدوال المخزنة في ذاكرة الحاسوب على اختصار البرنامج إذ يكتفي باستدعائها باسمها فقط لتقوم بالعمل المطلوب.
2. تساعد الدوال المخزنة في مكتبة الحاسوب على تلافي عمليات التكرار في خطوات البرنامج.
3. تساعد الدوال الجاهزة على تسهيل عملية البرمجة نفسها.
4. توفر مساحة من الذاكرة المطلوبة
5. اختصار زمن البرمجة وتنفيذ البرنامج بأسرع وقت ممكن.

أمثلة لبعض المكتبات والدوال التي تحويها:

1. المكتبة math تحوي كل الدوال الرياضية.
2. المكتبة conio تحوي دوال التعامل مع الأحرف وتثبيت المخرجات.
3. المكتبة string تحوي دوال التعامل مع السلاسل النصية.

==========

بعض الدوال الرياضية:

وهذه الدوال تحتاج عند تنفيذها إلى المكتبة <math.h>

والجدول التالي يبين بعض الدوال الرياضية المستخدمة:

الدالة	اسم الدالة باللغة الانجليزية	معنى الدالة باللغة العربية
abs (a)	Absolute value	تعيد القيمة المطلقة للمتغير a
log (b)	Log faction	تعيد قيمة اللوغاريتم للمتغير b
pow (x,y)	Power function	ترفع x للقوة y هكذا x^y
sin (x)	Sin function	دالة الجيب الهندسية
cos (x)	Cos function	دالة جيب التمام الهندسية
sqrt (z)	Square root function	دالة الجذر التربيعى
strcpy (s1,s2)	Copy string	نسخ جملة S1 إلى جملة S2

موجهات ما قبل المعالجة:

العبارة التى يبدأ بها البرنامج وهى <iostream.h> #include ليست جزء من البرنامج بل هى إعلان عن ملف يحتوى على تعريف مجموعة الدوال المبنية التى نحتاجها أثناء البرمجة.

وتبدأ العبارة #include بما يسمى علامة توجيه قبل المعالجة وهى علامة العدد # وتسمى هاش ويليها موجه المعالج include.

==========

7-2) تطبيقات

تطبيق (1)

قم بكتابة كود برنامج يقوم بعرض الجملة التالية على الشاشة:

Hallow World

I am a programmer

الحل:

كما سنرى فإننا لن نستخدم أى متغيرات لأننا لن نقوم بتخزين أى شئ بل كل ما علينا هو عرض الجمل المذكورة على الشاشة.

Code

```
#include <iostream >
 int main()
{
 cout <<"Hallow World \n I am programmer" << endl;
 return 0;
}
```

من الكود السابق تجد أن سطراً واحدا هو الذى نفذ المطلوب وهو سطر دالة الإخراج cout والمستخدم بها سلسلة الإفلات \n وهى للانتقال إلى سطر جديد.

-نجد إننا ذكرنا اسم ملف المكتبة iostream بدون الامتداد(.h) والسبب فى ذلك استخدام السطر الثانى الذى يحدد مساحة الأسماء std.

==========

أسئلة الباب الثانى

1-عرف مايلي:

-الثوابت -المتغيرات

مع ذكر مثال لكلا منهما

2-ماهى أنواع المتغيرات المختلفة؟

3-بين نوع المتغيرات الآتية:- int a,b ; float x,y ; char c,d ;

4-بين كيف يمكن الإعلان عن المتغيرات الصحيحة والحقيقية والحرفية مع تعريف كل نوع وإعطاء مثال لكل نوع .

5- ماهى أولويات تنفيذ العمليات الحسابية في لغة C++؟

6- ماهى نتائج تنفيذ الاتي:

```
a- cout<<"this is my first program\n";
cout<<"the result is";
 b-cout<<"one\t\t";
cout<<"two\n";
c- cout<<100<<200<<300;
d- cout<<10<<endl<<20<<"\n"<<30
```

7- اكتب برنامج يطلب من المستخدم إدخال رقم صحيح من لوحة المفاتيح ثم يقوم بطباعة مكعب هذا الرقم على الشاشة.

8- قم بدراسة البرنامج الآتي ثم أجب عن الأسئلة الآتية:

```
#include<iostream.h>
void main()
{
int x,y;
```

==========

```
float f;
cout<<"enter the two integer numbers\n";
cin>>x>>y;
f=(x*x-3*y)/(2*x-y);
cout<<"f="<<f;
}
```

أ- ما هو الغرض من البرنامج السابق؟

ب-هل الإجابات التالية صحيح ولماذا عندما تكون x=3 y=2 ؟

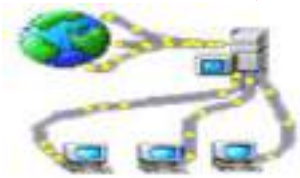
(1) 1.0 (2)0.75 (3) 0 (4) error

ج- أى الإجابات التالية صحيح ولماذا عندما x=1 y=2 ؟

(1) -5.0 (2) -2 (3) 0 (4) error

==========

الباب الثالث:



هياكل التحكم

1-3 تطوير الخوارزميات

2-3 التعبيرات المشروطة

3-3 جمل الاختيار

4-3 جمل الدورانات loops

5-3 تطبيقات

==========

الباب الثالث

هياكل التحكم:

كثيراً ما نحتاج أن نقوم بتنفيذ بعض الجمل عند تحقيق شرط معين أو بعض الشروط - أو تكرار عملية من عمليات الإدخال أو الإخراج أو الحساب عدد من المرات – حسب متطلبات وطبيعة المشكلة المراد حلها بالبرنامج – لذا نحتاج أن نتعلم أساليب الشرط والتكرار وكيفية التحكم فى مسار البرنامج كما تعد أساليب الشرط والتكرار والتحكم بمثابة قلب وروح فى جسم لغات البرمجة وبدونها لا يمكن تنظيم أى برنامج وتوفر لغة C++ للمبرمج عددا من الأساليب والدوال الفعالة , المتعلقة بهذا الشأن , وتمتاز هذه الأساليب بأنها بنائية structured أى يمكن تنظيم عمليات التحكم والتكرار فيها ذاتياً من بداية العمليات وحتى آخر البرنامج دون تدخل من المبرمج أثناء هذه العمليات للإشراف على التوجيه والتخطيط لكل خطوة من خطوات البرنامج.

وعرف المتخصصون فى لغة C++ البرمجة البنائية أنها البرمجة التى لا تستعمل جملة الانتقال GOTO , لتوجيه البرنامج فى كل خطوة , ومع ذلك فأن لغة C++ توفر جملة الانتقال GOTO ولكنها لا تستعمل إلا للضرورة.

وحيث أن جواب الشرط إما أن يكون صوابا true أو خطأ false, ولغة C++ تعطى قيمة عددية لكل من حالتى الصواب والخطأ بحيث تكون القيمة العددية الممثلة لحالة الصواب تختلف عن الصفر وتعطى قيمة صفر لحالة الخطأ (عند عدم تحقيق الشرط) مما يؤكد مرونة لغة C++ فى استخدام عدد كبير من الدوال ,وفى توجيه البرنامج بطريقة فعالة.

==========

1-3) تطوير الخوارزميات.

كما عرفنا من قبل يمكن تعريفها بأنها مجموعة من القواعد والعمليات المعرفة جيداً لحل المشكلة في عدد محدد من الخطوات.

وتستخدم عوامل مساعدة على كتابة خوارزميات المشكلة المطلوب حلها ببرنامج بلغة الـ C++ مثل المخطط الانسيابي والمعاملات الحسابية والمنطقية وخطوات ثابتة يتبعها المبرمج لتصميم الخوارزم للبرنامج وتحويله الى برنامج مكتوب بتعليمات وأوامر لغة الـ C++.

الطرق الأساسية لكتابة الخوارزم:

أ- أن تكون كل خطوة معرفة جيداً.

ب- أن تتوقف العمليات بعد عدد محدد من الخطوات.

ج- أن تؤدي العمليات إلى الحل الصحيح للمسألة.

العمليات الحسابية و الأدوات المستخدمة فيها:

للجمع تستخدم	+	للضرب	*
للطرح	-	للزيادة بمقدار واحد	++
للقسمة	/	للنقصان بمقدار واحد	--
باقي القسمة	%		

أولويات العمليات الحسابية

١ -الزيادة و النقصان عندما تأتي قبل العدد	٥ -الجمع و الطرح
٢ -الأقواس	٦ -التساوى
٣ -إشارة السالب	٧ -الزيادة و النقصان المتأخرة بعد العدد
٤ -القسمة و باقي القسمة و الضرب	

ملاحظة:

في حال وجود عمليتين لهما نفس الأولوية نبدأ بتنفيذ العملية الأقرب إلى اليسار

==========

الأدوات العلاقية:

وهي حسب هذا الجدول:

الأكبر	>
الأصغر	<
أكبر أو يساوي	=>
أصغر أويساوي	=<
إن كان يساوي	==
إن كان لايساوي	=!

2-3)التعبيرات المشروطة : Conditional Expressions

هل تتذكر المعاملات العلائقية ، ستظهر فائدتها هنا لنفرض أن لدينا ثلاثة متغيرات ، حيث أننا نقوم بكتابة برنامج يقوم بمقارنة أي عددين وحساب الأكبر منهما ، لنفرض أن المتغيرين أو العددين المراد مقارنتهما هما a,b والمتغير الثالث Max

```
1 if (a > b )
2 max = a ;
3 if ( b > a)
4 max = b ;
5 if ( b == a)
6 max = a = b;
```

لنفرض أن المتغير b هو أكبر من المتغير a حينها سيتم تنفيذ السطر الثاني أما في حال لم يكن كذلك فلن يتم تنفيذ السطر الثاني وسيواصل البرنامج عمله وينتقل إلى السطر الثالث.

انظر أيضاً إلى عملية المقارنة في السطر الخامس وهي == أي هل في حال كانا متساويان(المتغير aيساوي المتغير b) فإن السطر السادس سيتم تنفيذه ، انظر أيضاً أننا في حالة المساواة لم نقم بكتابة

==========

المعامل (=) والسبب أن المعامل (=) كما قلنا سابقاً هو معامل إسناد أي يأخذ القيمة التي على يمينه ويضعها على يساره ولا يقوم بمقارنة أبداً أما المعامل(= =) فيقارن بين القيمتين.

الجمل الشرطية:

تتعامل لغة الـ C++ مع ثلاثة أنواع من جمل الشرط وهى:

1-جملة إذا الشرطية ومشتقاتها if statement

2-جملة التوزيع switch statement

3-جملة أداة الشرط

استخدام عبارة التحكم IF:

تقوم عبارة if باختبار شرط معين , إذا تحقق هذا الشرط يتم تنفيذ عملية أو مجموعة عمليات وإلا يتم تجاهل هذه العملية لذا فالصيغة العامة لعبارة if تكون كالآتى:

```
If (expression) Statement;
```

أو

```
If (condition) Statement1;
```

إذا لم يتحقق الشرط condition ينقل تسلسل تنفيذ البرنامج إلى الجملة statement1

أو تستخدم بشكل أعمق كالتالى:

```
If (condition) Statement1;
Else Statement2;
```

فى هذه الحالة ستنفذ الجملة statement1 إن تحقق الشرط وإلا فإن التنفيذ ينتقل إلى السطر التالى وتتحقق الجملة statement2.

==========

حيث يعبر expression أو condition عن التعبير المستخدم (الشرط) ويوضع بين قوسين فإذا كانت قيمته لا تساوى صفر يكون الشرط صحيح ويتم تنفيذ العبارة statement.

==========

للتعرف على طريقة تشغيل العبارة if ندرس المثال التالى:

برنامج بسيط بلغة C++ لإظهار العبارة "x is positive" على الشاشة

```
# include <iostream.h>
main()
{
int x ;
cout<<"Enter the test number X ";
cin>>  x ;
if (x>0)
cout << " x is positive";
return 0;
}
```

نتيجة البرنامج:

إذا أدخلت قيمة العدد المطلوب اختباره يساوى 7 وضغطت enter يعطى الرسالة

x is positive Press any key to continue

وإذا أدخلت قيمة العدد (-7) يعطى البرنامج رسالة press any key to continue

مثال (1):

يمكن استخدام الصيغة الكاملة if.....else

ليعطى البرنامج رسالة فى كلا الحالتين عند إدخال قيمة x موجبة يعطى رسالة القيمة موجبة وعند إدخال قيمة سالبة يقوم البرنامج بطباعة رسالة أن القيمة المدخلة سالبة كما هو موضح فى البرنامج البسيط التالى:

==========

الحـل:

```
# include <iostream.h>
main()
{
int x
cout<<"Enter the test number x ";
cin>>  x ;
if (x>0)
cout << " x is positive\n";
else
cout<<"x is negative\n";
return 0;
}
```

وعند تطبيق البرنامج السابق نجد أن البرنامج يطبع العبارة (x is positive) في حالة القيمة الموجبة ويطبع العبارة (x is negative) في حالة القيمة السالبة ولكن في كل من البرنامجين السابقين هناك خطأ منطقي لا يكتشفه المترجم عند الترجمة وهو ماذا لو تم إدخال قيمة x = 0 في هذه الحالة سيطبع البرنامج العبارة (x is negative) وهذا غير صحيح لأن الصفر لا يكون موجبا أو سالبا وهنا يجب على المبرمج الاهتمام وملاحظة مثل هذه الأخطاء وتجربة البرنامج أكثر من مرة وإعطاؤه عينات كثيرة من الأرقام المختلفة ولحل هذه المشكلة يمكن استخدام هذه الطريقة الموضحة بالبرنامج التالى:

==========

```
# include <iostream.h>
main()
{
int x
cout<<”Enter The test number X ”;
cin>>  x ;
if (x>0)
cout << “ x is positive\n”;
else if (x<0)
cout<<"x is negative\n";
else if(x= =0)
cout<<" x=0\n";
return 0;
}
```

ونلاحظ هنا أننا وضعنا أكثر من جملة if في هذا البرنامج وبالطبع يمكن كتابة البرنامج السابق بعدة طرق وكلما كانت الطريقة أقصروتقوم بالغرض المطلوب كلما كان هذا أفضل.

كما يجب أن تلاحظ علامة التساوي (= =) في جملة if الأخيرة.

==========

3-3) جمل الاختيار:

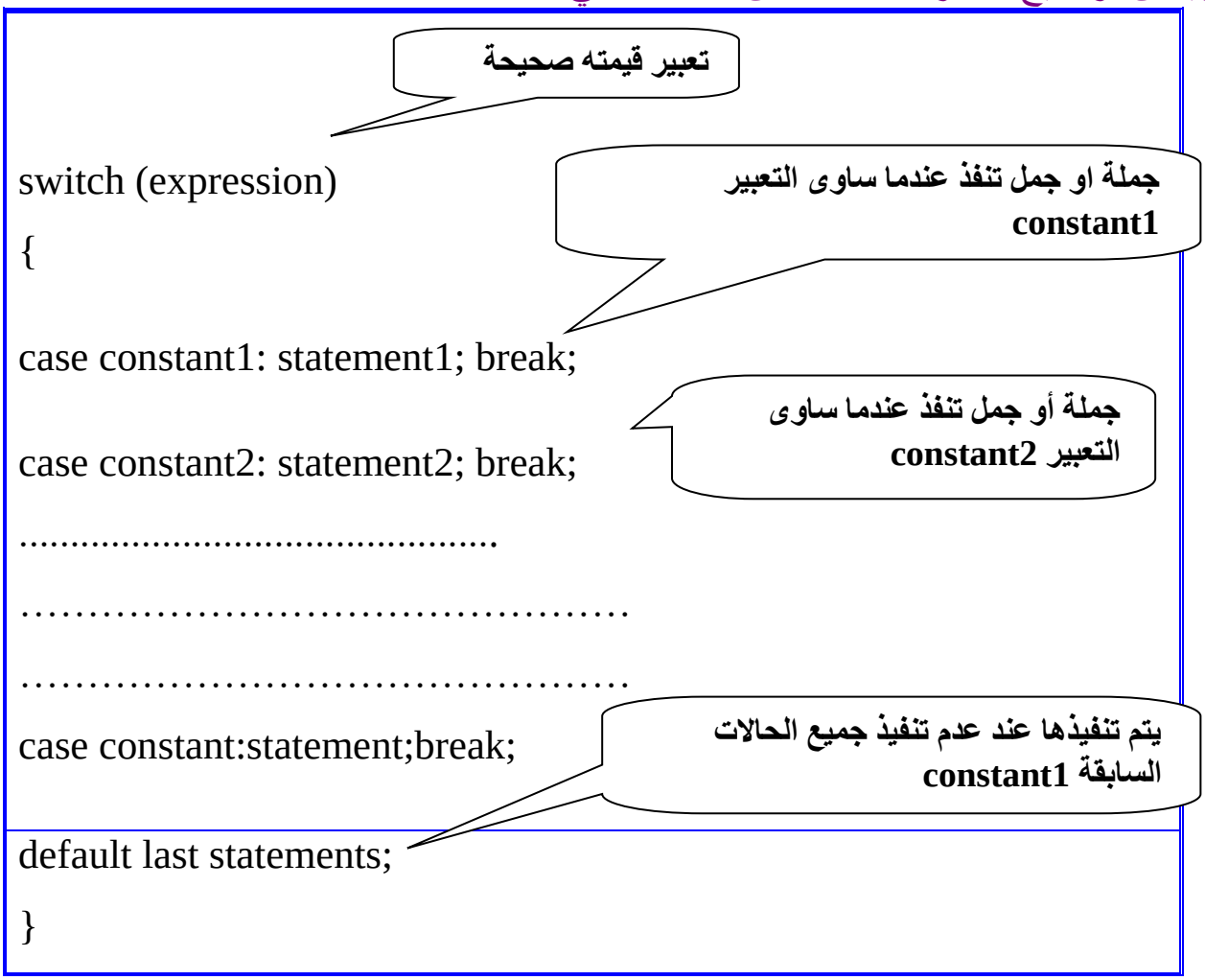
عبارة switch:

تستخدم عبارة if إذا كان جواب الشرط عبارة عن احتمالين أو ثلاثة احتمالات على الأكثر أما إذا زاد عدد الاحتمالات على ذلك فمن الأفضل استخدام عبارة switch وصيغتها كالتالى:

```
Switch (condition)
{
case value1 ;
statement1;
break;
case value2;
 statement2;
break;
case value N;
statementn;
break;
default:
statement def;
break;
}
```

==========

ويمكن توضيح العبارة switch من الشكل التالي:



وكما نرى أن الاختيار المتعدد البدائل يبدأ بكلمة (Switch) يليها متغير الاختيار والذي تحدد قيمته الاختيار الذي سيتم تنفيذه، ويلي ذلك قوس بلوك كبير يحتوي داخله بلوكات صغيرة كل منها يمثل اختيارا من البدائل المطروحة وكل بلوك من بلوكات البدائل يبدأ بكلمة(case) متبوعة بقيمة لمتغير الاختيار والتي تمثل الشرط وبعد ذلك تأتي عبارة النتيجة.

ويختتم بلوك البديل بكلمة (break) والغرض منها هو منع الكمبيوتر من تنفيذ عبارة النتيجة التالية.

وهنا يتبادر للذهن سؤال - ألم يتحقق الشرط الأول مثلا فماذا يدفع الكمبيوتر لتنفيذ بقية عبارات النتائج ؟

والإجابة عن هذا السؤال هى أن عبارة الإدخال من متعدد البدائل لا ترسل للكمبيوتر أمرا بالتوقف بعد تحقق أى شرط فيها لذا لزم الاستعانة بكلمة (break).

وبعد نهاية بلوكات البدائل تأتي كلمة (default) متبوعة بعبارة أو بعبارات ينفذها الكمبيوتر في حالة عدم تحقق أى من الشروط السابقة.

مثال (2):

برنامج لطباعة قيمة عندما يتحقق شرط من شروط متعددة باستخدام دالة case

```
#include < iostream.h>
void main()
{
int s1;
s1=2;
switch (s1)
{
case 2:cout<<"y";
break;
case 3: cout<<"x";
break;
case 4: cout<<"m";
break;
default: cout<<"w";
}
}
```

وتكون نتيجة البرنامج طباعة حرف(y) وذلك لتحقق الشرط الأول وهو s1=2

==========

جملة أداة الشرط:

وهى أداة سريعة وهى مكافئة لعبارة (if.....else)

وصورتها العامة هى:

Variable=(condition)? Result1:result2;

ومعناها: أنه يتم تنفيذ النتيجة الأولى result 1 عندما يكون جواب الشرط condition متحققا (true) وإلا فيتم تنفيذ النتيجة الثانية resulte2 عندما يكون جواب الشرط (false).

مثال(1):

```
#include < iostream.h>
void main()
{
int a,b;
a=5;
if (a>1) b=10;
else
b=20;
cout<<b;
}
```

ومعناها أن b تأخذ القيمة 10 إذا تحقق الشرط a>1 وتأخذ القيمة 20 إذا لم يتحقق الشرط نفسه.

وتكون نتيجة البرنامج 10 حيث a=5 أى a>1.

==========

4-3) جمل الدورانات loops والحلقات التكرارية:

كثيرا ما نحتاج في البرامج إلى تكرار أمر موجه للكمبيوتر عددا من المرات، وتوفر لغة C++ عدة وسائل تمكن المبرمج من أداء هذا التكرار.

وعادة ما تسمى هذه الوسائل (الحلقات التكرارية) ويوجد العديد من الحلقات التكرارية في لغة C++ سنتناول منها هنا:

1- الحلقة for(for loop)

2- الحلقة while(while loop)

3- الحلقة (do-while loop)

1- الحلقة for : (for loop)

تستخدم الحلقة for لتكرار أمر معين (أو مجموعة من الأوامر) عددا من المرات وتحتاج الحلقة إلى ثلاث عناصر أساسية كما هو موضح بالشكل التالي:

```
for ( counter statement; condition; step)
```

وهذه العناصر هى:

- العداد (counter): وظيفته هى تسجيل عدد مرات التكرار.
- الشرط (condition): هو الشرط الذي يحدد نهاية التكرار إذ يظل التكرار قائما حتى ينتفي الشرط.
- الخطوة (step): وهى القيمة التي تحدد عدد مرات التكرار.

==========

مثال(1):

لتنفيذ حلقة تكرارية لطباعة الأعداد من 1 إلى 20 باستخدام for loop:

```
#include <iostream.h>
main()
{
int counter;
for (counter=1; counter<=20; counter++)
cout<<counter;
return 0;
}
```

ومن البرنامج السابق نجد أن الحلقة for بدأت بكلمة (for) متبوعة بقوسين بينهما ثلاثة عبارات تفصل بينها علامة الفاصلة المنقوطة.

العبارة الأولى تخزن القيمة الابتدائية في العداد.

والعبارة الثانية هى الشرط وهنا الشرط أن قيمة العداد أقل من أو تساوى 20.

أما العبارة الثالثة فهى تحدد الخطوة، وفي هذا البرنامج يزداد العداد بمقدار 1 كل مرة تنفذ فيها الحلقة.

والبرنامج السابق ينتج عنه طباعة الأرقام من 1 إلى 20 كالاتي:

12345678910111213141516171819 20

ملاحظات:

- العبارات الثلاثة المكونة لحلقة for يجب أن تنفصل عن بعضها بالفاصلة المنقوطة، وهذا الخطأ من الأخطاء الشهيرة جدا في عالم البرمجة لذا وجب توخي الحذر.
- في حالة تكرار أكثر من أمر يتم استبدال العبارة التي تلي بداية الحلقة for في المثال السابق أى (cout<<counter;) ببلوك يحوي العبارات المراد تنفيذها.

==========

الصيغة العامة الثانية لجملة for:

```
for ( intial- value; condition; increment)
{
statement;
}
```

جملة أو أكثر

ولفهم هذه الصيغة نتابع الأمثلة التالية:

مثال(1):

```
#include < iostream.h>
main ()
 {
 int x,y,z;
y=-4;
for(x=1;x>y;x=x-2)
{
z=x;
cout<<x<<endl;
}
return 0;
}
```

والناتج سيكون كالآتي:

1

-1

-3

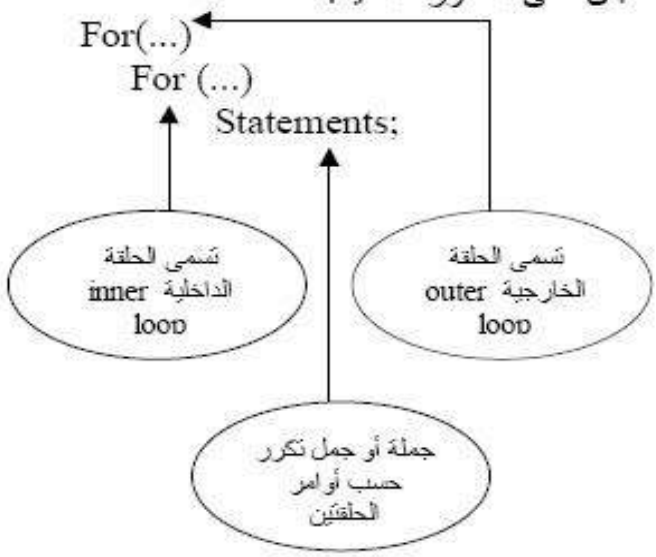
==========

حلقات التكرار المتداخلة Nested (Multiple) for Loops

وهى تأخذ الصيغة الآتية:

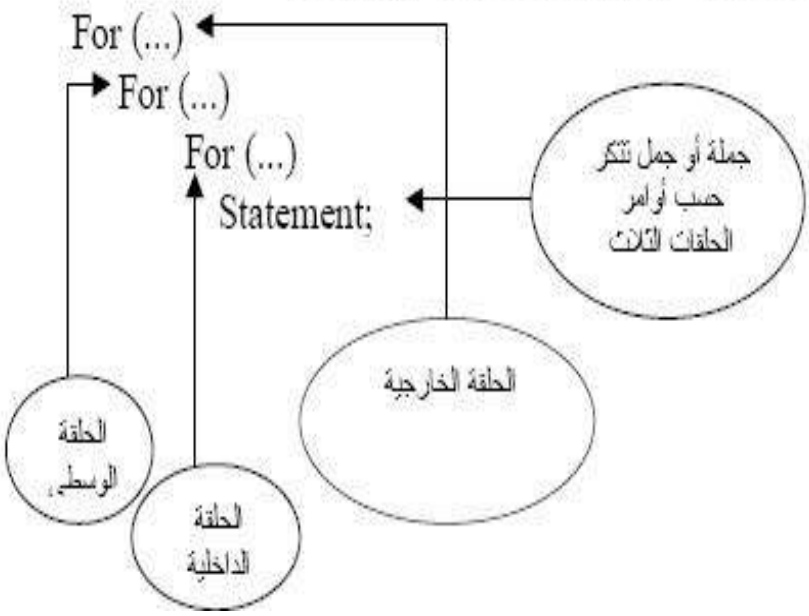
```
For (…)
   For (…)    …….
      For (…)  …….

                    Statements;
```

فلو أخذنا حالة حلقتين متداخلتين فانهما تكتبان على الصورة التالية:



وتكون في هذه الحالة الجملة (أو الجمل) جزءا مكررا مرتبطا بالحلقة الداخلية والحلقة الداخلية تتكرر حسب أوامر الحلقة الخارجية وهكذا.

وفي حالة الثلاث حلقات المتداخلة ، فإنها تكتب على الصورة التالية:



مثال(1):

تنفيذ حلقة داخلية أربعة مرات داخل حلقة تتكرر ثلاث مرات.

```
#include < iostream.h>
main ()
{
int i,j;
for (i=1;i<=3;++i)
for (j=1;j<=4;++j)
cout<<i*j;
return 0;
}
```

ونلاحظ في هذا المثال أن الحلقة الخارجية تتكرر ثلاث مرات والحلقة الداخلية تتكرر أربع مرات لكل قيمة من قيم i أى مجموع الدورانات إثنا عشر مرة.

ويكون الناتج 1234246836912

==========

الحلقة (while loop):

في هذه الحلقة التكرارية نحتاج إلى الشرط فقط وطالما كان هذا الشرط متحققا استمرت الحلقة في التكرار.

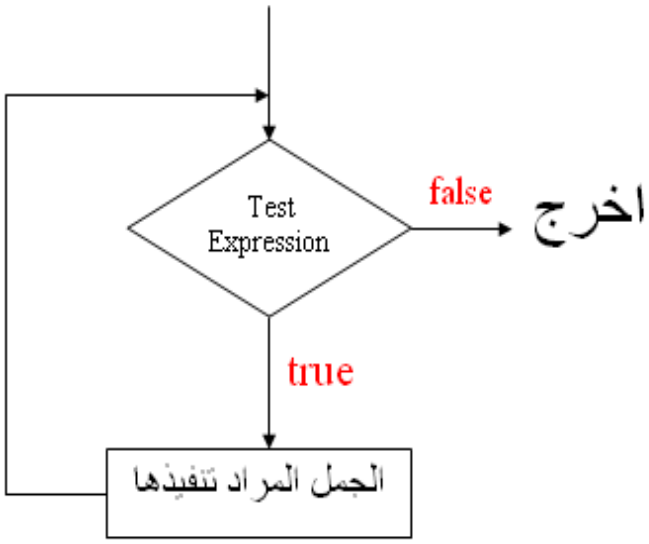
والصورة العامة للحلقة while موضحة كما بالشكل التالي:

```
while (condition )
{
statement 1;
statement 2;
--
--
statement n;
}
```

حيث الـ (condition):

هو الشرط اللازم لأداء التكرار والعبارات بداخل أقواس البلوكات هى العبارات المراد تكرارها..

ومن المخطط التالي نستطيع فهم طريقة عمل العبارة



==========

مثال (1):

يوضح استخدام الحلقة while لطباعة الأعداد من 1 إلى 20.

```
#include <iostream.h>
main()
{
int counter=1;
while (counter <=20 )
      {
cout<< counter;
counter++;
      }
return 0;
}
```

من المثال السابق يمكننا استخلاص النتائج التالية عن الحلقة while

1. تخصيص القيمة الابتدائية للعداد تتم خارج الحلقة while.
2. زيادة العداد تتم داخل الحلقة while.

ويكون ناتج البرنامج كالتالي:-

12345678910111213141516171819 20

مثال (2): طباعة الأعداد من 1 الى 10 مع تنسيق الخرج:

```
1  //*****************************************************
2  // برنامج يستخدم حلقة while لاظهار الارقام من 0 الي 10
3  //*****************************************************
4  #include<iostream>       //for cin cout
5  #include<conio.h>        //for getch()
6  using namespace std;
7  //*****************************************************
8  void main()
9  {
10      int i=0;
11      while(i<=10)
12      {
13          cout<<"the number = "<<i<< " "<<endl;
14          i++;          //  الزيادة بمقدار واحد
15      }
16      getch();          //  هذا الامر حتى لاتغلق الشاشة بسرعة
17 }
```

==========

الهدف من البرنامج : طباعة الأعداد من 1 إلى 10 مع تنسيق الخرج.

ويكون ناتج البرنامج كالتالي:

```
the number = 0
the number = 1
the number = 2
the number = 3
the number = 4
the number = 5
the number = 6
the number = 7
the number = 8
the number = 9
the number = 10
```

الحلقة التكرارية (do-while):

تختلف هذه الحلقة عن الحلقتين السابقتين في مكان كتابة الشرط، حيث يكتب الشرط هنا بعد العبارات المطلوب تكرارها.

والصيغة التالية توضح الصورة العامة للحلقة do –while

```
do
{
statement 1;
statement 2;
--
--
statement n;
}
while (condition)
```

==========

وأهم ملاحظة على الحلقة التكرارية do- while أنها تنفذ العبارات المطلوب تكراراها مرة واحدة على الأقل حتى ولو كان الشرط غير متحقق !!!

وتفسير ذلك أن التحقق من الشرط يتم بعد التنفيذ وليس قبله كما في الحلقتين السابقتين.

مثال (1):

إدخال أسماء حرفية إلى البرنامج حسب حاجة المستخدم وعدد مرات التكرار

```
//*********************************************************
// do-while برنامج يشرح استخدام**********************
//*********************************************************
#include<iostream>
#include<string>
#include<conio.h>
using namespace std;
//*********************************************************
void main()
{
    string      name;
    char        choice;
    do
    {
        cout<<"Enter name :";
        cin>>name;
        cout<<"you want enter anothr name ?";
        cin>>choice;
    }
    while(choice=='y');
    getch();
}//end main()
```

==========

شرح المثال السابق:

في هذا المثال يطلب منك البرنامج إدخال الاسم أى اسم (name) ويضعه في متغير حرفي من النوع string ثم يسألك البرنامج بعد إدخال الاسم هل تريد إدخال اسم آخر فإذا ضغطت(y) فإنه سيطلب منك إدخال اسم آخر أما إذا كتبت أى حرف آخر سيتم الخروج من البرنامج.

- **حلقات while المتداخلة while nested Loops:**

تشبه حلقات while المتداخلة حلقات for المتداخلة ويمكن فهمها من المثال التالي:

==========

جملة الإيقاف break:

من معنى الاسم نستطيع أن نلاحظ أن وظيفة break هى إيقاف حلقة التكرار عند تحقق شرط أو شروط معينة وعند تنفيذها يتم القفز إلى سلسلة الجمل التالية للبرنامج أو حلقة التكرار وتستعمل break أيضا في إيقاف حلقة التكرار اللانهائية أو الخروج منها إلى الجمل التي تليها ويمكن فهم طريقة عملها من الأمثلة الآتية:

مثال(1):

لطباعة الأعداد من 1 إلى 10.

```
# include <iostream>
main()
{
int I;
for(i=1; i<100;- -i)
{
cout<<i
if(i==10)break;
}
return 0;
}
```

هذه الجملة لوقف تنفيذ حلقات التكرار عندما يصبح i=10

وطبعا سيقوم البرنامج بكتابة الأعداد من 1 الى 10.

12345678910

==========

مثال (2): لطباعة الأعداد الفردية من 100 : 1 على الشاشة.

```
#include < iostream.h>
main ()
{
int number;
for (number=1; number<=100;++ number)
{
if (number%2) // عدد فردي
break;
cout<<"number"<<number;
}
cout<< number<<endl;
return 0;
  }
```

ونلاحظ هنا أن السطر cout<<"number"<<number لن يتم تنفيذه لتحقق شرط جملة if وتفعيل الأمر break والذي يقوم بالخروج من الحلقة وتنفيذ أمر الطباعة cout<<number<<endl;

==========

جملة الاستمرار continue:

تعمل جملة الاستمرار continue على تجاوز تنفيذ بقية الجمل في التكرار خلال الدورة الحالية والانتقال إلى الدورة الثانية:

مثال (1):

```
#include<iostream.h>
main()
{
Int x,n;
do
{
cin>>x>>n;
If(n>1)continue;
 //تعمل على تجاوز تنفيذ الجملتين التاليتين وتبدأ دورة جديدة إذا تحقق الشرط
cout<<x;
++n;
}
while(n>1);
return0;
}
```

مثال (2):

اكتب برنامج يقوم بطباعة الأرقام التى تقبل القسمة على 2 ،4 ، 6 بدون باقى بين الأعداد من 1 إلى 100.

==========

الحل:

```
#include < iostream.h>
main ()
{
int number;
for (number=1;number<=100;++number)
{
if (number%2)
continue;
else if (number%4)
continue ;
else if (number%6)
continue;
else
cout<<number<<endl;
}
return 0;
}
```

ويكون الناتج كالتالى:

```
12
24
36
48
60
72
84
96
```

==========

جملة الانتقال goto:

وهى من الجمل المستخدمة للتحكم في مسار البرنامج أى التي تحول مجرى البرنامج إلى أسطر أخرى بمعنى أذهب إلى.

مثال (1): لطباعة العدد x طالما تحقق شرط أن قيمة x أقل من 10

```
#include < iostream.h>
main ()
{
int x;
input: cin>>x;
if (x<10)
cout<<"value is under 10"<<endl;
goto input;
return 0;
}
```

في هذا البرنامج يطلب من المستخدم إدخال قيمة فإذا كانت القيمة أقل من 10 حسب الشرط فإنه يعرض لك رسالة value is under10 ثم تخبره العبارة goto بالذهاب مرة أخرى إلى سطر الإدخال input. أما إذا كانت أكبر من العدد 10 فإنه لا يقوم بطباعة شيء لعدم تحقق الشرط وتخبره العبارة goto بالعودة مرة أخرى إلى عبارة الإدخال في السطرinput.

==========

3-5) تطبيقات

تطبيق (1):

اكتب برنامجا لطباعة الأعداد الفردية من 1 إلى 15.

```
#include < iostream.h>
main ()
{
int a;
for (a=1;a<=15;a=a+2)
cout<<a<<endl;
return 0;
}
```

ومن الملاحظ أننا جعلنا قيمة الزيادة 2 وليس 1 لأن المطلوب عدد فردي

وتكون النتيجة كالاتي:

```
1
3
5
7
9
11
13
 15
```

==========

تطبيق (2):

أكتب برنامج يطلب من المستخدم إدخال قيمة عددية وطالما أن القيمة المدخلة أكبر من أو تساوي (0) يطبع العلامة (*) على سطر جديد.

```
#include<iostream.h>
main()
{
int a;
cout<<"Please enter a number";
cin>>a;                          10    ندخل الرقم وليكن
while(a>=0)    طالما أن الرقم المدخل أكبر من أو يساوى صفر
{
cout<< "" <<endl;          إذا تحقق الشرط أطبع
cout<<"Please enter a number";
cin>>a;
}
return 0;
}
```

تطبيق (3):

برنامج يقوم بعمل الآلة الحاسبة باستخدام العبارة switch

```
#include<iostream.h>
1- int main()
2- {
3- float a,b;
4- char x;
5- cout << "Enter Number1:\t" ;
```

```
6- cin >> a;
7- cout << "Enter Number2:\t" ;
8- cin >> b;
9- cout << "Enter the operator\t";
10- cin >> x;
11- cout << endl << endl;
12- cout << "Result:\t";
13- switch (x) {
14- case '+':
15- cout << a+b ;
16- break;
17- case '-':
18- cout << a-b;
19- break;
20- case '*':
21- cout << a*b;
22- break;
23- case'/':
24- cout << a/b;
25- break;
26- default:
27- cout << "Bad Command";
28- }
29- cout << endl;
30- return 0;
```

==========

31- }

شرح البرنامج:

هذا البرنامج هو تطبيق شامل للعبارة switch ونلاحظ أنه تم ترقيم السطور لسهولة التعامل مع البرنامج ولكن عند كتابته في المترجم يجب إزالة الترقيم.

ويقوم هذا البرنامج بالعمليات الحسابية المختلفة من جمع وطرح وضرب وقسمة.

وعند تنفيذ البرنامج يطلب منك الاتي:

1- إدخال الرقم الأول.

2- إدخال الرقم الثاني.

3- إدخال رمز العملية(+ - * /).

4- ثم يقوم بالعملية الحسابية وطباعة الناتج.

==========

أسئلة الباب الثالث

1- قم بكتابة برنامج يقوم بطباعة الأعداد من 1 إلى 300 على أن تكون المخرجات على الشاشة مرة متجاورة ومرة أخرى تحت بعضها

وذلك باستخدام عبارات التكرار الآتية:

(1)FOR LOOP (2) WHILE LOOP (3)do while

2- بين الأخطاء الموجودة في البرنامج التالي مع التصحيح:

```
#include iostream.h
main[]
int x,4.5;
float 5;
c=r+x;
cout>>c;
cin<<g;
for(g=0;g<=100;--g)
cout g;
return 0;
}
```

3- قم بكتابة برنامج يطلب من المستخدم إدخال رقمين ثم يقوم بطباعة الرقم الأكبر؟

4- صمم برنامج بلغة C++ لجمع الأعداد من 1 : 20