أساسيات البرمجة بلغة C#

منشورات جامعة دمشق
الكلية التطبيقية

البرمجة (1)

أساسيات البرمجة بلغة C

الدكتور المهندس مأمون يونس
أستاذ مساعد
في قسم هندسة الحواسيب والأتمتة

جامعة دمشق :

الفهرس

الفصل الأول

البنية الأساسية للحاسوب

الفصل الثاني

مبادئ الخوارزميات

الفصل الثالث

أساسيات لغة #C

الفصل الرابع

بنى التحكم والتكرار

الفصل الخامس

التوابع (الطرائق)

الفصل السادس

المصفوفات

الفصل السابع

الجلسات العملية

مقدمـــــة

إن توسع استخدام النظم الحاسوبية وتطورها في السنوات الأخيرة ، فرض وجود شروط محددة على بنية الحواسيب ، وذلك من خلال المعدات والبرمجيات والتطبيقات الحاسوبية ولغات البرمجة التي تم تصميمها للاستخدام على نطاق واسع ، لأن لها تطبيقات متنوعة في مختلف مجالات الحياة. حيث إن اتساع مجالات استخدام الحواسيب وانتشارها ، وزيادة تطور علوم صناعة البرمجيات أدى إلى ظهور العديد من اللغات البرمجية عالية المستوى والتي يستطيع الدارس أن يتعلمها وفقاً لحاجته واختصاصه .

إن الحاسوب لا يمكنه أن يعمل بكفاءة إلا بتوجيهه من طرف المبرمج، ومعنى ذلك برمجته لتنفيذ متطلبات المبرمج . إذاً فبرامج الحاسوب مجموعة من الأوامر يقوم الحاسوب بتنفيذها، وتتراوح هذه الأوامر أو التعليمات ابتداء من بعض الأوامر القليلة التي تؤدي مهمة بسيطة إلى قائمة أوامر أكثر تعقيداً ؛ التي من الممكن أن تحتوي جداول من المعطيات . فبرامج الحاسوب لا تتم كتابتها مباشرة بلغة الآلة لأن البرمجة بهذه اللغة عملية مملة جداً وخطأ بسيط يكلف وقتا كثيراً لإيجاده ومن ثم تصحيحه ما يؤخر عملية الإنتاج لدى المبرمج وعوضاً عن هده اللغة ظهرت لغات البرمجة عالية المستوى ؛ التي تتم ترجمتها آلياً إلى لغة الآلة عن طريق برامج خاصة تسمى المفسرات والمترجمات.

وقد ظهرت مجموعة من لغات البرمجية عالية المستوى ، وقريبة من لغة الإنسان مثل :

BASIC, FORTRAN ,Pascal, PROLOG,C\C++,Java ,C#,...

وفي هذا الكتاب نلقي الضوء على مرتكزات أساسية في البرمجة بلغة # C وذلك من خلال البدء بفهم الخوارزميات وكيفية صياغتها وفق الطرائق المعروفة لصياغة الخوارزميات ومن ثم الانطلاق بكتابة البرامج اللازمة بلغة #C.

تُعد لغة #C من لغات البرمجة عالية المستوى ، وفي الوقت نفسه قريبة من لغة التجميع ذات المستوى المحدود ، كما أنها تعد لغة برمجة غرضية التوجه (البرنامج المكتوب عبارة عن صفوف و تستخدم الخواص المتاحة من التغليف و تعددية الأشكال و الوراثة والتركيب...). وهي لغة ناشئة من لغة JAVA ولغة C++ وتشمل لغة #C جميع مزايا لغة C++ و Java بالإضافة إلى مزايا البرمجة غرضية التوجه .

ويصف هذا الكتاب البنية الأساسية للحاسوب والبرمجة بلغة C# ، وتعلم أساسيات البرمجة بلغة C# ، حيث يتدرج من المفاهيم البسيطة وينتهي بشرح معمق لأكثر مواضيع اللغة تعقيداً ، ويدعم الشرح بالأمثلة الواضحة والقوية في آن واحد . كما يلي كل فصل مجموعة من المسائل العامة التي تدرب الطالب على حل المسائل بنفسه وتعمق فهمه للمواضيع وتطبيقاتها العملية .

والهدف الأساسي لهذا الكتاب هو تغطية منهاج مقرر " البرمجة(I) " في قسم تقنيات الحاسوب لطلاب السنة الأولى. وقد تم وضع مادة هذا الكتاب بحيث تمكن مختلف القراء من الاستفادة منه في مختلف التطبيقات . ويتكون من سبعة فصول وسرد للمراجع الأجنبية والعربية والمصطلحات العلمية .

ويتضمن الفصل الأول البنية الأساسية للحاسوب ، والهدف من هذا الفصل التعرف على المكونات الأساسية للحاسوب، وكما هو معروف فإن مكونات أي حاسوب لا تعمل بشكل مستقل منفصلة عن بعضها بعضاً ، بل تعمل معاً كنظام متكامل .

والفصل الثاني خُصص للتعرف على مبادئ الخوارزميات وأنواعها وكيفية صياغتها من خلال بعض الأمثلة التطبيقية.

والفصل الثالث يغطي أساسيات البرمجة في لغة C#، وذلك من خلال التعرف على الخطوات اللازمة لكتابة برنامج بلغة C# وتنفيذها والقواعد الواجب إتباعها وأيضاً التعرف على أنواع المعطيات والعمليات الحسابية والمنطقية .

وخُصص الفصل الرابع للتعرف على بنى التحكم والتكرار وأنواعها و الفرق بينهما ومتى تستخدم واحدة دون الأخرى.

ويغطي الفصل الخامس التوابع من حيث التصريح عنها وتعريفها وكيفية استدعائها لتنفيذ بعض المهام، بالإضافة إلى عملية ربط التوابع مع المصفوفات.

وخُصص الفصل السادس للتعرف على المصفوفات أحادية البعد والمصفوفات ثنائية البعد وكيفية التصريح عنها والتعامل معها من حيث إدخال قيماً لعناصرها وطباعة هذه القيم وأيضاً كيفية استخدام المصفوفات لإجراء بعض العمليات الحسابية ، وأيضاً تمرير المصفوفات إلى التوابع كوسطاء لها .

ويغطي الفصل السابع الجلسات العملية لمقرر البرمجة(1) ، ويتكون من ثمان جلسات عملية يتم تنفيذها في مخبر البرمجة(1) .

ونأمل أن نكون قد وضعنا للقارئ المبادئ الأساسية للانطلاق في مجال بنية الحاسوب وتنظيمه ولغات البرمجة، وأسهمنا بتقديم المادة العلمية المتواضعة لكوادرنا الهندسية لتساعدهم في تطوير معلوماتهم ، بحيث تتماشى مع التطور الكمي والنوعي في مجال هندسة الحواسيب والأتمتة وتقنيات الحاسوب.

ولا يفوتنا أن نتقدم بالشكر والعرفان إلى جميع الزملاء الذين أسهموا في إبراز هذا العمل إلى حيز الوجود .

والله ولي التوفيق

المؤلف

دمشق /تموز / 2019

الفصل الأول

البنية الأساسية للحاسوب

البنية الأساسية للحاسوب

١-١- مقدمة

إن تطور الحواسيب منذ إنتاج أول آلة حاسوبية ميكانيكية في عام 1632 وحتى بداية الحرب العالمية الثانية كان تطوراً عادياً وكان بعضها ميكانيكياً والآخر تطوراً كهربائياً ولكن الحواسيب التي تم إنتاجها خلال فترة الجيل الأول تعد تحولاً كبيراً في استخدام المعدات الميكانيكية والكهربائية و تطوير أجهزة الكترونية مبرمجة .

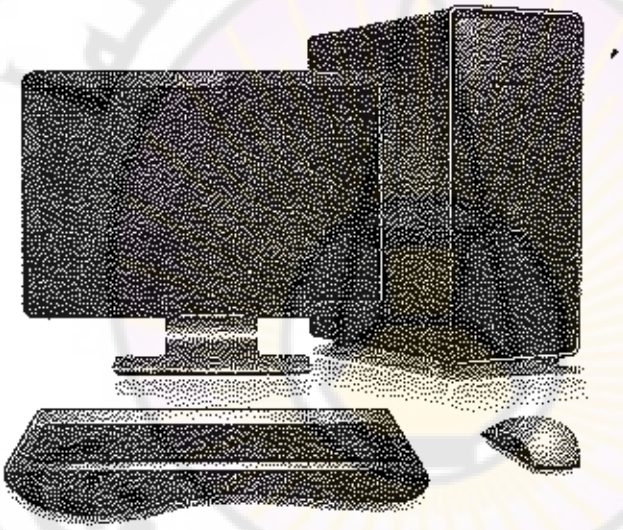
ويمكن القول إن الحواسيب تطورت بشكل فعلي منذ بداية الخمسينات وحتى اليوم تطوراً هائلاً تجاوز كل التوقعات نتيجة لتطور الإلكترونيات الدقيقة Microelectronics بشكل كبير ، فظهرت أجيال متعاقبة من الحواسيب ، وكان آخرها حواسيب الجيل الخامس ، والتي تميزت باستخدام كبير للدارات المتكاملة الفائقة. إن كلمة computer مشتقة من الفعل compute ويعني " أحسب" ، وإن كلمة computer تعني " حاسب" ، ولقد قام مجمع اللغة العربية بتعريب هذه الكلمة اصطلاحا فأطلق عليها لفظ "حاسوب"، وفيما يتعلق بالتعريف العلمي للحاسوب فيمكن أن يعرف بعدة أشكال.

١-٢- تعريف الحاسوب

يمكن تعريف الحاسوب(computer) بعدة أشكال كما يلي :

- الحاسوب آلة الكترونية قابلة للبرمجة تقوم باستقبال المعطيات التي يتم إدخالها بوساطة وحدات الإدخال حيث يتم تخزينها وإجراء العمليات الحسابية والمنطقية عليها بوساطة التعليمات المكونة للبرامج وذلك للحصول على النتائج وإخراجها من وحدات الإخراج المختلفة.

- الحاسوب آلة الكترونية يمكن برمجتها لكي تقوم بمعالجة المعطيات وتخزينها واسترجاعها وإجراء العمليات الحسابية والمنطقية عليها .
- الحاسوب آلة حساب الكترونية سريعة تستقبل معلومات دخل رقمية وفقاً لمنهجية تحددها تعليمات برمجية مخزنة ثم تصدر نتائج المعالجة إلى الوسط الخارجي .
- الحاسوب مجموعة من الأجهزة أو الوحدات المستقلة Hardware يؤدي كل منها وظيفة معينة وتعمل هذه الوحدات فيما بينها بأسلوب متناسق من خلال البرمجيات Software , ويوضح الشكل(١-١) نظام الحاسوب



الشكل(١-١) – يوضح نظام الحاسوب

ويتميز الحاسوب بالخواص التالية :

١- القدرة على تخزين المعطيات واسترجاعها : كالأرقام ، والحروف الهجائية ، والصور .

٢- إمكانية معالجة المعطيات وإجراء العمليات الحسابية عليها : كالجمع ، والطرح ، والقسمة , والضرب ، وإجراء العمليات المنطقية كالمقارنة بين قيمها .

٣- إمكانية برمجة الحاسوب أي إعطاء تعليمات وأوامر للحاسوب لكي يقوم بتنفيذ أعمال محددة .

١-٣ - المكونات الأساسية للحاسوب

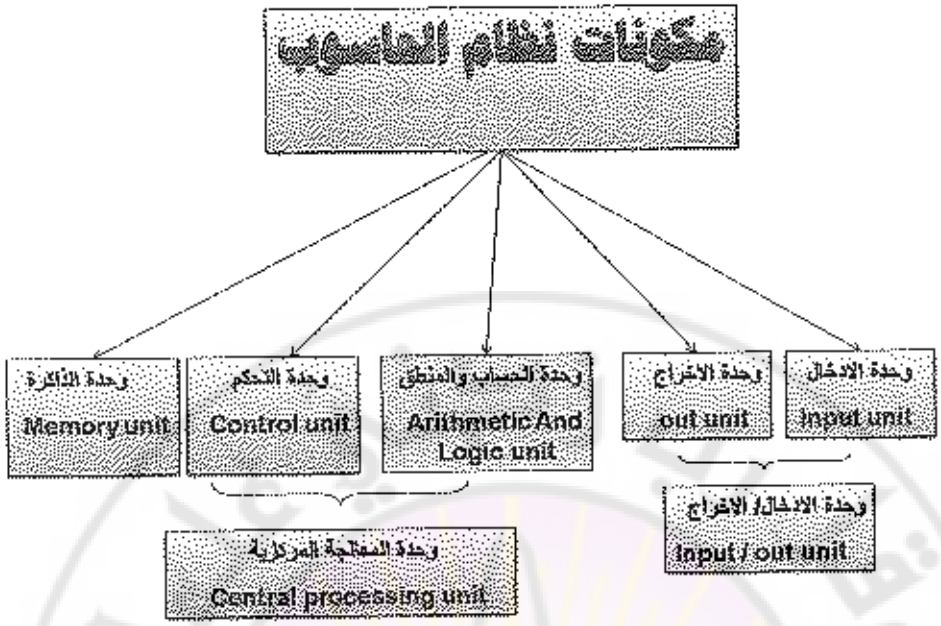
يتكون النظام الحاسوبي من قسمين رئيسين هما :

- العتاد (Hardware) .
- البرمجيات (Software) .

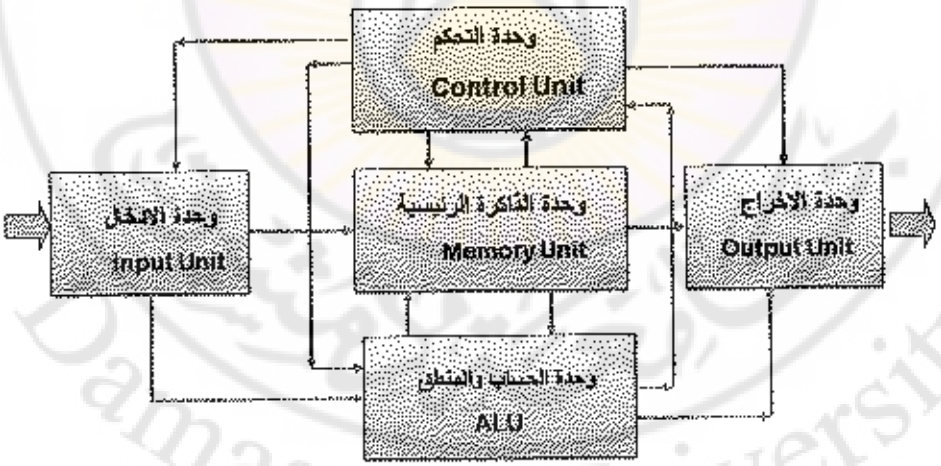
أما المكونات الأساسية لنظام الحاسوب حسب " **فان نيومان**" فتتكون من خمس وحدات أساسية هي :

١. وحدة الإدخال
٢. وحدة الإخراج
٣. وحدة الحساب والمنطق
٤. وحدة التحكم
٥. وحدة الذاكرة الرئيسية

ويوضح الشكل (١-٢) المكونات الأساسية للحاسوب , أما الشكل (١-٣) يوضح المخطط الصندوقي لمكونات الحاسوب ومبدأ عمله .



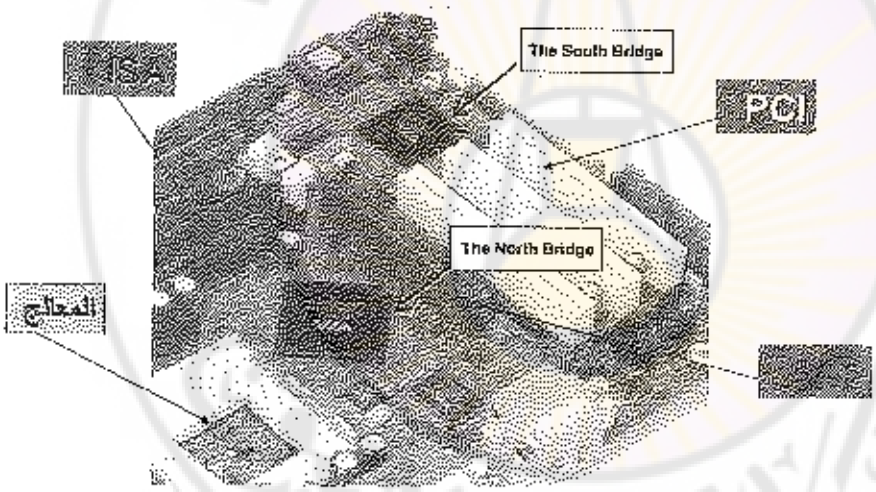
الشكل (١-٢)- يوضح المكونات الأساسية للحاسوب



الشكل (١-٣) – يوضح المخطط الصندوقي لمكونات الحاسوب ومبدأ عمله

١-٤- لوحة النظام (System Board)

وتسمى أيضاً باللوحة الأم (Mother Board) ، وهي اللوحة الأساسية ، وتقع داخل صندوق النظام وتتكون من مجموعة كبيرة من القطع الإلكترونية المثبتة عليها ، تصل بها جميع أجزاء الحاسوب ، ويتم تغذيتها بالطاقة الكهربائية (Power Supply) ، ويطلق على كل قطعة من هذه القطع الإلكترونية بالدارة المتكاملة ، ويتم تصنيع هذه الدارات المتكاملة عن طريق وضع عدد كبير من الدارات الإلكترونية داخل حجم صغير من مادة السيلكون ، والتي تحوي في داخلها على الدارات الإلكترونية أو الكهربائية كالترانزستورات والمقاومات والمكثفات وغيرها ، ويوضح الشكل (١-٤) اللوحة الأم .



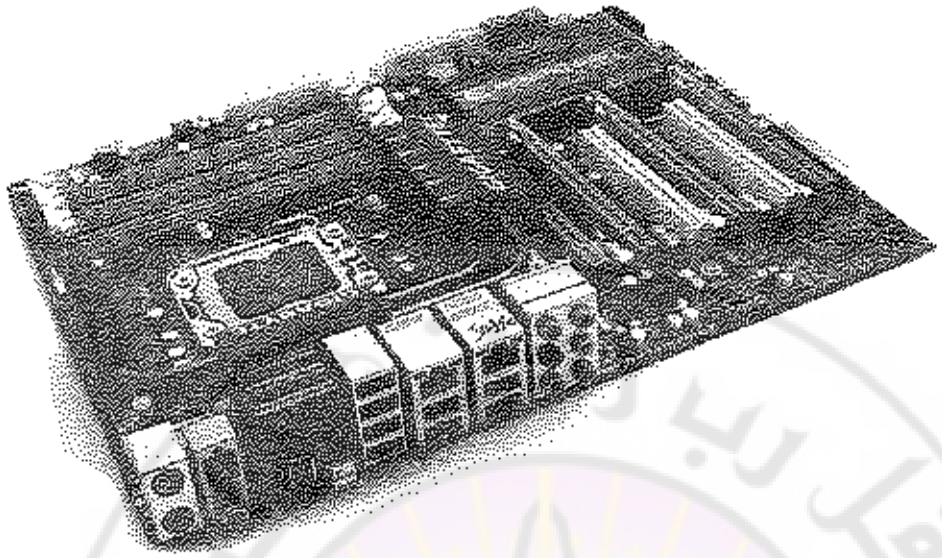
الشكل (١-٤) - يوضح مكونات اللوحة الأم

وتتكون لوحة النظام من الوحدات التالية :

- وحدة المعالجة المركزية Central Processing Unit (CPU).
- وحدة الذاكرة الرئيسية Main Memory Unit.
- وحدة الترابط مع الحاسوب (Interface Unit) أو المواجهة .

١ – ٤ – ١ – وحدة المعالجة المركزية (CPU)

إن وحدة المعالجة المركزية بمثابة العمود الفقري للحاسوب الشخصي ، و بدون هذه الوحدة يكون الجهاز مجرد آلة صماء غير قادرة على القيام بأي عمل مفيد ، ويطلق على هذه الوحدة في الحاسوب الشخصي بالمعالج الصغري (Micro-processor) ، وهو عبارة عن دارة الكترونية عالية التكامل بمعالجة المعطيات وبتنفيذ أوامر البرمجة التي ترسل إليها من الدارات الالكترونية الأخرى في لوحة النظام . لذلك فإن هذه الوحدة هي وحدة التشغيل المركزية ، وهي الوحدة التي تتحكم في نظام الحاسوب في كل عملياته، والجزء الأكثر أهمية لاحتوائها على جميع الإمكانيات الضرورية لإنجاز مهام المعالجة وتداول المعطيات، بالإضافة إلى رقابة وتوجيه جميع الوحدات والأجهزة الأخرى وتنسيق العمل بينها ، ويبين الشكل (١–٥) وحدة المعالجة المركزية CPU .



الشكل (١-٥) - يوضح وحدة المعالجة CPU

ومن المهام الأساسية التي تقوم بها وحدة المعالجة CPU كما يلي:

- تقوم وحدة المعالجة المركزية بتنفيذ جميع العمليات الخاصة بالتشغيل .
- تقوم بعمليات المقارنة المنطقية والحسابية التي تكون موجودة في البرنامج المراد تنفيذه .
- تقوم بتنظيم نقل المعطيات من/و إلى الوحدات المساعدة حيث تتم العملية باستقبال المعطيات وإرسال تلك المعطيات إلى وحدات محددة كما هو مطلوب وذلك في الوقت المناسب.
- تقوم بتمرير المعطيات من/و إلى الذاكرة الرئيسية Main Memory.

وتتكون وحدة المعالجة المركزية من :

- وحدة التحكم Control Unit.
- وحدة الحساب والمنطق Arithmetic and Logic Unit.
- المسجلات الداخلية Registers .

١ – وحدة التحكم Control Unit

إن عمل وحدة التحكم مهم بالنسبة لوحدة المعالجة المركزية و لما تقوم به من تنسيق جميع أعمال الحاسوب ، فتقوم بمهام الإشراف والتحكم والتوجيه والضبط لجميع العمليات التي تقوم بها وحدات الحاسوب الداخلية، وتقوم بمراقبة وتوجيه جميع الوحدات الأخرى وتعمل على توليد النبضات الزمنية اللازمة لتحقيق تسلسل تنفيذ العمليات ، وتقوم هذه الوحدة بتنظيم وتنسيق عمل الوحدات في الحاسوب دون أي خلل وبأسلوب متناسق ومنتظم . وتتكون هذه الوحدة من الوحدات الجزئية التالية :

- وحدة العنونة Address Unit
- وحدة التعليمات Instruction Unit
- وحدة التوقيت والتتابع الزمني Timing Unit

وتقوم هذه الوحدات الجزئية بالمهام التالية:

١. تقوم بتحديد عنوان التعليمة المراد تنفيذها ويتم وضع هذا العنوان (عنوان بداية البرنامج) على مسرى العناوين Address Bus .
٢. تقوم بجلب محتوى الذاكرة حسب العنوان المذكور ووضعه على مسرى المعطيات Data Bus
٣. تقوم بتفسير التعليمة من أجل إصدار الأوامر المناسبة لتنفيذها في وحدة ALU
٤. تقوم بالإشراف على تخزين النتائج في أماكن تخزينها.

٥. يتم زيادة عداد البرنامج PC بمقدار مساوٍ لطول التعليمة وذلك للانتقال إلى عنوان التعليمة التالية في البرنامج الذي يجري تنفيذه ، وفيما بعد يتم تكرار العمليات نفسها المذكورة أعلاه على جميع تعليمات البرنامج.

أما المهام والوظائف التي تقوم بها وحدة التحكم بشكل عام :

- تأهيل وحدات الإدخال والإخراج المناسبة حين الحاجة إليها.
- قراءة وتفسير تعليمات البرنامج ومن ثم إصدار الإشارات اللازمة للوحدات المختلفة لتنفيذ هذه التعليمات.
- التحكم بتدفق المعطيات والتعليمات من و إلى الذاكرة الرئيسية ومتحكمات وحدات الإدخال والإخراج .
- توجيه العمليات داخل وحدة المعالجة المركزية CPU .
- تنظيم وتنسيق عمل الوحدات في الحاسوب .

٢- وحدة الحساب والمنطق Arithmetic and Logic Unit

هي الوحدة التي تقوم بتنفيذ العمليات الحسابية والمنطقية ، حيث تقوم باستقبال المعطيات إما من وحدة الإدخال مباشرة لمعالجتها أو من وحدة الذاكرة الرئيسية وفيما بعد يتم إرسال نتائج المعالجة إلى وحدة الذاكرة الرئيسية لتخزينها أو إرسالها إلى وحدات الإخراج لإظهارها . وتتكون هذه الوحدة من مجموعة من السجلات المساعدة وذلك لحفظ العناوين والنتائج الجزئية ، وتتأثر وحدة الحساب والمنطق بسجل المؤشرات Flags الذي توضع فيه مؤشرات تصف خرج وحدة الحساب والمنطق ، وتقوم هذه الوحدة بالعمليات التالية :

- بجميع العمليات الحسابية (الجمع – الطرح – الضرب – القسمة)
- بجميع العمليات المنطقية (AND – OR – EX OR .....الخ)
- بعمليات المقارنة والنقل بين المسجلات

وتحتوي وحدة الحساب والمنطق على عدد كبير من السجلات Registers و عدد من دارات الجامع Adder وعدد من العدادات Counters ، ومن مهام الجامع Adder القيام بتنفيذ جميع العمليات الحسابية المختلفة (+ ، - ، * ، /) وكذلك عمليات المقارنة (>,<=,<,<=,==) وذلك عندما تكون المعطيات التي يتعامل معها بالشكل الثنائيBinary ، وتقوم هذه الوحدة بالمعالجة الفعلية للمعطيات تحت اشراف وحدة التحكم ، حيث تنتقل المعطيات أثناء دورة التنفيذ إلى أحد العدادات أو إلى أكثر من عداد وهناك يتم تداول المعطيات بوساطة دارات الجامع لكي يتم استخراج النتائج التي يمكن تخزينها بإحدى العدادات لجمع أو تحويل هذه النتائج إلى أمكنة أخرى بوحدة التخزين.

٣- السجلات الداخلية Registers

عبارة عن مواقع تخزين خاصة عالية السرعة ، تخزن المعطيات و المعلومات بشكل مؤقت لاستخدامها من قبل وحدة الحساب والمنطق ALU. وتحتوي وحدة المعالجة على أنواع مختلفة من السجلات ، كل منها مختص بتخزين نوع معين من المعطيات. وتتكون من سجلات للأغراض العامة GBR يستخدمها المعالج من أجل المتحولات والقيم التي يعالجها، وسجلات للأغراض الخاصة SBR كسجل عداد البرنامج PC ، وسجل البرنامج الذي يحتفظ بقيمة عنوان الذاكرة ومنها سجل أساسي يسمى المراكم Accumulator، ويتم دمج الأقسام الثلاثة المكونة لوحدة المعالجة المركزية جميعها في دارة متكاملة واحدة.

١ – ٤ – ٢ – وحدة الذاكرة الرئيسية Main Memory Unit

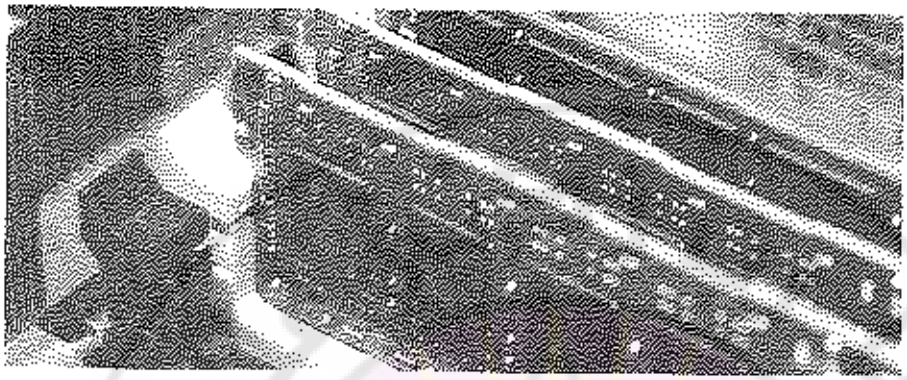
تعد وحدة الذاكرة الرئيسية من الأجزاء الرئيسية بجهاز الحاسوب ، وتتكون من مجموعة من الدارات الالكترونية المثبتة على لوحة نظام الحاسوب ، وتقوم بمهام الاحتفاظ بالمعطيات والأوامر التي يحتاجها المعالج عند إجراء العمليات المختلفة ، وإرسالها للمعالج عند طلبها ، وتحتفظ بالمعطيات الأساسية المطلوبة لتشغيل جهاز الحاسوب أو المتطلبات اللازمة لعمل نظام التشغيل ، ويتم في هذه الوحدة تخزين النتائج الوسيطية والنهائية لكل من المعطيات والتعليمات و تخزن أيضاً البرامج ليتم تنفيذها فيما بعد ، وترتبط هذه الوحدة مباشرة بوحدة المعالجة المركزية (CPU) وتتألف من مجموعة من المواقع وكل موقع مكون من عدد من الخلايا الثنائية (bits) وهي مواقع لتخزين البرامج والمعطيات التي يتم معالجتها .

وتتميز الذاكرات بالإضافة إلى سعتها وطول كلمتها بمقدارين هما زمن النفاذ Access Time الذي يُعرف بأنه الفاصل الزمني ما بين لحظة تطبيق عنوان الموقع المراد قراءته ولحظة ظهور هذا المحتوى على خطوط المعطيات ، وهذا الزمن ثابت بالنسبة لكل مواقع الذاكرة . وزمن الكتابة Write Time الذي يُعرف على أنه الزمن اللازم لكتابة المعطيات في أي موقع في الذاكرة بعد وضع العنوان على خطوط العنونة . وفيما يلي سنستعرض بعض أنواع الذاكرات :

١ – ذاكرة الوصول العشوائي Random Access Memory(RAM)

هي دارة الكترونية متكاملة تستخدم للاحتفاظ بالمعلومات مؤقتاً أثناء تشغيل الحاسوب ، ولا تحتفظ بهذه المعلومات عند قطع التيار الكهربائي عن الحاسوب . وهذا الجزء يكون تحت تصرف مستخدم الحاسوب حيث تحتفظ بالبرامج والمعطيات التي يدخلها مستخدم الحاسوب أثناء التشغيل ، أو يتم إخراجها لوحدات

الإخراج والتخزين ، و كلما زادت سعة الذاكرة RAM أمكن للحاسوب التعامل مع برامج كبيرة وحجم أكبر من المعطيات، ويبين الشكل (١-٦) الذاكرة RAM .



الشكل (١-٦) – يوضح الذاكرة RAM

تخزن في الذاكرة RAM البرامج والمعطيات التي يدخلها المستخدم ، وكذلك نتائج معالجة المعطيات حسب البرامج المدخلة . وتتميز الذاكرة RAM بما يلي :

- ✓ تُعد ذاكرة قراءة وكتابة .
- ✓ تُعد ذاكرة قراءة وكتابة بها لأكثر من مرة.
- ✓ تُعد ذاكرة مؤقتة لأنها لا تحتفظ بالمعلومات بعد انقطاع التغذية عنها .
- ✓ تستخدم لتخزين البرامج والمعلومات التي يحتاجها المستثمر أثناء العمل على الحاسوب .
- ✓ ذات حجم كبير تصل إلى 16GB أو أكثر .

ثمة نوعان من الذاكرة RAM:

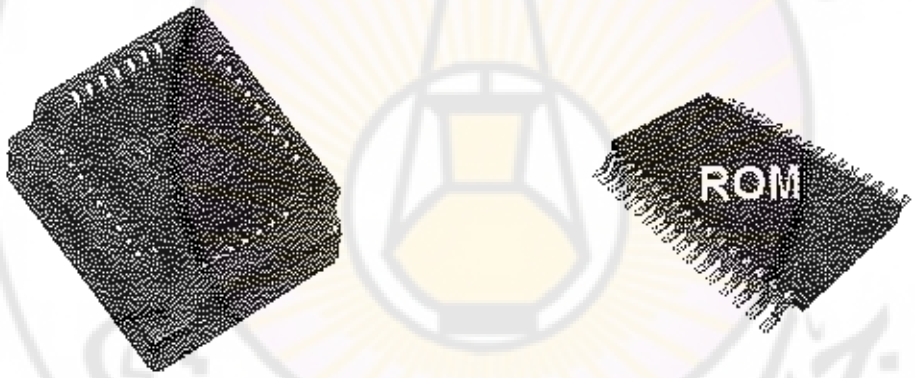
١- ذاكرة ساكنة Static RAM ويرمز إليها بـ SRAM

٢- ذاكرات ديناميكية Dynamic RAM ويرمز إليها بـ DRAM .

الذاكرات الديناميكية وهي أرخص من الذاكرات الساكنة ، وذات زمن نفاذ أعلى نسبياً ، والذاكرة الساكنة تحافظ على محتواها مادامت هناك تغذية كهربائية مطبقة عليها ، بينما تفقد الذاكرة الديناميكية محتواها بعد فترة قصيرة ، وتحتاج لإعادة إنعاش للاحتفاظ بالمعطيات(Refresh) .

٢- الذاكرة ROM (Read Only Memory)

تقوم الذاكرة ROM بالاحتفاظ الدائم بالمعلومات ، ومن خلالها يتم الاحتفاظ بالمعلومات الأساسية التي يحتاجها الحاسوب لبدء تشغيله ، مثل موقع وجود نظام التشغيل وعناوين بوابات الإدخال والإخراج وغيرها. ويتم تخزين البرامج عليها من قبل الشركة الصانعة وبذلك وتكون ضمن وحدات جهاز الحاسوب ، ويبين الشكل (١-٧) الذاكرة ROM .



الشكل (١-٧) – يبين الذاكرة ROM

وتتميز الذاكرة ROM بما يلي :

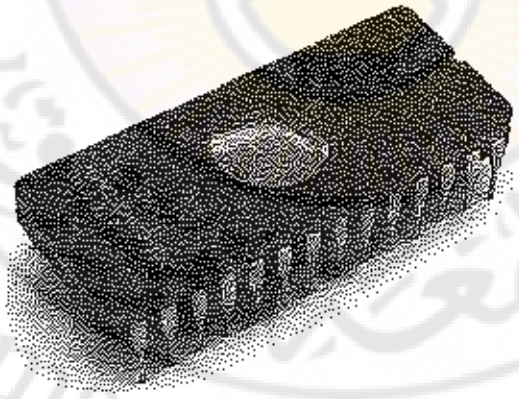
- ✓ تُعد ذاكرة قراءة فقط .
- ✓ يتم القراءة منها لأكثر من مرة ولا يمكن التعديل عليها أو الإضافة أو الحذف منها كليا أو جزئيا أو الكتابة عليها .
- ✓ تُعد ذاكرة دائمة ، لأنها تحتفظ بالمعلومات بعد انقطاع التغذية عنها .

- ✓ سعتها صغيرة مقارنة مع الذاكرة RAM .
- ✓ تستخدم هذه الذاكرة لتخزين البرامج والمعلومات الثابتة مثل برنامج إقلاع الحاسوب.

٣- الذاكرة PROM (Programmable ROM)

وهي ذاكرة مشابهة للذاكرة ROM ولكن يتم برمجتها من قبل المستخدم لمرة واحدة فقط ، ويتم الكتابة عليها ، بعد تصنيعها وذلك باستخدام جهاز مخصص لبرمجة ذاكرات PROM ويبين الشكل(١-٨) الذاكرة PROM، وتتميز الذاكرة PROM بما يلي :

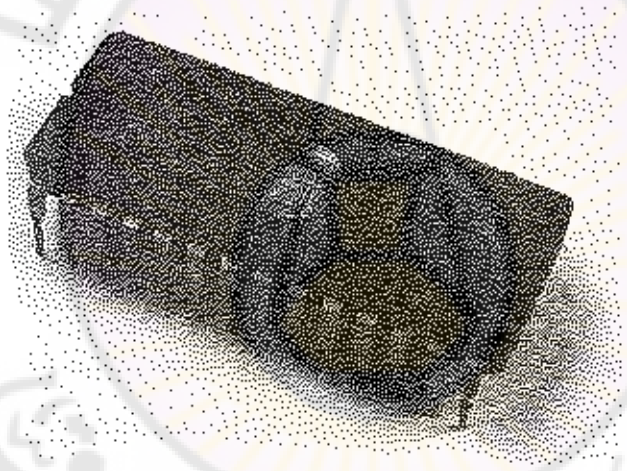
- ✓ تُعد ذاكرة دائمة .
- ✓ يمكن برمجتها من قبل المستثمر ولمرة واحدة فقط .
- ✓ تُعد ذاكرة قراءة لأكثر من مرة .
- ✓ لا يمكن إعادة برمجتها مرة أخرى أو تعديل البرامج عليها .



الشكل (١-٨) – يبين الذاكرة PROM

٤- الذاكرة EPROM (Erasable PROM)

وهي ذاكرة مبرمجة وقابلة للمحو ، ويستطيع المستخدم محو (إزالة) محتواها وإعادة برمجتها ، وهي ذاكرات يمكن إعادة الكتابة فيها بعد محوها ، ويتطلب هذا النوع من الذاكرات جهاز خاص لبرمجتها يسمى مبرمج EPROM ، ويتم محو محتواها بالأشعة ما فوق البنفسجية (Ultraviolet Light) ، ولا يمكن تغيير محتواها أو برمجتها خلال استخدامها في النظام ، ولا يمكن محو كلمة واحدة فقط، وبعد محو الذاكرة تكون جميع خلايا التخزين بحالة (1) منطقي ، وعند البرمجة يتم فقط تخزين (0) منطقي في كل خلية من خلايا الكلمة المراد برمجتها ، ويبين الشكل (١-٩) الذاكرة EPROM.



الشكل (١-٩) – يبين الذاكرة EPROM

ومن أهم التطبيقات ومجالات استخدام الذاكرات EPROM يكون في تطوير نظم الحواسيب الميكروية ، حيث يتم وضع برامج النظام في ذاكرات EPROM للتأكد من صحة عمل النظام أو أجراء التعديلات المناسبة ، وبعد ذلك يمكن نقل هذا البرنامج إلى ذاكرات (ROM) وذلك لتوفر مثل هذه الذاكرات لها الأرجل

نفسها للذاكرات EPROM . إن وجود ذاكرات من النوع ROM مكافئة للذاكرات من النوع EPROM يعطي منتجي نظم الحواسيب الميكروية إمكانية استخدام ذاكرات من النوع ROM والتي تشكل تكاليفها عند الإنتاج بكميات كبيرة ضئيلة بالنسبة لتكاليف الذاكرات من نوع EPROM .

٥ – ذاكرة الكاش Cache Memory

هي ذاكرة تتصل بالمعالج (وحدة المعالجة المركزية CPU) وتتميز بسرعة عالية جداً وتخزن عليها المعطيات والبرامج المستخدمة بكثرة من قبل المستخدم ما يوفر الزمن عند استدعائها من الذاكرة الرئيسية و بالتالي زيادة الإنتاجية وعادة ما تكون هذه الذاكرة بسعات مختلفة تصل إلى 1GB أو أكثر .

ومبدأ عمل ذاكرة الكاش يقوم على أنه يتم استقبال طلب قراءة Read من وحدة المعالجة المركزية CPU ، ويتم نقل كتلة(Blok) من كلمات الذاكرة التي تحتوي على الموقع المحدد إلى ذاكرة الكاش وبالتالي يتم قراءة محتويات هذه المواقع مباشرة منها ، ويتم تحديد العلاقة بين كتل الذاكرة الرئيسية وذاكرة الكاش بتابع الإسقاط Mapping Function.

– عندما تكون ذاكرة الكاش ممتلئة ويتم طلب كلمة ذاكرة غير موجودة فيها تقرر دارات تحكم الكاش أي كتلة يجب إزالتها من أجل وضع الكتلة الجديدة التي تحتوي على الكلمة المطلوبة ، وتسمى مجموعة القواعد التي تقوم باتخاذ هذا القرار بخوارزمية الاستبدال Replacement Algorithm .

وعندما لا تكون الكلمة المعنونة في عملية القراءة Read موجودة في الكاش يتم نسخ كتلة من الكلمات التي تحتويها من الذاكرة الرئيسية إلى الكاش ، ثم ترسل الكلمة المطلوبة إلى وحدة المعالجة المركزية CPU ، وكلما زاد حجم ذاكرة الكاش

أمكن تزويدها بالمزيد من المعطيات التي يحتاجها المعالج وبالتالي تقليل حدوث حالات انتظار أثناء تشغيل البرامج .

<u>من ميزاتها</u>

- ✓ متصلة بالمعالج CPU
- ✓ سرعتها عالية مقارنة بذاكرة RAM
- ✓ تقسم الذاكرة إلى مجموعة كتل Blocks مما يجعل تخزين وحذف المعطيات أسرع من الـ RAM إلا أنها أغلى ثمناً.
- ✓ تخزن البرمجيات و المعطيات المستخدمة بكثرة وبالتالي يسرع عمل الحاسوب
- ✓ سعتها عادة: 128 KB, 256 KB, 512KB, 1 MB, أو أكثر .

٦- ذاكرة الـ Flash (Flash Memory)

الذاكرة Flash نوع من أنواع الذاكرة الدائمة (غير المتطايرة) مثل ROM إلا أنها تخزن المعطيات في مجموعة كتل Blocks ، ويتم التخزين و المسح في الكتلة بحركة واحدة تدعى Flash ما يجعلها أسرع من الذاكرة RAM إلا أنها أغلى ثمناً. وتستخدم الآن في تخزين نظام الإدخال /الإخراج الأساسي الخاص بالحاسوب BIOS وهو عبارة عن برنامج يتم تحميله عند تشغيل الحاسوب للتعرف على وحدات الإدخال والإخراج المرتبطة معه. كما تستخدم أيضاً في الحواسيب المحمولة و الطابعات و الكاميرات الرقمية، والأجهزة الخليوية.

<u>من ميزاتها</u>

- ✓ ذاكرة دائمة
- ✓ تستخدم في تخزين نظام الـ BIOS

- ✓ تقسم الذاكرة إلى مجموعة كتل Blocks ما يجعل تخزين وحذف المعطيات أسرع من الـ RAM إلا أنها أغلى ثمناً.
- ✓ تستخدم في الحواسيب المحمولة و الكاميرات الرقمية و الأجهزة الخلوية.

١-٥- الوحدات المحيطية لنظام الحاسوب

Computer Peripheral

وهي عبارة عن أجهزة يتم ربطها بجهاز الحاسوب عبر بطاقات التوسعة وبوابات الحاسوب ، وتُعد صلة وصل بين مستخدمي الجهاز ولوحة النظام ، حيث يقوم المستخدم بالتعامل مع لوحة النظام من خلال الوحدات المحيطية ، وتشمل الوحدات المحيطية لنظام الحاسوب ثلاث وحدات هي :

- وحدات الإدخال .
- وحدات الإخراج .
- وحدات التخزين الثانوية .

١-٥-١- وحدات الإدخال Input Units

وهي الوحدات التي يتم عن طريقها إدخال المعطيات إلى جهاز الحاسوب ، وتستقبل وحدة الإدخال المعلومات المرمزة (معطيات Data أو التعليمات Instruction) من الأجهزة الخارجية للإدخال وترسلها إلى وحدة الذاكرة الرئيسية ليتم حفظها ولتعالج فيما بعد ، أو ترسل إلى وحدة الحساب والمنطق ALU لتعالج مباشرة ، ونتيجة المعالجة ترسل إلى وحدة الذاكرة لحفظها ، أو ترسل إلى وحدة الإخراج لإظهارها .

وهي وسيلة اتصال بين المستخدم والحاسوب ، حيث تقوم هذه الوحدات باستقبال المعطيات والتعليمات وإدخالها إلى الحاسوب، حيث إنها تقوم بعملية

توصيل المعلومات من لغة الإنسان المكونة من الأرقام والحروف إلى لغة الآلة المكونة من نبضات كهربائية (وجود نبضة كهربائية تمثل بـ 1 منطقي وعدم وجود نبضة كهربائية تمثل بـ 0 منطقي)، ومن أهم وحدات الإدخال يمكن أن نذكر:

- لوحة المفاتيح (Keyboard) .
- الفأرة (Mouse) .
- الماسح الضوئي (Scanner) .
- الكاميرا الرقمية (Digital Camera) .
- جهاز القلم الضوئي (Light Pen) .
- شاشة اللمس (Touch Screen) .
- السبورة الإلكترونية (Electronic Board) .
- عصا التحكم بالألعاب (Joy stick) .
- جهاز اللاقط (الميكروفون)(Microphone) .
- جهاز قارئ الأعمدة (Bar Code Reader) .
- بطاقات الاتصال (Telecommunication Cards) .
- الفاكس Fax .

١-٥-٢- وحدات الإخراج Output Units

وظيفتها إخراج المعطيات على شكل معلومات من الحاسوب بعد معالجتها ، ويتم في هذه الوحدة إظهار نتائج المعالجة من وحدة الحساب والمنطق أو إخراج وإظهار المعلومات من وحدة الذاكرة عبر أجهزة الإخراج ، وأهمها الشاشة ، التي توضح للمستخدم ماذا تفعل وحدة المعالجة وتظهر نتائج المعالجة ولذلك تسمى بالمُراقب Monitor ، ومن أهم وحدات الإخراج يمكن أن نذكر:

- الشاشة (Screen) .
- الطابعة (Printer).
- الراسمة (Plotter).
- السماعات الصوتية (Speakers).

١-٥-٣- وحدات التخزين الثانوية

Secondary Storage Units

يقوم جهاز الحاسوب بتخزين المعطيات في مكانين داخل الحاسوب ، وذلك بناء على مدى الحاجة لها من قبل وحدة المعالجة المركزية ، فإذا استخدمت المعطيات بشكل فوري من قبل وحدة المعالجة المركزية ، يتم الاحتفاظ بها في وحدة الذاكرة الرئيسية الموجودة على لوحة النظام ، وإذا كانت المعطيات لا تطلبها وحدة المعالجة المركزية بشكل فوري فانه يتم تخزينها في وحدات التخزين الثانوية بالجهاز والتي تكون داخل صندوق وحدة النظام أو خارجه وتتصل به عن طريق بوابة مناسبة لذلك .

وهي ذاكرات ذات سعات عالية إضافية للحاسوب ويمكن حفظ المعطيات والبرامج بصورة دائمة عليها لتستخدم عند الحاجة ، وهذا النوع يقع خارج وحدة المعالجة المركزية ويطلق على هذه الوحدات اسم وحدات التخزين الخارجية External Storage Units ، وتتميز الذاكرة عن وسائل وحدات التخزين الثانوية الملحقة بالجهاز بالأمور التالية :

١-إن الذاكرة هي دارة الكترونية بعكس وحدات التخزين الثانوية الملحقة بالجهاز ، والتي تتكون من أجهزة ميكانيكية المعتمدة على تقنية الكهرومغناطيسية أو على التقنية الضوئية .

٢-إن الذاكرة تتصل مباشرة بالمعالج على لوحة النظام ما يسمح بتمرير المعطيات بين المعالج والذاكرة بسرعة عالية (سرعة مرور النبضات الكهربائية) ، بينما تتصل وحدات التخزين الثانوية الملحقة بالجهاز بالمعالج عن طريق مسار خاص يسمى المسرى وعبر بوابات الجهاز ، ما يحتاج إلى زمن أطول بكثير لإمرار المعطيات . ويبين الشكل (١- ١٠) بعض وحدات التخزين الثانوية .

الشكل (١- ١٠) – يبين بعض وحدات التخزين الثانوية

وهذه الفروق بين الذاكرة ووحدات التخزين الثانوية ، لذلك يطلق أحياناً على الذاكرة الرئيسية تسمية وحدة التخزين الرئيسية ، بينما يطلق على أجهزة التخزين تسمية وحدات التخزين الثانوية . وتتميز وحدات التخزين الثانوية بما يلي :

١. ذات تكلفة صغيرة (رخيصة الثمن) .

٢. ذات سعة أكبر بكثير من الذاكرة الرئيسية .

٣. ذات سرعة وصول للمعلومات أصغر بكثير من الذاكرة الرئيسية .

٤. يوجد منها عدة أنواع ، لكل منها مشغل خاص .

ويمكن قياس القدرة التخزينية لوحدات التخزين وكذلك لحجم المعطيات بداخلها بواسطة وحدات القياس التالية :

1. **Byte** = 8 bit

2. **Kbyte** = 1024 Byte

3. **Mbyte** = 1024 K byte = 1024* 1024 Byte

4. **Gbyte** = 1024 M byte = 1024 * 1024 * 1024 Byte

5. **Tbyte** = 1024 Gbyte = 1024 * 1024 * 1024 * 1024 Byte

6. **Bbyte** = 1024 Tbyte = 1024 * 1024 * 1024 * 1024 * 1024 Byte

7. **Ebyte** = 1024 Bbyte = 1024 * 1024 * 1024 * 1024 * 1024 * 1024 Byte

١ – ٦ – أداء الحاسوب Computer Performance

يقصد هنا سرعة تنفيذ التعليمات لوحدة المعالجة المركزية CPU ، وتتحدد هذه السرعة بعدة عوامل :

١. تردد ساعة الحاسوب Clock Speed

٢. حجم الذاكرة الرئيسية Main Memory

٣. القرص الصلب Hard Disk

٤. المسرى Bus Speed

٥. بطاقة الشاشة Graphics Acceleration Card

أما الميزات التي تميز معالج عن معالج آخر هي :

١- سرعة وحدة المعالجة CPU speed

يزود كل حاسوب بمولد للنبضات يغذي المعالج بنبضات كهربائية منتظمة تتحكم في تزامن تنفيذ التعليمات تسمى نبضات الساعة، وهي التي تحدد سرعة المعالجة أي عدد العمليات المنفذة في الثانية ، وكلما زادت سرعة المعالج أمكن تنفيذ عدد أكبر من العمليات داخل جهاز الحاسوب بالثانية الواحدة (سرعة معالجة الحاسوب) وتقاس سرعة المعالجة بالهرتز Hz (أي عدد النبضات في الثانية الواحدة) .

٢- مسرى المعطيات Data Bus

ويحدد مسرى المعطيات أكبر عدد من الأرقام الثنائية التي يقوم المعالج بالتعامل معها في كل عملية ، وتقاس بالبت مثلاً: 8bit, 16bit,32bit,64bit وكلما كان عدد الخطوط أكبر، كان المعالج أسرع في جلب المعلومات ، ويزداد طول هذا المسرى كلما تطور الحاسوب .

٣- مسرى العناوين Address Bus

ويحدد مسرى العناوين أكبر سعة ذاكرة يمكن التعامل معها وتقاس بالبت 16bit,32bit,64bit . ويزداد طول هذا المسرى كلما تطور الحاسوب.

٤- عرض الكلمة Word size

ويحدد عرض الكلمة أكبر عدد من الأرقام الثنائية ، ويمكن التعامل معه بعملية واحدة ، أو طول السجل وتقاس أيضا بالبت .

٥- معالج مساعد داخلي Internal CO-processor

يستخدم هذا المعالج المساعد للقيام بالعمليات الحسابية المعقدة مثل الفاصلة العائمة).

٦- ذاكرة كاش داخلية Internal cache memory

وهي ذاكرة ساكنة SRAM ذات زمن نفاذ قليل ، أسرع بكثير من الذاكرة الرئيسية وتوضع لتسريع المعالجات .

وتوجد معالجات من صنع شركات متخصصة متعددة ، ويختلف تصميمها حسب هذه الشركات ، ومن الأسماء المشهورة لهذه المعالجات 80X86 و Pentium من إنتاج شركة إنتل Intel ، ومعالجات شركة AMD ، ومعالجات RISC المستخدمة في أجهزة أبل وماكنتوش .

١-٧- البرمجيات Software

البرمجيات عبارة عن مجموعة من الأوامر المرتبة منطقياً ،و يتم تنفيذها بوساطة وحدة المعالجة المركزية للحاسوب ، والتي تكون مخزنة على شكل ملفات في وحدات التخزين الثانوية ، وهذه البرامج تقوم بالإشراف على المكونات الأساسية للحاسوب كافة ، وتشرف على تسلسل تنفيذ عمليات المعالجة المختلفة ، فالتطور الذي يحدث في أجهزة الحاسوب ومكوناته يصاحبه أيضاً تطور وتحديث دائم في عالم البرمجيات .

يمكن أن نعرف البرمجيات كما يلي :

البرمجيات اصطلاح يطلق على جميع البرامج اللازمة لتشغيل الحاسوب وتنظيم عمل وحداته وكذلك تنسيق العلاقة بين هذه الوحدات .

ويشمل هذا التعريف نظم التشغيل وكذلك البرامج المعيارية التي يقوم مصنعو الحاسوب بإعدادها والتي تمكن المستخدمين من استغلال عمل الحاسوب على

أفضل وجه ، وكذلك يشمل هذا التعريف على البرامج التطبيقية التي تلزم لاستخدام الحاسوب . وتضم هذه البرمجيات ما يلي:

١ – برمجيات نظم التشغيل Operating Systems

نظام التشغيل برنامج حاسوبي يقوم بضبط وإدارة التحكم بجميع الوحدات الأساسية للحاسوب(المكونات العتادية + وحدات التخزين الثانوية) ، ولا يمكن الاستغناء عن نظام التشغيل في أي نظام حاسوبي، وبصورة أدق فإن الحاسوب لا يعمل على الإطلاق إذا لم يكن مزوداً بنظام تشغيل ، و برمجيات ضرورية لتشغيل الحاسوب وتنظيم عمل الوحدات المحيطية به , مثل :

Windows(XP,7, 10, NT) , Unix , DOS , Linux

٢ – البرمجيات التطبيقية Application Software

وهي مجموعة من البرامج الجاهزة يتم كتابتها بإحدى لغات البرمجة لتشكل بيئة يستطيع من خلالها المستخدم التعامل مع الحاسوب ليقوم بإجراء عمليات معالجة المعطيات وهذه البرمجيات تساعد المستخدم على أداء الأعمال اليومية بكل يسر وسهولة ، نذكر منها :

أ– البرمجيات التطبيقية الجاهزة (Package Software)

وهي مجموعة من البرمجيات المتخصصة والجاهزة التي يمكن أن يستخدمها أي شخص لتنفيذ عمليات محددة على المعطيات، ونذكر منها :

- برمجيات معالجة النصوص Word.
- برمجيات معالجة الجداول الإلكترونية Excel .
- برمجيات التصميم AutoCAD .
- برمجيات قواعد المعطيات Access.
- برمجيات عرض الشرائح Power Point .

- برمجيات معالجة الصور Photo Shop .
- برمجيات التراسل الإلكتروني Outlook Express .

ب- برمجيات النظم التطبيقية Application Systems

وهذه البرمجيات يتم تصميمها وتطويرها لإنجاز عمل محدد ، مثل البرنامج المستخدم لإدخال درجات الطلاب وبرنامج نظام الرواتب والمحاسبة أو برامج مخصصة لتنفيذ أعمال أخرى .

٣- لغات البرمجة Programming Languages

وهي لغات عالية أو متدنية المستوى يتم بوساطتها كتابة البرامج اللازمة للمستثمر لحل المسائل المطلوبة مثل لغات :

BASIC , Pascal , C++ , C# , ...

ومن خلالها يستطيع المستخدم كتابة جميع أنواع البرمجيات ، سواءً أكانت برمجيات تطبيقية أو برمجيات أخرى .

٤- البرمجيات الخدمية Utilities Programs

وهي برمجيات تقوم بإجراء الكثير من الوظائف والعمليات لمستخدم الحاسوب ، مثل التحكم في الحاسوب وصيانته بشكل سهل ، وتوفر عليه المعرفة الكاملة لأوامر نظام التشغيل اللازمة لأداء وتنفيذ هذه العمليات ، وكذلك تمكن المستخدم من أداء الكثير من الوظائف التي لا يقدمها نظام التشغيل مباشرة ، وتقوم بعض البرمجيات الخدمية بما يلي :

- تشخيص المشاكل المتعلقة بالأقراص وحلها .
- خدمة إصلاح القرص .
- إعادة بناء القرص واستعادة معطياته بعد مسحه عن طريق الخطأ .
- ضغط الملفات لتقليل حجم تخزينها في الأقراص .
- تسريع أداء الجهاز .

- تكوين النسخ الاحتياطية للأقراص بسرعة وسهولة .
- حفظ وتأمين الملفات بضغطها أو إخفائها أو حمايتها بكلمة مرور .

٥- **مترجمات لغات البرمجة**

يوجد عدد من مترجمات لغات البرمجة ، ويكون لكل لغة برمجة ، مترجمها الخاص بها مثل: المترجم Compiler و المفسر Interpreter والمجمع Assembler .

١-٧-١- لغات البرمجة

استخدمت البرمجة كعلم مع ظهور أول حاسوب في الأربعينات حيث تم استخدام البرمجة اليدوية عن طريق اختيار المفاتيح المخصصة ، وبعد ذلك استخدمت اللوحات الإلكترونية بدل المفاتيح ، وتطورت البرمجة مع تطور الحاسوب حيث استخدمت الأرقام الثنائية لبرمجة الحاسوب ، واستخدمت الذاكرة لتخزين هذا البرنامج وقد سمي هذا البرنامج بلغة الآلة Machine Language نظراً لاستخدامها الأرقام الثنائية ، ومع اتساع استعمال الحاسوب أصبح من الصعب استخدام لغة الآلة نظراً لصعوبة تذكر الأرقام ولكبر حجم البرنامج وظهر ما يسمى بلغة الاختصارات ، وفي الفترة التي ظهرت فيها لغة الاختصارات (1952) تم استخدام المترجمات (Compilers) لتقوم بتحويل لغة الاختصارات إلى لغة الآلة ، ثم طرأ بعد ذلك تحسين على هذه اللغة وظهر ما يسمى بلغة التجميع Assembler ، واستمر الإنسان في تطوير لغات البرمجة وتقريبها من لغة الإنسان وظهرت مجموعة كبيرة من لغات البرمجة ، سميت بلغات البرمجة عالية المستوى . وتنقسم مستويات لغات البرمجة إلى مستويين :

١- **لغات البرمجة منخفضة المستوى LLL** (Low Level Languages)

صنفت منخفضة المستوى نظراً لصعوبتها وابتعادها عن لغة الإنسان ، إذ

يحتاج المبرمج إلى تحديد السجلات الداخلية للمعالج بشكل مفصل ونذكر منها :

- لغة الآلة Machine Language
- لغة الاختصارات Mnemonic
- لغة التجميع Assembly Language

و للغات البرمجة منخفضة المستوى الخصائص التالية :

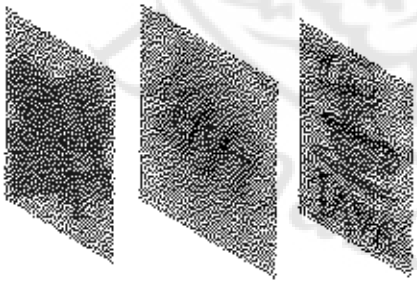
١. لغة صعبة التعلم والفهم .
٢. إن عملية البرنامج بهذه اللغة بطيئة ومتعبة للمبرمج .
٣. صعوبة تعديل البرنامج واكتشاف الأخطاء .
٤. يوجد لكل حاسوب لغة تجميع خاصة به ولهذا لا يمكن تنفيذ نفس البرنامج على أجهزة مختلفة .
٥. تتطلب من المبرمج معرفة كبيرة بالبنية الداخلية للحاسوب

٢ - لغات البرمجة عالية المستوى HLL (High Level Languages)

سميت بلغات عالية المستوى نظراً لقربها من لغة الإنسان وهي مكتوبة بلغة مفهومة وتستخدم أوامر سهلة مثل : Read , write , print , go to , ...

ولكي يتم تنفيذ اللغة عالية المستوى في الحاسوب تحتاج إلى مترجم خاص يقوم بتحويل اللغة عالية المستوى إلى لغة الآلة ، نذكر منها :

- لغة بيزك BASIC
- لغة فورتران FORTRAN
- لغة Pascal
- لغة C++ و Java و C#



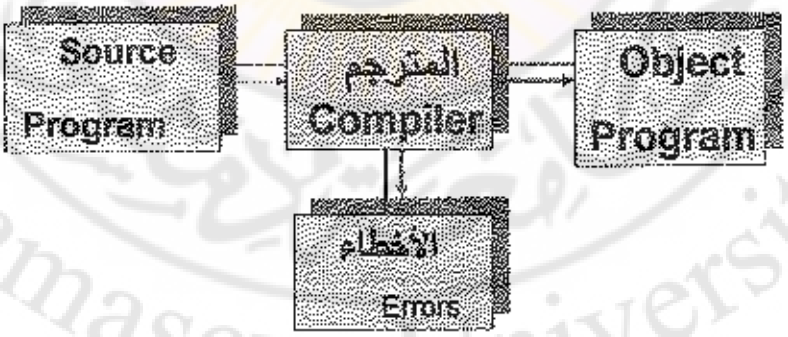
أما خصائص لغات البرمجة عالية المستوى هي :

١. سهولة تعلمها واستعمالها لأنها قريبة من لغة الإنسان .
٢. لا تتطلب من المبرمج معرفة كبيرة بالبنية الداخلية للحاسوب .
٣. سهولة كشف الأخطاء وتصحيحها .
٤. سهولة تعديل البرنامج .
٥. الزمن اللازم لكتابة البرنامج أصغر بكثير من اللغات منخفضة المستوى .
٦. معظم لغات البرمجة عالية المستوى لها خاصية التوثيق الداخلي .
٧. معظم لغات البرمجة عالية المستوى متخصصة بعضها يستعمل للأغراض التجارية وبعضها للأغراض الهندسية وبعضها للاثنين معاً .

١-٧-٢- مترجمات لغات البرمجة عالية المستوى

a) المترجم Compiler

هو عبارة عن برنامج يقوم بتحويل (ترجمة) البرنامج الأولي المكتوب من قبل المبرمج بإحدى لغات البرمجة عالية المستوى إلى برنامج مكتوب بلغة الآلة ، ويبين الشكل(١-١١) المخطط الصندوقي للمترجم .



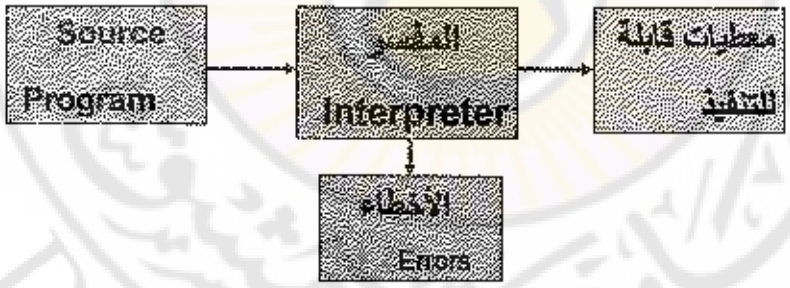
الشكل(١-١١) - يبين المخطط الصندوقي للمترجم

للمترجم عدد من الوظائف منها :

١. ترجمة البرنامج الأولي المكتوب بلغة عالية المستوى دفعة واحدة أي بشكل كامل (البرنامج كاملاً) .
٢. بيان مواقع الأخطاء الإملائية والقواعدية الموجودة في البرنامج وذلك من خلال إصدار قائمة بالأخطاء على الشاشة أو في ملف خاص .
٣. تزويد المستخدم ببعض المعلومات عن البرنامج المترجم كحجم البرنامج وزمن الترجمة .

٢- المفسر Interpreter

وهو عبارة عن برنامج يقوم بترجمة وتنفيذ تعليمات البرنامج الأولي (المصدر) الواحدة تلو الأخرى مع إمكانية اكتشاف الأخطاء الإملائية والقواعدية إن وجدت ، ويبين الشكل(١-١٢) المخطط الصندوقي للمفسر .



الشكل(١-١٢) - يبين المخطط الصندوقي للمفسر

أما الاختلاف بين المفسر والمترجم هي :

١. لا يقوم المفسر بتوليد برنامج الهدف ، لذا نحتاج إلى ترجمة جديدة لبرنامج المصدر عند تنفيذه مرة ثانية .
٢. زمن تنفيذ البرنامج باستخدام المفسر عادة أطول منه في حالة استخدام المترجم .
٣. في حالة استخدام المترجم نحتاج إلى ذاكرة لتخزين برنامج الهدف .
٤. نستخدم المفسر عادة في الحالات التي لا نحتاج فيها إلى تكرار تنفيذ البرنامج لفترات طويلة .
٥. يخزن المفسر عادة في ذاكرة الـ ROM بينما يخزن المترجم في وحدات التخزين المساعدة .
٦. يُعد المترجم أكثر تعقيداً وكلفة من المفسر .

١-٨- نظم التشغيل Operating System

تُعد برامج نظام التشغيل من أهم البرامج وبدونها لا يعمل جهاز الحاسوب ، وهو أول برنامج يتم تحميله إلى ذاكرة الحاسوب RAM ، وبعدها يصبح الحاسوب جاهزاً لاستقبال وتنفيذ أوامر المستخدم وتحميل وتشغيل البرامج التطبيقية ، وهو الوسيط بين المكونات العتادية لجهاز الحاسوب Hardware وبين البرامج التطبيقية Software ، وبالتالي فهو يوفر بيئة أو واجهة عمل User Interface من خلال واجهة التطبيق السهلة الاستخدام التي تحتوي على النوافذ والقوائم والرموز وغيرها، والتي تمكن المستخدم من تشغيل البرامج التطبيقية دون الدخول في تفاصيل الكيفية التي تعمل بها هذه المكونات. وتقوم بالإشراف على عمليات وحدة المعالجة المركزية ووحدات الإدخال والإخراج ، ويبين الشكل(١- ١٣) طبقات البرمجيات .

الشكل(١-١٣) – يبين طبقات البرمجيات

١-٨-١- تعريف نظام التشغيل

The Definition of The Operating System

يمكن القول بأن نظام التشغيل (Operating System) هو مجموعة من البرامج المتكاملة التي توفر الاتصال بين المستخدم والحاسوب ، وتستطيع التحكم بالحاسوب وملحقاته ، وتستخدم جميع الموارد الحاسوبية بالطريقة المثلى ، وتستطيع أن توفر واجهة تخاطب (Interface) سهلة وسلسة بين المستخدم والحاسوب بحيث يتمكن المستخدم من الوصول إلى الموارد كافة.

وبكلام آخر يمكن القول بأن نظام التشغيل (Operating System) هو منصة التحكم التي يستخدمها الإنسان (المستخدم العادي أو المبرمج) للتحكم بالحاسوب وملحقاته ، ومن ثم استثماره وذلك باستخدام مجموعة منتهية من الأوامر (Commands) .

وتختلف مهام نظام التشغيل باختلاف أحجام الحاسوب ، فالحواسيب الكبيرة تحتاج إلى نظم تشغيل عالية الكفاءة تمكنها من التعامل مع العديد من وحدات الحاسوب والوحدات المحيطية به (مثل الطابعات ، ووسائط التخزين وغيرها) ، وكذلك تمكنها من توفير إمكانية تشغيل متعدد لمستخدمي الحاسوب في الوقت نفسه ، أما في الحواسيب الشخصية فان نظم التشغيل تكون أقل تعقيداً ، حيث إنها تتعامل عادة مع حاسوب واحد لمستخدم واحد .

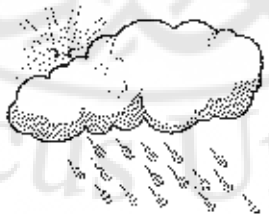
ويمكن تعريف نظام التشغيل على أنه مجموعة البرمجيات الجاهزة المسؤولة عن ضبط وإدارة التحكم بكافة الوحدات الأساسية المكونة للحاسوب وما تحتويه هذه الوحدات من معلومات ومعطيات ، مثل :

- إدارة الذاكرة الرئيسية
- إدارة المعالج
- إدارة وحدات الإدخال والإخراج
- إدارة المعلومات والمعطيات ووسائط التخزين

إذاً هو عبارة عن حزمة من البرامج تقوم بدورين أساسيين:

١. إدارة مكونات الحاسوب العتادية (المعالجات ، الذاكرة ، الأقراص بأنواعها المختلفة) و البرمجية (البرامج و الملفات و التطبيقات المختلفة).

٢. ربط التطبيقات(البرامج) بالمكونات العتادية .



١-٨-٢-عمل نظام التشغيل في الحاسوب

يتم تخزين نظام التشغيل عادة على قرص مرن أو على قرص ضوئي أو على القرص الصلب ، وعند بدء تشغيل الحاسوب يقوم المعالج تلقائياً بالبداية بقراءة وتنفيذ التعليمات المخزنة على ذاكرة القراءة فقط (ROM) ، والتي تقوم بفحص المكونات المادية للجهاز والتأكد من سلامتها ، فإذا ما وجد عطل معين ، يتوقف الحاسوب عن العمل ويعرض رسالة على الشاشة عن العطل الحاصل ، وأما إذا ما وجد المعالج جميع الوحدات سليمة فيقوم بالبحث عن نظام التشغيل ويبدأ أولا بالبحث في الأقراص المرنة أو الضوئية على الجهاز ، فان لم يجد نظام التشغيل هناك ، يقوم المعالج بالبحث في القرص الصلب وبعد التعرف على الوحدة التي تخزن نظام التشغيل يقوم المعالج بتحميله إلى الذاكرة الرئيسية (RAM) ، ثم يبدأ في تنفيذ تعليمات النظام واحدة تلوى الأخرى وعادة يقوم النظام بعد تحميله بإعطاء الفرصة لمستخدم الجهاز لإدخال أوامره فمثلاً إذا ما طلب المستخدم تنفيذ برنامج معالج النصوص ، يقوم نظام التشغيل بالبحث عن هذا البرنامج في وحدات التخزين وإذا ما وجده يقوم بتحميله إلى مواقع مخصصة بالذاكرة الرئيسية (RAM) وينقل التحكم بالمعالج بعد ذلك إلى برنامج معالج النصوص حيث يتم تنفيذ تعليماته من قبل المعالج وعند الانتهاء من تنفيذ البرنامج التطبيقي يعود التحكم تلقائياً إلى نظام التشغيل والذي يسمح لمستخدم الجهاز بإعطاء أوامر جديدة وهكذا ... ، ويمكن تلخيص مهام نظام تشغيل للحاسوب بالخطوات التالية :

١. قراءة وتنفيذ التعليمات والأوامر المخزنة في ذاكرة القراءة فقط (ROM) .
٢. فحص وحدات الحاسوب للتأكد من سلامتها .
٣. تحميل نظام التشغيل من الأقراص المرنة أو الضوئية أو من القرص الصلب .
٤. استلام أوامر مستخدم الجهاز .

٥. تحميل البرامج التطبيقية وتنفيذ تعليماتها .

٦. العودة إلى نظام التشغيل وانتظار أوامر المستخدم وتكرار الخطوات السابقة .

وتصنف برمجيات نظم التشغيل إلى أربع مجموعات برمجية هي :

١. برمجيات إدارة الذاكرة الرئيسية (Memory Management Programs) .

٢. برمجيات إدارة المعالجات (Processes Management Programs).

٣. برمجيات وحدات الإدخال والإخراج (Input/Output Management Programs) .

٤. برمجيات إدارة المعلومات(Information Management Programs)

وتتم هذه البرمجيات بلغة الآلة (Machine Language) .

١-٨-٣- أنواع نظم التشغيل

Operating System Types

تنقسم نظم التشغيل من حيث قدرتها على تشغيل أكثر من برنامج للمستخدم في الوقت نفسه إلى قسمين:

١. أنظمة متعددة المهام Multi-Tasking .

٢. أنظمة وحيدة المهام Single-Tasking .

كما تنقسم نظم التشغيل من حيث قدرتها على السماح لأكثر من مستخدم بتشغيل برامجها في الوقت نفسه إلى قسمين:

١. أنظمة متعددة المستخدمين Multi-user .

٢. أنظمة وحيدة المستخدم Single-user .

- أنظمة وحيدة المستخدم (Single _ User O.S)

وهو النظام الذي يسمح لمستخدم واحد فقط بالوصول إلى موارد الجهاز وبرامجه ، ومن الأمثلة على هذه الأنظمة (Windows 95 – DOS) .

- أنظمة متعدد المستخدمين (Multi _ User O.S)

وهو النظام الذي يسمح لأكثر من مستخدم بالوصول إلى موارد الجهاز وبرامجه ، ومن الأمثلة على هذه الأنظمة (Windows XP , Windows NT , Novell , Unix , Windows10 ,Linux) .

١ – ٨ – ٤ – بنية نظم التشغيل العامة

يقوم نظام الـ BIOS بتحميل ملفات نظام التشغيل إلى ذاكرة الحاسوب الرئيسية (RAM) عند بدء تشغيل الحاسوب ، ووظيفة هذه الملفات القيام بتحميل وتشغيل نظام الحاسوب أو إعادته لحالته الأولى، وبعضها يبقى في الذاكرة لإبقاء النظام عاملا حتى يتم وقف التشغيل ، وباقي الملفات يتم تحميلها عند الحاجة إليها. وعلى العموم فإن أكثر نظم التشغيل لها بنية عامة مؤلفة من مجموعة من الطبقات على شكل دوائر متحدة المركز ، الدائرة الداخلية هي المكونات العتادية وهي كما يلي :

١ – النواة (The Kernel)

هو البرنامج الرئيسي القائد ، وهو المسؤول عن عمليات الدخل والخرج الأساسية ويشرف على التعامل مع وسائط التخزين ، وهي عبارة عن مجموعة الملفات التي تقوم بإدارة كل من أسس الأعمال التي لا نراها، مع أنها في الواقع هي الأساس في العمل.

٢- برنامج الغلاف (The Shell)

هو البرنامج الذي يفسر أوامر المستخدمين ويحولها إلى تعليمات يفهمها المعالج وينفذها ، وتظهر نتائجها للمستثمر ، وهو الذي يعطي رسائل التحذير والخطأ .

٣ - البرامج التطبيقية المساعدة Tools and Application

وهي البرامج المتممة لعمل نظام التشغيل ، مثل برامج تنظيم القرص والاختبار وإلغاء تجزئة القرص وغيرها .

١-٨-٥- مهام نظام التشغيل الرئيسية

The Basic Missions of O.S

تختلف مهام نظام التشغيل من حاسوب إلى آخر ، وهي مهام ووظائف متعددة تتعلق بعمليات تشغيل الحاسوب في مكوناته وأجهزته المادية وملحقاته . ومن أهم وظائف نظام التشغيل ما يلي :

١. **التحكم في مسار المعطيات** : حيث يقوم نظام التشغيل بنقل المعطيات داخل الحاسوب من وحدة إلى أخرى ، كما يقوم بتنظيم تبادلها بين الوحدات المختلفة بجهاز الحاسوب ، وكذلك ينظم عمليات حفظ المعطيات والبرمجيات ،ويحتفظ بمعلومات مفصلة عن حجمها وأماكن حفظها .

٢. **تحميل البرامج التطبيقية إلى الذاكرة الرئيسية** : لأن من الوظائف المهمة التي يقوم بها نظام التشغيل تحميل البرامج التطبيقية إلى الذاكرة الرئيسية من الوحدات المحيطية أو من وسائط التخزين المرتبطة بالحاسوب . ويقصد بعملية تحميل البرامج هو نقلها من وسائط التخزين إلى الذاكرة الرئيسية ثم إلى وحدة المعالجة المركزية لتنفيذها ، وبعد تنفيذ البرنامج يقوم نظام التشغيل

بإزالة البرنامج من الذاكرة الرئيسية ، وذلك إذا لم يكن البرنامج من البرامج الدائمة في الحاسوب ، وذلك لإفساح المجال لتحميل وتنفيذ برامج أخرى .

٣. **تنظيم التسلسل الزمني لتنفيذ العمليات في الحاسوب** : فينظم ترتيبا (دوراً) للمسائل الواردة للمعالجة ويعيد نتائجها إلي أماكن الطلب .

٤. **تنفيذ الأوامر الصحيحة الواردة من المستخدم** : وذلك لأن هذه الأوامر هي كلمات تكتب من لوحة المفاتيح أو تحكم بأيقونة أو رموز من واجهة تخاطب بيانية رسومية ، ودور نظام التشغيل هنا هو ترجمة هذه الأوامر إلى تعليمات يفهمها المعالج ويقوم بتنفيذها وإعطاء النتيجة (القيام بالعملية المطلوبة) للمستخدم .

٥. إدارة الذاكرات بمختلف أنواعها واستخدامها على الوجه الأمثل .

٦. إدارة عمليات التخزين على وسائط التخزين وفق بروتوكولات معينة والقراءة منها (إدارة التعامل مع الملفات) .

٧. الإشراف على جميع الموارد وتقاسمها وتنظيمها في حال تعدد المستخدمين أو تعدد الطلب عليها .

٨. التحكم في كامل عمليات الدخل والخرج المختلفة ومراقبتها ، وأيضاً التحكم في وحدات الإدخال والإخراج ، ويشمل ذلك عمليات التحكم في إدخال المعطيات عن طريق لوحة المفاتيح أو الفأرة أو غيرهما ، وعمليات عرض المعلومات على الشاشة أو إرسالها إلى الطابعة أو أي وحدات أخرى .

٩. التعامل مع البرامج والتطبيقات المختلفة المنفذة على الحاسوب .

١٠. القدرة على التعامل مع شبكات الحواسيب وإدارتها .

١١. حماية التطبيقات والنظام نفسه في حالات التوقف المفاجئ (Abort) والعطب (Failure) ، وذلك عن طريق إضافة برنامج لإدارة المهام (Task Manager) لإغلاق البرمجيات التي لا تستجيب بدلاً من إعادة

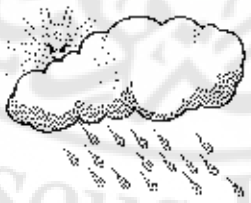
التشغيل كما في الإصدارات السابقة ، وذلك عن طريق الضغط بالوقت نفسه على الأزرار الثلاثة Alt + Ctrl + Del.

١٢. القدرة على إضافة إمكانيات جديدة إلى نظام التشغيل أو إلغاء إمكانيات أصبحت غير مطلوبة (تجزئة نظام التشغيل) ، وذلك دون الحاجة إلى كتابة برنامج جديد لنظام التشغيل ، وتتعلق هذه الخاصية بتصميم برنامج نظام التشغيل حيث يتم تصميمه على شكل بنيوي (Structured) ، أي أن لكل وظيفة يؤديها نظام التشغيل برنامجاً خاصاً بها (Module) فعندما يراد إلغاء إحدى الوظائف يتم فصل البرنامج الخاص بها ، كما يمكن ببساطة إضافة برنامج (Module) آخر يحقق وظيفة جديدة .

١٣. القدرة على أداء عدة وظائف في الوقت نفسه (تعدد المهام) Multitasking . ويقصد بتعدد المهام هو قدرة نظام التشغيل على تنفيذ أكثر من عملية على الحاسوب في الوقت نفسه ، فمثلاً عند تشغيل معالج النصوص ، يمكن في الوقت نفسه طباعة ملف على الحاسوب ، وسماع أغاني على السواقة الليزرية .

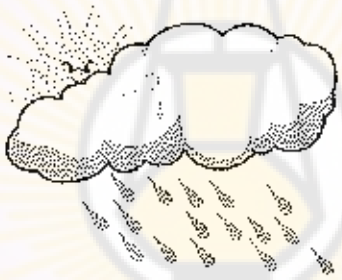
١٤. التحكم بالوصول إلى الأقراص ووسائط التخزين .

إن المهام السابقة مهام عامة ، وقد لا توجد جميعها في نظام تشغيل واحد ولكن معظمها يمكن أن يكون في نظام تشغيل محدد .



الفصل الثاني

مبادئ الخوارزميات



مبادئ الخوارزميات

٢-١- مقدمة

الخوارزمية Algorithm مفهوم قديم يعود إلى مطلع القرن التاسع الميلادي(568- 668 هـ)، في أوج الدولة العباسية زمن الخليفة المأمون، ويعود الفضل في إيجاد هذا المفهوم الرياضي إلى العالم الجليل محمد بن موسى الخوارزمي الذي عاش في بغداد وهو من مدينة خوارزم ، وكلمة خوارزم (algorism) في الأصل كانت مقتصرة على القوانين الرياضية التي تستخدم الأرقام العربية و طُورت في اللاتينية من الخوارزمي (al-Khwarizmi) لتصبح (algorithm) في القرن الثامن عشر الميلادي لتشمل جميع إجراءات حل المشكلات و تنفيذ المهمات.

٢-٢- تعريف الخوارزمية

يمكن تعريف الخوارزمية بعدة أشكال نذكر منها:

١. الخوارزمية مجموعة من الخطوات المتسلسلة والمحددة والمنتهية التي تؤدي في حال تنفيذها إلى حل قضية معينة والوصول إلى نتائجها.

٢. الخوارزمية طريقة الوصول إلى هدف معين.

٣. الخوارزمية مجموعة متتالية من العمليات المعرفة والمنتهية اللازمة لإنجاز عمل ما، أو لحل مسألة ما والحصول على نتيجة صحيحة.

تعالج الخوارزمية معطيات مُدخلة في معظم الحالات، والمعطيات المعالجة لا تقتصر على الأعداد والأرقام، بل تشمل الرموز والنصوص والرسوم والصور والأصوات كُمدخلات ومُخرجات، ويتزايد الاهتمام بالخوارزميات بشدة مع ظهور

الحواسيب لضرورة استخدامها في حل المسائل في جميع المجالات العلمية والتقنية والاجتماعية والاقتصادية والصناعية والتجارية بواسطة الحاسوب.

٢-٣- خطوات حل مسألة باستخدام الحاسوب

عند حل أي مسألة باستخدام الحاسوب لا بد من اتباع مجموعة من الخطوات سنقوم بإيجازها فيما يلي:

١. **تعريف وتحليل المسألة:** إن تعريف المسألة هو عبارة عن دقة التعبير في تطبيق المسألة بحيث يجعلها معروفة ومفهومة بصورة واضحة وبدون أي غموض لجميع الأشخاص العاملين ضمن مجال الاختصاص الذي تخضع له المسألة، أما تحليل المسألة ووضع طريقة الحل فهو أصعب وأشق الخطوات، و من أجل الوصول إلى حل صحيح فإن كثيراً من القوانين والطرق الرياضية المناسبة لحل المسألة يجب أن تستعمل وربما تحتاج أيضاً إلى تطوير هذه القوانين والطرق لنجعلها تناسب الحل في كثير من الأحيان ففي هذه الخطوة يجب تحديد:

- طبيعة المخرجات(النتائج) و تنظيم كتابتها.
- المدخلات (البيانات أو المعلومات) و تحديد نوعها و تنظيم إدخالها إلى الحاسوب.
- طرق الحل المناسبة و تقييمها بما يتلاءم مع كيفية تنفيذها بالحاسوب و في ضوء ذلك يتم اختيار الحل الأفضل.

٢. **برمجة الحل خطياً:** بعد اختيار طريقة الحل المثالية و تحديد كل ما تشمله من علاقات رياضية، يتم التعبير عنها على شكل خطوات متسلسلة ومترابطة منطقياً، ودقيقة الوصف تؤدي إلى الوصول إلى حل المسألة ، و هذه الخطوات المتسلسلة تعرف بخوارزمية المسألة Algorithm of the

Problem ، و يمكن تمثيل هذه الخوارزمية بعد إيضاح جميع التعليمات والأوامر المتسلسلة التي يراد تنفيذها في كل خطوة بمخطط وصفي تسلسلي يدعى بالمخطط التدفقي Flowchart وذلك باستخدام مجموعة من الأشكال الاصطلاحية الرمزية.

٣. **برمجة الحل باستخدام إحدى لغات البرمجة:** إن المخطط التدفقي هو عبارة عن تخطيط تصوري(بياني) مساعد وسهل الملاحظة بالنسبة للمبرمج ولكنه غير مفهوم عند الحاسوب، لذلك فإن طريقة الحل الممثلة بالمخطط التدفقي يجب أن تكتب بإحدى لغات البرمجة التي يفهمها الحاسوب والتي تتلاءم مع حل المسألة ، ويلي ذلك كتابة البرنامج على الوسط الخارجي المناسب وإدخال البرنامج إلى الحاسوب ، و البرنامج الناتج من هذه الخطوة والمكتوب بإحدى اللغات الرمزية أو اللغات عالية المستوى يسمى بالبرنامج المصدري source program.

٤. **ترجمة البرنامج المصدري:** بعد الانتهاء من كتابة البرنامج المصدري يتعين إدخاله إلى الحاسوب للتأكد من صحة كتابته من جهة، ثم لترجمته إلى لغة الآلة بوساطة برنامج الترجمة الخاص في حالة عدم وجود أخطاء في البرنامج المصدري الذي يسمى بالمترجم compiler ، وتمر عملية الترجمة في المراحل الآتية:

- **مرحلة التحليل اللفظي Lexical analysis:** في هذه المرحلة يتم مطابقة مفردات برنامج المصدر والعلاقات والأسماء مع تلك المسموح بها في اللغة واكتشاف أي أخطاء فيها.
- **مرحلة التحليل القواعدي Syntax analysis:** في هذه المرحلة تجري عملية مطابقة تعليمات البرنامج المصدري مع القواعد اللغوية المستخدمة،

و اكتشاف أي أخطاء فيها، بالإضافة إلى عملية تحويل البرنامج المصدري إلى تعليمات وأوامر رمزية بلغة التجميع.

- **مرحلة ترجمة البرنامج إلى لغة الآلة:** في هذه المرحلة نحصل على برنامج الهدف object program والذي بموجبه يمكن البدء في عملية التنفيذ.

٥. **اختبار البرنامج و تنفيذه:** بعد الحصول على البرنامج الهدف، يتم اختباره للتأكد من صحته منطقياً وذلك باستخدام عينة من المعطيات الاختبارية Test Data، فإذا ثبت صحة طريقة الحل بمطابقة النتائج الخارجة من الحاسوب مع النتائج التي تم الحصول عليها يدوياً على سبيل المثال، يمكن تنفيذ البرنامج على المعطيات الحقيقية.

٢ – ٤ – أنواع الخوارزميات

لا بد من الإشارة إلى أننا نستخدم مفهوم الخوارزمية في حياتنا اليومية من حيث لا ندري، حيث إن أي عمل نقوم به وننفذه بالشكل الصحيح لا بد أن يخضع لخطة عمل محددة ، وبشكل عام يمكن تقسيم الخوارزميات إلى خوارزميات حسابية وخوارزميات غير حسابية.

٥-٤-١ – الخوارزميات غير الحسابية

من خلال التسمية نلاحظ أن هذا النوع من الخوارزميات يكون بعيداً كل البعد عن العمليات الحسابية وربما كانت هذه الخوارزميات أكثر الخوارزميات استخداماً ونذكر منها:

- ✓ خوارزميات معالجة النصوص.
- ✓ خوارزميات تخزين المعلومات واستعادتها.

- ✓ خوارزميات إدارة قواعد البيانات والمساعدة في اتخاذ القرار في جميع نواحي الحياة.
- ✓ خوارزمية التدقيق الإملائي.

وفيما يلي سنطرح بعض الخوارزميات غير الحسابية ومن حياتنا اليومية:

مثال (٢-١): خوارزمية طرح سؤال ضمن المحاضرة.

خوارزمية طرح السؤال هي مجموعة الخطوات الواجب إتباعها للاستفسار عن قضية معينة ضمن المحاضرة، وهذه الخطوات يمكن سردها كما يلي:

١. الانتظار حتى يصل المحاضر إلى نهاية المقطع الكلامي.
٢. رفع اليد حتى يسمح للطالب بالكلام.
٣. خفض اليد والمباشرة في طرح السؤال.
٤. الاستماع للإجابة حتى النهاية.

تؤدي هذه الخطوات مجتمعة ومرتبة إلى طريقة سليمة لطرح السؤال.

مثال (٢-٢): خوارزمية عبور شارع ذي اتجاهين من قبل إنسان عادي.

المسألة هي عبور الشارع بسلامة ولتحقيق هذا الهدف لا بد من إتباع مجموعة من الخطوات التالية:

١. إذا كانت الشارة خضراء، عندئذ لا تقطع الشارع.
٢. إذا كانت الشارة حمراء اقطع الشارع إلى منتصفه، ناظراً إلى السيارات من اليسار.

٣. إذا كانت الشارة خضراء للنصف الثاني من الشارع، عندئذ لا تقطع الشارع.

٤. إذا كانت الشارة حمراء للنصف الثاني من الشارع ، عندئذ اقطع الشارع ناظراً إلى السيارات من اليمين.

من الواضح أن هذه الخوارزمية صالحة لكي يقطع الشارع أي إنسان عادي، وإن تطبيقها على هذا النحو، يضمن حياة العابر من خطر السيارات.

٢-٤-١- الخوارزميات الحسابية

أطلق اسم الخوارزميات الحسابية على تلك الخوارزميات التي تتعامل مع المقادير الرياضية ، وفيما يلي سنطرح بعض الخوارزميات الحسابية:

مثال (٢-٣): خوارزمية حساب وطباعة قيمة الكسر المعرف بالشكل التالي:

$$Y = \frac{x+3}{x-8}$$

من الواضح أن الحل بسيط جداً، ويستطيع انجازه أي شخص لديه إلمام بسيط بالعمليات الحسابية ولا يحتاج إلا إلى معرفة قيمة المتغير x ، ولكن الغاية من هذا المثال هو كيف نصل إلى المطلوب من خلال إتباع مجموعة من الخطوات المتسلسلة والمحددة والتي يمكن صياغتها كما يلي:

١. إدخال قيمة للمتغير x .

٢. حساب قيمة البسط $x + 3$.

٣. حساب قيمة المقام $x - 8$.

٤. قسمة البسط $x+3$ على المقام $x-8$ وإسناد الناتج إلى Y .

٥. طباعة قيمة Y .

لكي يصبح أي تعبير رياضي خوارزمية لا بد أن يقترن بتسلسل تنفيذ عملياته، أي لا بد من إضافة بعض الشروط والقواعد مثل: البدء دوماً وفق أفضليات تحدد بحسب نوع المسألة المطروحة، وجعل هذه الأفضلية قواعد للتنفيذ تمكننا من الوصول إلى خوارزمية صالحة للتنفيذ.

٢-٥- أولوية تنفيذ العمليات الحسابية والمنطقية

إن أولوية تنفيذ العمليات الحسابية والمنطقية كما يجريها الحاسوب موضحة في الجدول (٢-١) .

العملية	رمز العملية	رقم الأولوية	جهة التنفيذ عند التساوي	ملاحظات
فك الأقواس	( )	1	من اليسار	
الزيادة أو النقصان	++ أو --	2	من اليمين	رمز برمجي C++
العمليات الحسابية	% / *	3	من اليسار	
العمليات الحسابية	- +	4	من اليسار	
عمليات المقارنة	< <= > >=	5	من اليسار	
عمليات المقارنة	!= ==	6	من اليسار	رمز برمجي C++

العمليات المنطقية	&& ||	7	من اليسار	رمز برمجي C++
العمليات الشرطية	? :	8	من اليمين	رمز برمجي C++
عمليات الإسناد	= += -= *= /= %=	9	من اليمين	

الجدول (٢-١) – يبين جدول أولوية تنفيذ العمليات الحسابية والمنطقية

٢-٦- طرائق صياغة الخوارزميات

الخوارزمية هي مجموعة قواعد وقوانين مكتوبة ومحددة مسبقاً ، تستعمل لوصف الطرائق الرياضية والخطوات المنطقية اللازمة لتنفيذ حل المسائل والمشكلات بوساطة الحاسوب ، ومن أهم خواص الخوارزميات ما يلي :

١. يجب أن تكون الخوارزمية واضحة ومحددة ويمكن تقسيمها إلى خطوات معينة ومتتالية تؤدي إلى النتيجة نفسها .
٢. أن تكون الخوارزمية واحدة مهما اختلفت أساليب المعالجة.
٣. أن تكون الخوارزمية صالحة لحل جميع المسائل من النوع أو الطراز نفسه .
٤. إن تصميم الخوارزمية يتطلب مهارة علمية وعملية عالية .

ويمكن صياغة الخوارزميات بطرق عديدة تتفاوت فيما بينها من حيث دقة التعبير وسهولة الفهم ومن أهمها :

٢-٦-١- الطريقة اللغوية Language Method

تستخدم الطريقة اللغوية لصياغة الخوارزميات ، وفيها يتم استخدام اللغة المحكية(العربية، الإنكليزية،....) من أجل التعبير عن خطوات الحل ويجري تنفيذ تعليمات الخوارزمية بالتسلسل وفق ورودها في نص الخوارزمية في خطوات متسلسلة محددة ومعرفة تحدد سياق هذا التنفيذ ، وهذه الطريقة تستخدم على نطاق واسع في صياغة بعض أنواع الخوارزميات الموجهة للجمهور كتعليمات تشغيل الأجهزة واستخدامها ، وفي وصفات تحضير أطباق الطعام وغيرها.

مثال (٢-٤): خوارزمية حساب وطباعة مساحة المستطيل.

١. البداية(بداية الخوارزمية).
٢. إدخال الطول والعرض.
٣. حساب المساحة وفق الدستور: المساحة = الطول × العرض
٤. طباعة مساحة المستطيل.
٥. النهاية (نهاية الخوارزمية) .

٢-٦-٢- الطريقة الترميزية Pseudo code Method

تعتمد الطريقة الترميزية على قواعد محددة يمكن أن تكون مستنتجة من المفاهيم الرياضية، وفي هذه الطريقة يتم استخدام الرموز الرياضية بدلاً من استخدام الكلام المحكي للتعبير عن خطوات الحل ، وتُعد هذه الطريقة الأقرب إلى لغات البرمجة وتسمى عند البعض " طريقة شبه البرنامج ". ويوضح المثال(2- 4) هذه الطريقة .

مثال (٢-٥): خوارزمية حساب وطباعة مساحة المستطيل.

1.Start

2. Input x ,y

3. Let s = x*y

4.Print s ,x,y

5.Stop

٢-٦-٣- طريقة المخططات التدفقية

Flowchart Method

تُستخدم طريقة المخططات التدفقية في صياغة الخوارزميات وفيها يتم استخدام أشكال هندسية خاصة وأسهم تصل بينها، للتعبير عن خطوات تنفيذ الخوارزمية ، إضافة إلى عبارات باللغة الطبيعية أو تعابير رياضية ومنطقية ، وتُعد هذه الطريقة أكثر الطرق انتشاراً واستخداماً في توصيف الخوارزميات.

من المعلوم أن الخوارزمية هي عبارة عن مجموعة من الخطوات المتسلسلة التي تصف بصورة مضبوطة وبدون أي غموض جميع الخطوات الرياضية والمنطقية اللازمة لحل مسألة ما، ولكن هذا الوصف في كثير من الأحيان يكون معقداً وصعب الملاحظة والتتبع لذلك فإن المخطط التدفقي الذي يمثل وصفاً تصويرياً لخطوات الخوارزمية يكون أكثر وضوحاً ، ويقوم المخطط التدفقي مقام الخوارزمية ويمكن بوساطته ملاحظة تتبع التسلسل المنطقي لحل المسألة بكل سهولة، وغالباً ما يكون استخراج الخوارزمية من المخطط التدفقي أسهل بكثير من كتابة الخوارزمية مباشرة ، ويبين الشكل(٢-١٤) أهم الرموز الاصطلاحية التي يمكن اعتمادها في توصيف العمليات عند استخدام المخططات التدفقية .

مثال	العملية التي يمثلها	الرمز
Stop Start	البداية والنهاية Start and End	
C=A+B	عملية الإسناد (العمليات الحسابية) Process	
Input A,B Print C	إدخال البيانات وإخراج المعلومات Input/Output Data	
Yes A=B No	اتخاذ القرار Decision	
For I=1 to 10	حلقة التكرار Loop	
	اتجاه سير العمليات Flow line	

الشكل(٢-١٤) – يبين بعض الأشكال الهندسية الخاصة

٢-٧- فوائد استخدام المخططات التدفقية

جاءت المخططات التدفقية كضرورة لتسهيل عمل المبرمج عندما تصبح الخوارزمية أكثر تعقيداً، أي عندما تزداد خطواتها والمقارنات المنطقية فيها، والمخطط التدفقي هو تمثيل بياني (رسومي) للخوارزمية الذي يعطينا تمثيلاً جيداً لها ويُستخدم في هذه المخططات رموز وأشكال هندسية متنوعة لها دلالات محددة، وفيما يلي بعض فوائد استخدام المخططات التدفقية:

١. المخططات التدفقية تمكن المبرمج من الإلمام الكامل بالمسألة المراد حلها والتحكم بكل أجزائها بحيث تساعده على اكتشاف الأخطاء المنطقية (Logic Error) والتي تُعد من أهم الأخطاء التي تجهد المبرمج.

٢. المخططات التدفقية تساعد وبسهولة المبرمج على تعديل البرامج الموضوعة بمجرد النظر إليها.

٣. يُعد الاحتفاظ بالمخططات التدفقية لحلول مسائل معينة أمراً مهماً إذ يكون مرجعاً عند إجراء تعديلات عليها أو استخدامها لحل مسائل أخرى مشابهة دون الحاجة للرجوع إلى المبرمج الأول باعتبار أن الحلول الأولى قد صيغت في خطوات واضحة بسيطة و مفهومة.

٤. المخططات التدفقية توفر وسيلة مناسبة ومساعدة في كتابة البرامج ذات التفرعات الكثيرة.

٢-٨- أنواع المخططات التدفقية

بعد أن تعرفنا على مفهوم المخططات التدفقية وفوائد استخدامها، يمكننا الآن تصنيفها إلى عدة أنواع هي التالية:

- المخططات التدفقية التتابعية البسيطة (Simple sequential Flowchart).
- المخططات التدفقية التفرعية (Branched Flowchart).
- المخططات التدفقية الحلقية (Loop Flowchart).

ويمكن للبرنامج الواحد أن يشتمل على أكثر من نوع واحد من هذه الأنواع .

٢-٨-١ – المخطط التدفقي التتابعي البسيط

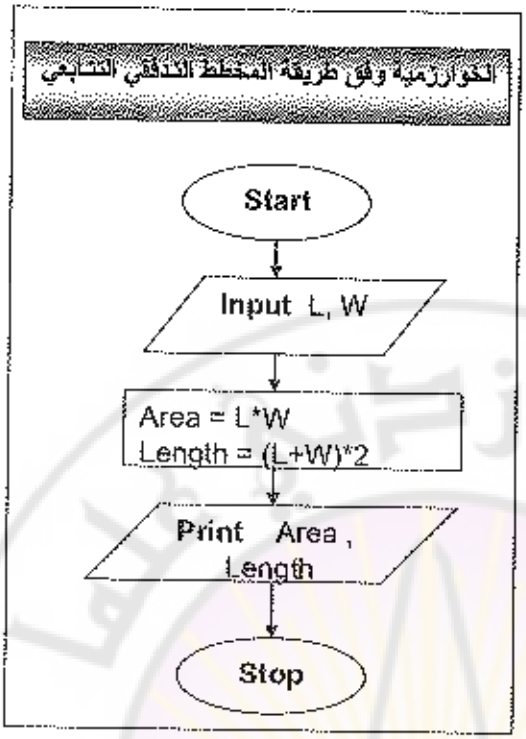
يُستخدم هذا النوع من المخططات في المسائل التي يقتضي حلها تتالياً محدداً للخطوات التي تؤدي إلى النتيجة دون الحاجة إلى تغيير سياق التنفيذ ، وتنفذ هذه الخطوات المحددة والمعرفة خطوة خطوة حتى الوصول إلى النهاية، دون تجاهل أو تكرار أي من الخطوات الموجودة، أي أن هذا النوع من المخططات يخلو من التفرعات والتكرارات أو اتخاذ القرارات ، والأمثلة التالية توضح لنا كيفية استخدام المخططات التدفقية التتابعية.

مثال (٢-٦)

ارسم المخطط التدفقي اللازم لحساب وطباعة مساحة ومحيط مستطيل طوله L وعرضه W، ثم اكتب الخوارزمية بطريقة Pseudo code .

يبين الشكل (٢-١٥) المخطط التدفقي اللازم لحساب وطباعة مساحة ومحيط المستطيل ، أما الخوارزمية وفق طريقة Pseudo code كما يلي :

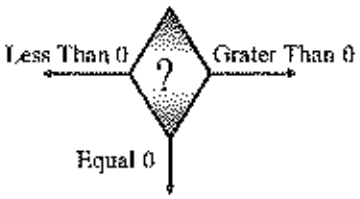
1.Start
2. Input L, W
3. Area=L*W
4. Length=(L+W)*2
5.Print Area, Length
6.Stop

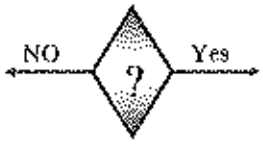


الشكل(٢-١٥) – يبين المخطط التدفقي التتابعي للمثال(٢-٦)

٢-٨-٢- المخطط التدفقي التفرعي

يستخدم هذا النوع من المخططات في المسائل التي تخضع لشروط تحدد التتالي المناسب للخطوات المطلوب تنفيذها، حيث يفرض تحقق الشرط أو عدمه ، الاختيار بين طريقين أو عدة طرق ، وبمعنى آخر فإن هذا النوع من المخططات يتضمن اتخاذ القرارات أو المفاضلة بين خيارين أو أكثر وهناك أسلوبان في تنفيذ القرار هما: قرار ذو تفرعين و قرار ذو ثلاثة تفرعات ، ويبين الشكل(٢-١٦) أساليب تنفيذ القرار .





قرار ذو ثلاثة تفرعات　　قرار ذو تفرعين

الشكل(٢-١٦) – يبين أساليب تنفيذ القرار

ويقدم المثال (٢ – ٧) إحدى الخوارزميات التي يمكن توصيفها بالمخططات التدفقية التفرعية .

مثال (٢-٧)

ارسم المخطط التدفقي اللازم لحساب وطباعة قيمة الدالة الرياضية المعرفة كما يلي : $Y = \frac{4x + 6}{6x - 8}$ ، وذلك من أجل قيمة وحيدة للمتغير x ، ثم اكتب الخوارزمية بطريقة Pseudo code .

يبين الشكل (٢-١٧) المخطط التدفقي اللازم لحساب قيمة الدالة الرياضية ، أما الخوارزمية وفق طريقة Pseudo code كما يلي :

1.Start

2. Input x

3.A=4x+6

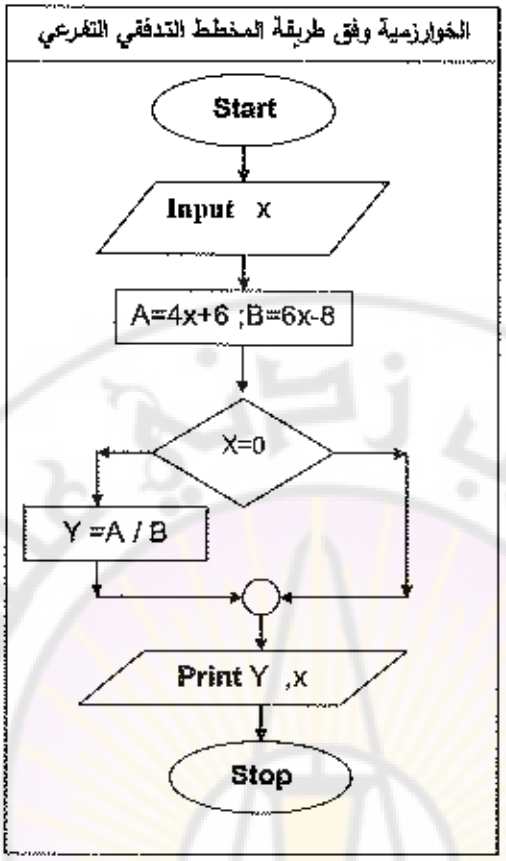
4. B=6x-8

5.if B=0 then go to 2

6.Y=A/B

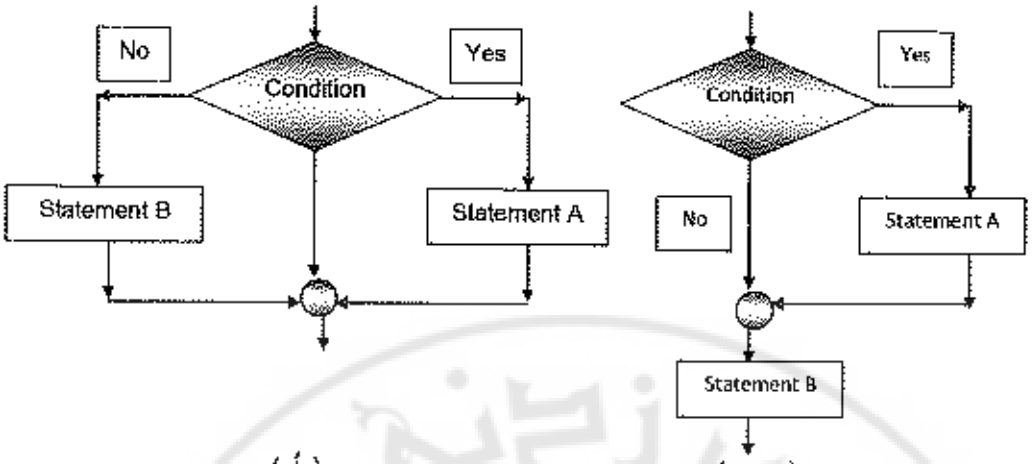
7.Print Y, x

8.Stop



الشكل(٢-١٧) – يبين المخطط التدفقي التفرعي للمثال(٢-٧)

وبشكل عام فإن المخطط التفرعي يمكن أن يأخذ إحدى الصورتين الآتيتين كما هو موضح في الشكل (٢-١٨) . يمكننا ملاحظة أن الشكل (٢- ١٨ - أ) يبين أنه إذا كان جواب الشرط (Condition) YSE فإن التعليمة التالية في التنفيذ تكون Statement A ، أما إذا كان الجواب NO فإن التعليمة التالية تكون Statement B، كما يمكننا أن نلاحظ في الشكل (٢- ١٨ - ب) أنه إذا كان جواب الشرط YSE فإن التعليمة التالية في التنفيذ تكون Statement A ثم يتبعها Statement B ،أما إذا كان جواب الشرط NO فإن التعليمة التالية تكون Statement B مباشرة.

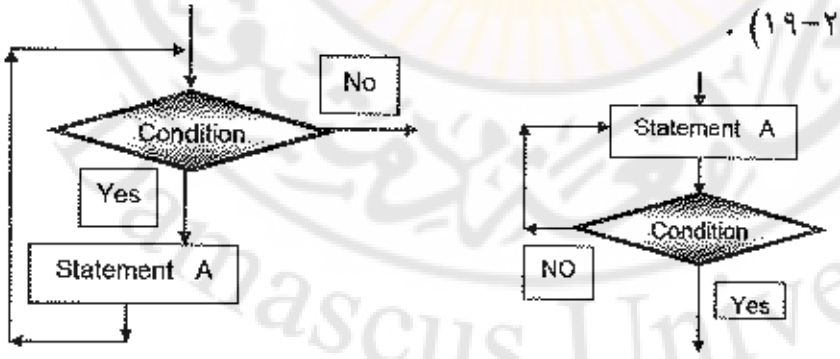


(أ) (ب)

الشكل(٢-١٨) – يبين أساليب التفرع

٢-٨-٣- المخطط التدفقي الحلقي

يستخدم هذا النوع من المخططات في المسائل التي يتضمن حلها تكرار تعليمة واحدة أو مجموعة تعليمات، أكثر من مرة، حيث يحدد الشرط الذي يقرر عدد المرات التي يجب تكرارها بوساطة صناديق الاختبار المرتبطة بزيادة أو إنقاص متحول يسعى إلى تحقيق الشرط المحدد في صندوق الاختبار، وبشكل عام فإن المخطط الحلقي يمكن أن يأخذ احد الشكلين الآتيين كما هو موضح في الشكل (٢-١٩) .



الشكل(٢-١٩) – أساليب التكرار

مثال (٢-٨)

ليكن لدينا 10 طلاب ونرغب في إدخال درجات هؤلاء الطلاب في ثلاثة مقررات ثم حساب معدل كل طالب في المقررات الثلاثة ، والمطلوب:

- ارسم المخطط التدفقي الذي يقوم بما يلي:
  - ❖ إدخال درجات الطلاب العشر في المقررات الثلاثة.
  - ❖ حساب معدل كل طالب.
  - ❖ طباعة المعدل مع الدرجات لكل طالب.
  - ❖ ثم اكتب الخوارزمية بطريقة Pseudo code .

يبين الشكل (٢-٢٠) المخطط التدفقي اللازم ، أما الخوارزمية وفق طريقة Pseudo code كما يلي :

1.**Start**

2.I=1

3. **Input** A,B,C

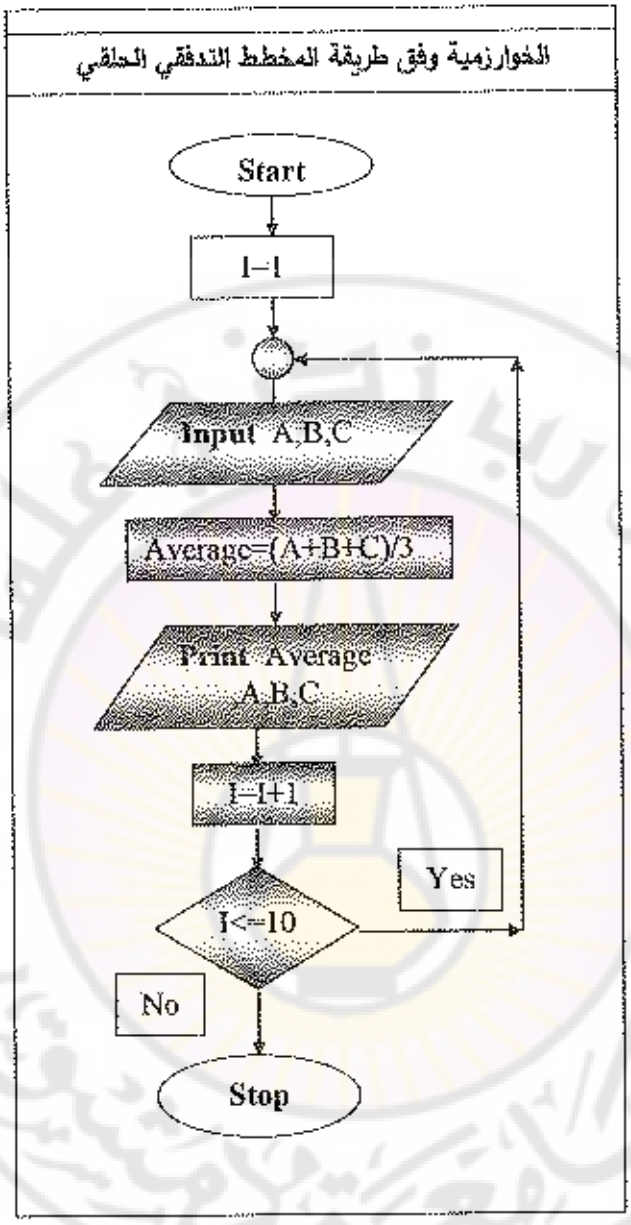
4.Average = (A+B+C)/3

5. **Print** Average , A , B , C

6.I=I+1

7.**if** I<=10 then go to 3

8.**Stop**



الشكل(٢-٢٠) – يبين المخطط التدفقي الحلقي للمثال(٢-٨)

٥-٩- العداد Counter

في كثير من الأحيان نحتاج في برامجنا الحاسوبية إلى مفهوم العدّ Counting، فقد نريد مثلاً أن نعد عدد مرات تحقق شرط ما عند حل مسألة معينة، وقد تكون هذه العملية سهلة للإنسان لأنها أصبحت ضمن قدراته العقلية التي يكتسبها من الطفولة، إلا أن الحاسوب يحتاج إلى تصميم خوارزمية للعد Counting Algorithm تتضمن خطوات معينة إذا اتُبعت يستطيع عندئذ القيام بمهمة العد. ويمكن تحديد الخطوات التي يتبعها الحاسوب حتى يتمكن من العد كما يلي:

١. نجعل العداد مساويًا للصفر.

٢. نجعل القيمة الجديدة للعداد مساوية للقيمة القديمة له زائد واحد ، أي أن:

قيمة العداد (الجديدة)= قيمة العداد (القديمة)+١

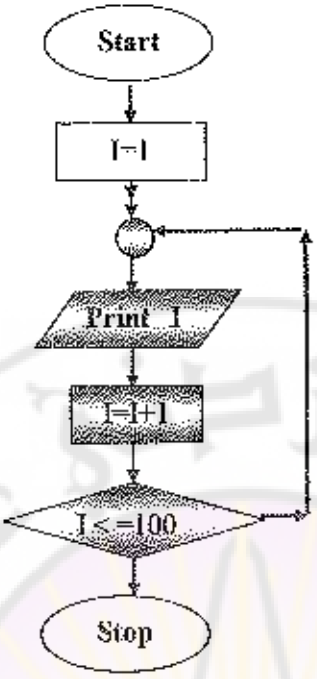
٣. كرر الخطوات ابتداء من الخطوة ٢.

مثال (٢-٩)

ارسم المخطط التدفقي اللازم لطباعة الأعداد المحصورة بين العددين 1 و 100.

يبين الشكل(٢-٢١) المخطط التدفقي اللازم . أما الخوارزمية وفق طريقة Pseudo code كما يلي :

```
1.Start
2.I=1
3. Print I
4.I=I+1
5.if I <=100 then go to 3
6.Stop
```



الشكل(٢-٢١) – يبين المخطط التدفقي الحلقي للمثال(٢-٩)

٢-١٠- المجاميع الإجمالية

في حالات كثيرة تحتاج بعض البرامج الحاسوبية إلى جمع مجموعة كبيرة من الأعداد التي تمثل معطيات ظاهرة معينة، فمثلاً قد نرغب في إيجاد المتوسط الحسابي لأعمار طلاب الجامعة، ولتحقيق هذا أولاً يجب أن نحسب مجموع أعمار الطلاب، وطبعاً ليس عملياً إعطاء رمز أبجديّ لكل عمر طالب فقد تحتاج لأكثر من عشرة آلاف رمز، في مثل هذه الحالات نصمم خوارزمية معينة للتجميع تسمى خوارزمية التجميع Summers Algorithm تتضمن خطوات محددة إذا اتبعها الحاسوب استطاع أن يجمع أي كمية من المعطيات باستخدام متغيرين اثنين أحداهما المتغير الذي نجمعه والآخر هو الجمع الإجمالي (المجمع)، ويمكن تحديد الخطوات التي يجب أن يتبعها الحاسوب لتحقيق ذلك في أربع خطوات هي:

١. اجعل المجمع مساوياً الصفر.

٢. ادخل قيمة واحدة للمتغير.

٣. اجعل القيمة الجديدة للمجمع تساوي القيمة القديمة له زائد القيمة المدخلة للمتغير، أي أن:

قيمة المجمع الجديدة = قيمة المجمع القديمة + آخر قيمة مدخلة للمتغير

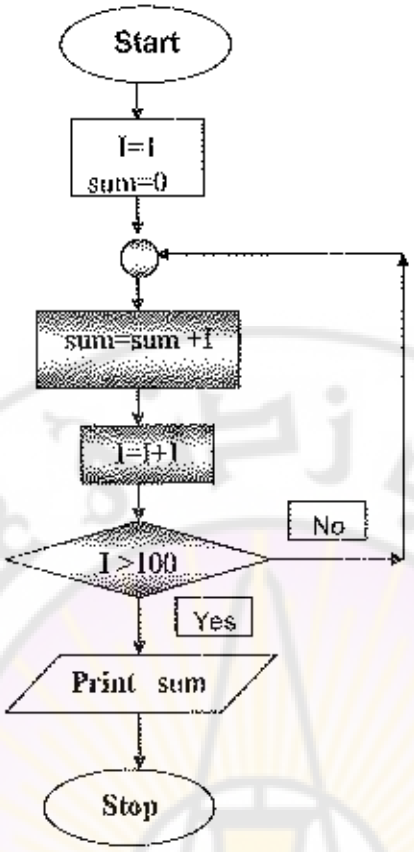
٤. كرر ابتداءً من الخطوة الثانية.

مثال (٢-١٠)

ارسم المخطط التدفقي اللازم لحساب وطباعة مجموع الأعداد المحصورة بين العددين 1 و 100، ثم اكتب الخوارزمية بطريقة Pseudo code .

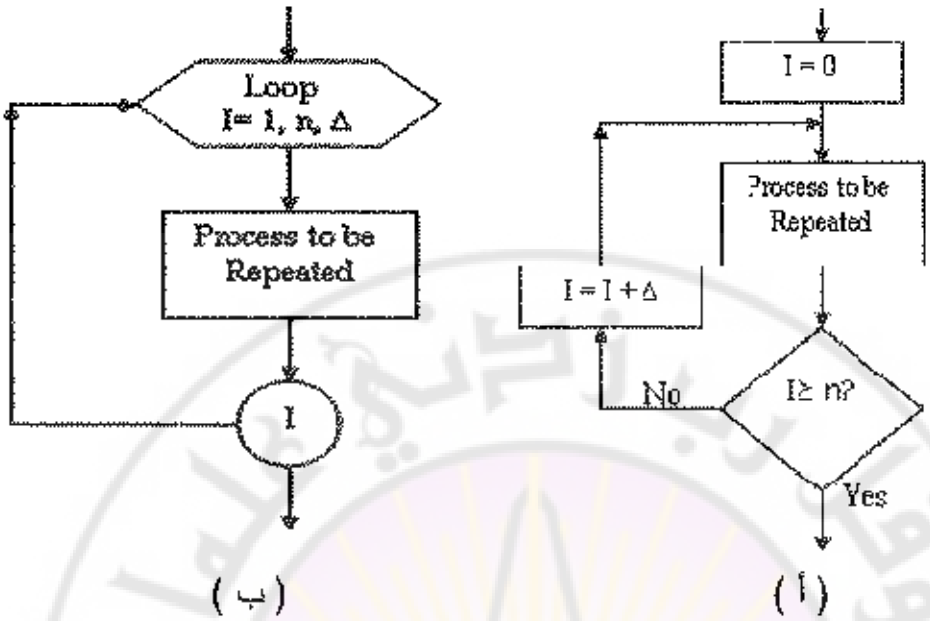
يبين الشكل(٢-٢٢) المخطط التدفقي اللازم . أما الخوارزمية وفق طريقة Pseudo code كما يلي :

```
1.Start
2.I=1
3.sum=0
4.sum=sum +I
5.I=I+1
6.if I >100 then go to 8
7. go to 4
8. Print sum
9.Stop
```



الشكل(٢-٢٢) يبين طريقة المخطط التدفقي للمثال(٢-١٠)

ويمكننا استخدام الشكل الاصطلاحي للتكرار كما هو موضح في الشكل (٢-٢٣).



الشكل(٢-٢٣) – يبين الشكل الاصطلاحي للتكرار

نلاحظ في الشكل(٢-٢٣) أننا نحتاج إلى العناصر الآتية:

- ✓ القيمة الأولية للعداد I (هنا I=1).
- ✓ القيمة النهائية للعداد I (هنا n).
- ✓ قيمة الزيادة عند نهاية كل دورة (هنا Δ)

نلاحظ في الشكل(٢-٢٣-أ) إن إجراءات التكرار كانت تتم طبقاً للخطوات الآتية والمفصلة من قبل المبرمج:

١. أعط I قيمة أولية.

٢. أتم الإجراءات المطلوب إعادتها.

٣. (تقرير) إذا كانت قيمة العداد I وصلت إلى القيمة النهائية n اخرج إلى الخطوة التالية في البرنامج وإلا فاذهب إلى الخطوة (٤).

٤. زد I بمقدار الزيادة Δ .

٥. عد إلى الخطوة (٢).

يمكننا استبدال الخطوات المفصلة(1,2,3,4,5) في الشكل(٢-٢٣-أ) بخطوة مجملة واحدة مبينة في الشكل (٢-٢٣- ب) الذي يبين الشكل الاصطلاحي للتكرار حيث تنفذ هذه الخطوات بصورة أوتوماتيكية من قبل الحاسوب، وهذا من شأنه تسهيل عملية البرمجة واختصار عدد التعليمات في البرنامج وتجنب بعض الأخطاء.

ملاحظة

تُعد قيمة Δ تساوي 1 دائمًا إذا لم تعط قيمة أخرى بخلاف ذلك، وفي حالة عدم ذكر قيمة Δ يصبح الشكل الاصطلاحي الوارد في الشكل(٢-٢٤) كما يلي حيث تكون قيمة Δ تساوي 1 وبصورة أوتوماتيكية.

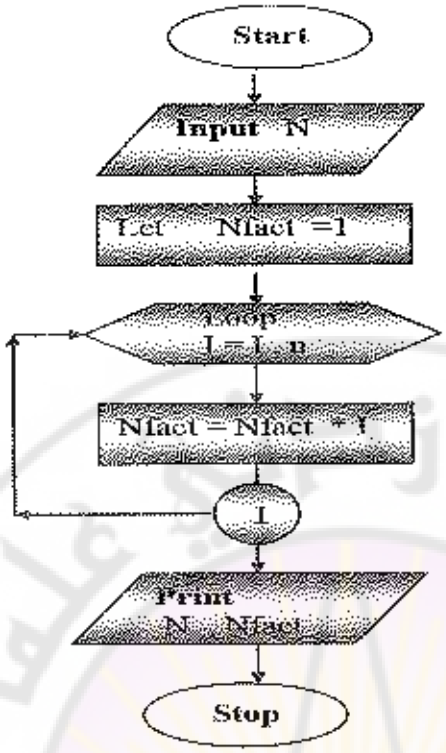
الشكل(٢-٢٤) – يبين الشكل الاصطلاحي للتكرار

مثال (٢-١١)

ارسم المخطط التدفقي لإيجاد قيمة العاملي للعدد N (N!).

الحل:

N!=N (N-1) (N-2) ...3*2*1 , خطوات الحل مبينة في الشكل(٢-٢٥) .



الشكل(٢-٢٥) - يوضح المخطط التدفقي لحساب !N للمثال(٢-١١)

٢-١١- تطبيقات على الخوارزميات

فيما يلي نستعرض مجموعة من التمارين التي بدورها تساعد الطالب على فهم أنواع المخططات التدفقية بشكل جيد، حيث إننا عند طرح التمرين لا نأتي على ذكر المخطط التدفقي اللازم استخدامه، والذي يحدد ذلك هو طبيعة المسألة.

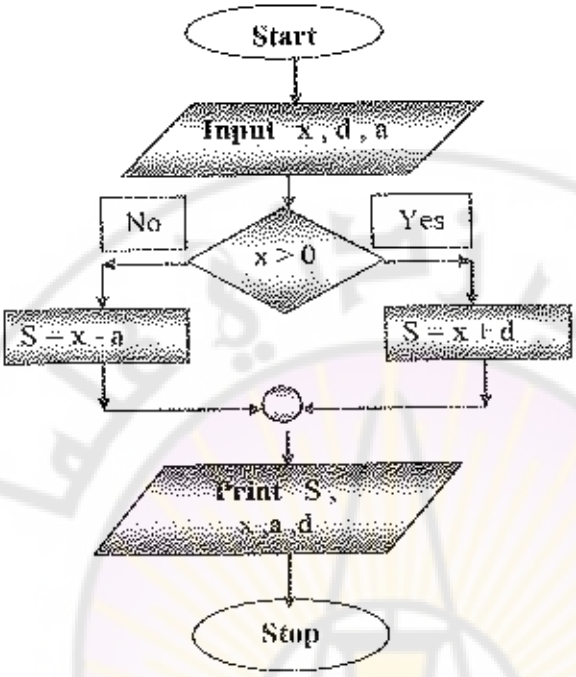
مثال (٢-١٢)

ارسم المخطط التدفقي اللازم لحساب وطباعة قيمة الدالة الرياضية التالية:

$$s=\begin{cases} x+d \ldots\ldots if..x>0 \\ x-a \ldots\ldots if..x\leq 0 \end{cases}$$

الحل:

يبين الشكل (٢- ٢٦) المخطط التدفقي .

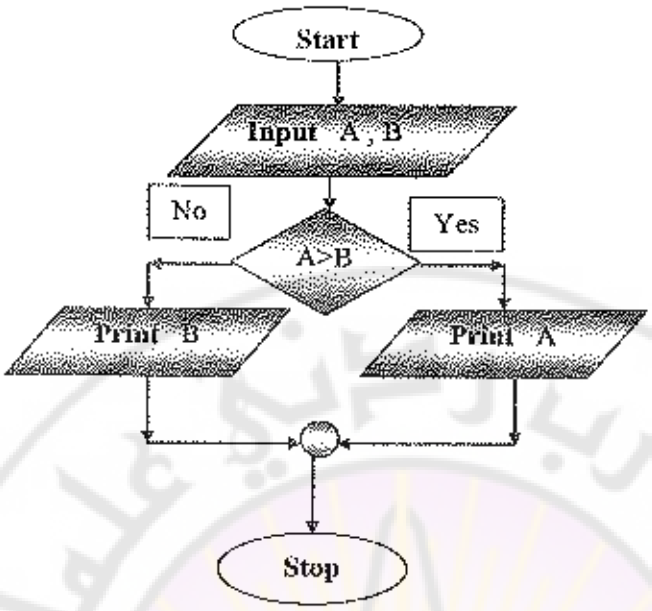


الشكل (٢- ٢٦) – يبين المخطط التدفقي للمثال(٢-١٢)

مثال (٢-١٣)

ارسم المخطط التدفقي اللازم لإيجاد وطباعة العدد الأكبر والأصغر من بين عددين مُدخلين A و B .

الحل: يبين الشكل (٢- ٢٧) المخطط التدفقي اللازم .



الشكل (٢- ٢٧) – يبين المخطط التدفقي للمثال(٢-١٣)

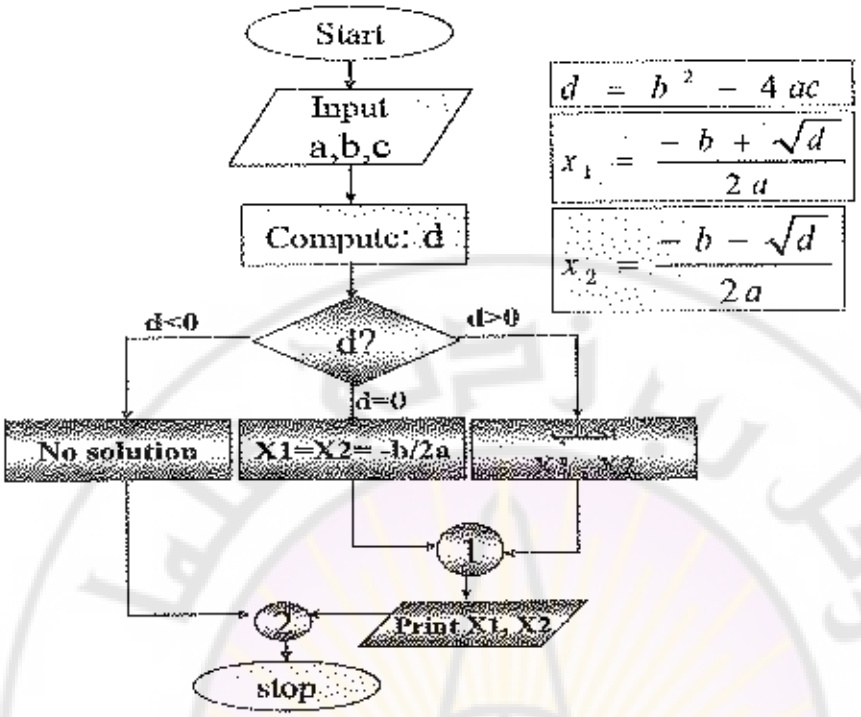
مثال (٢-١٤)

ارسم المخطط التدفقي اللازم لحساب وطباعة جذور معادلة من الدرجة الثانية والتي لها الشكل التالي:

$$ax^2 + bx + c = 0$$

<u>الحل:</u>

يبين الشكل (٢- ٢٨) المخطط التدفقي اللازم .



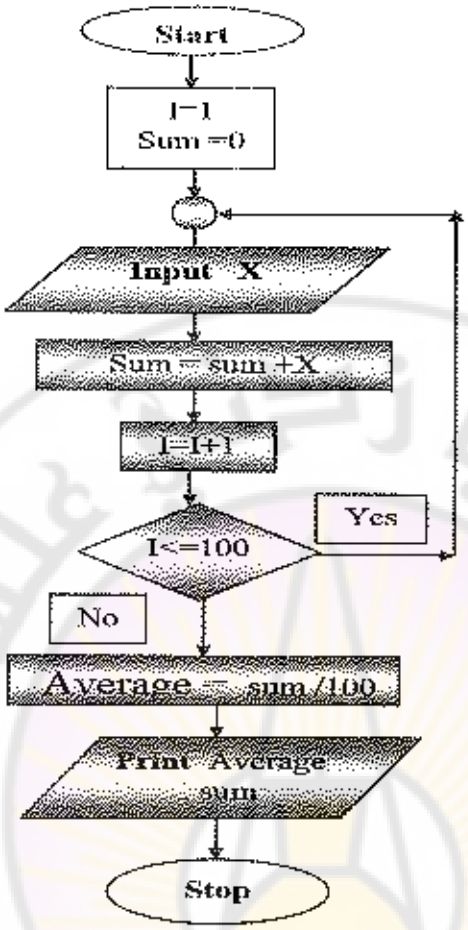
الشكل (٢- ٢٨) – يبين المخطط التدفقي للمثال(٢-١٤).

مثال (٢-١٥)

ارسم المخطط التدفقي اللازم لحساب وطباعة المتوسط الحسابي لـ 100 عدد صحيح يتم إدخالها من قبل المستخدم (ليست الأعداد المدخلة متتالية).

الحل:

يبين الشكل (٢- ٢٩) المخطط التدفقي اللازم



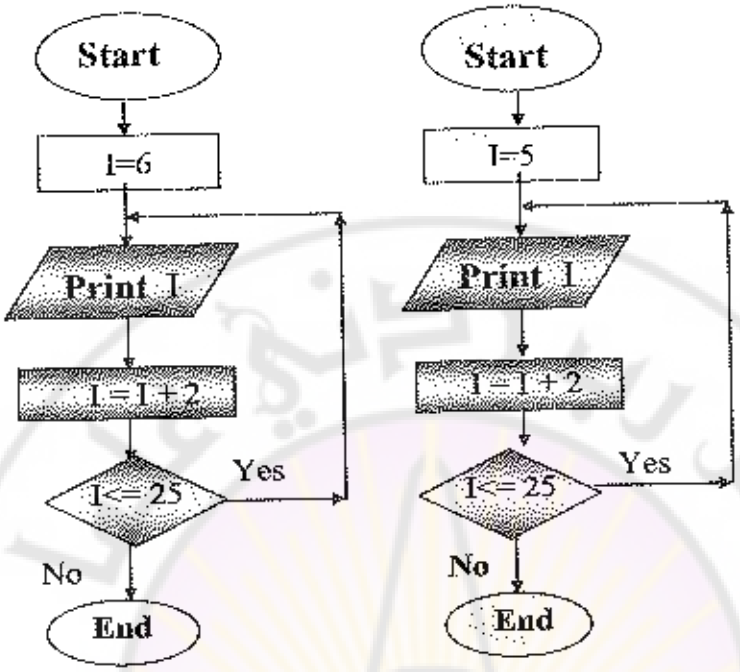
الشكل (٢- ٢٩) – يبين المخطط التدفقي للمثال(٢-١٥)

مثال (٢-١٦)

ارسم المخطط التدفقي اللازم لطباعة الأعداد الفردية من 5 حتى 25 .

أعد السؤل من أجل طباعة الأعداد الزوجية من 5 حتى 25 .

<u>الحل:</u> يبين الشكل (٢- ٣٠) المخطط التدفقي اللازم .



الشكل (٢- ٣٠) -- يبين المخطط التدفقي للمثال(٢-١٦)

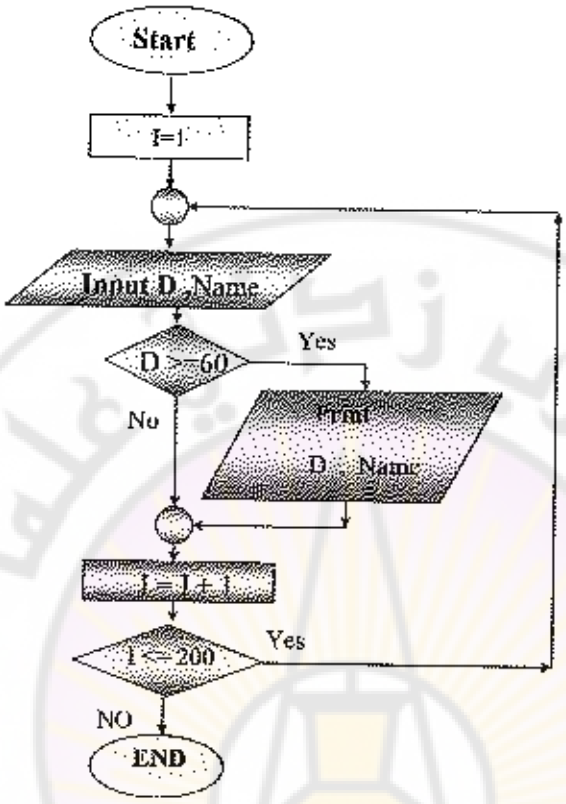
مثال (٢-١٧)

يبلغ عدد طلاب السنة الأولى في قسم تقنيات الحاسوب 200 طالباً في مقرر البرمجة(١) ، والمطلوب :

اكتب خوارزمية (المخطط التدفقي) لإدخال أسماء ودرجات جميع الطلاب ويطبع فقط أسماء ودرجات الطلاب الناجحين .

الحل:

يبين الشكل (٢- ٣١) المخطط التدفقي اللازم .



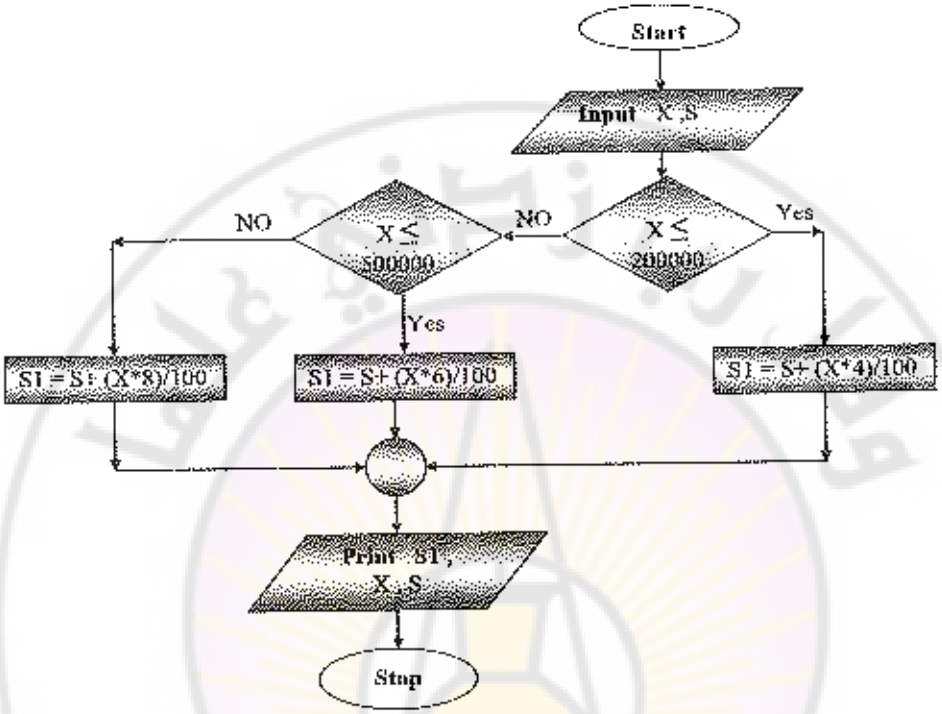
الشكل (٢- ٣١) – يبين المخطط التدفقي للمثال(٢-١٧)

مثال (٢-١٨)

رجل مبيعات يتقاضى راتباً شهريا مقداره S= 10000 ليرة سورية ويتقاضى عمولة مقدارها 4% إذا كانت المبيعات الشهرية لا تتجاوز 200000 ليرة سورية ويتقاضى عمولة مقدارها 6% إذا كانت المبيعات الشهرية لا تتجاوز 500000 وإذا زادت المبيعات عن 500000 فان نسبة العمولة تصبح 8 % والمطلوب : اكتب خوارزمية (المخطط التدفقي) لحساب وطباعة الدخل الشهري الكامل للموظف, علماً أن الراتب الشهري الكلي يساوي :

الراتب الشهري الكامل = الراتب الأساسي + العمولة (S1 = S + W)

الحل: يبين الشكل (٢- ٣٢) المخطط التدفقي اللازم .



الشكل (٢- ٣٢) – يبين المخطط التدفقي للمثال(٢-١٨)

مسائل عامة

١- ارسم المخطط التدفقي اللازم لحساب وطباعة المتوسط الحسابي لـ 500 عدد حقيقي يتم إدخالها من قبل المستخدم (ليس من الضروري أن تكون الأعداد المدخلة متتالية).

٢- ارسم المخطط التدفقي اللازم لحساب مجموع الأعداد الفردية الصحيحة المحصورة بين العددين n و m (حيث n<m وأن n فردي).

٣- شركة تجارية يبلغ عدد موظفيها 300 موظفاً ويتقاضون رواتب شهرية مختلفة والمطلوب:

ارسم المخطط التدفقي اللازم لقراءة أسماء ورواتب كل الموظفين ومن ثم طباعة أسماء ورواتب الموظفين الذين تزيد رواتبهم الشهرية عن 10000ليرة سورية.

٤- ارسم المخطط التدفقي اللازم لحساب وطباعة قيمة المقدار Y والمعرف كما يلي:

$$Y = \sum_{i=1}^{n} \frac{i}{i+1}$$

$$\ldots = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \ldots + \frac{n}{n+1}$$

٥- ارسم المخطط التدفقي اللازم لإدخال عدد حقيقي R من لوحة المفاتيح ويطبع كلمة Positive إذا كان العدد موجباً ويطبع كلمة Negative إذا كان عدد سالباً .

٦-- يرغب قسم هندسة الحواسيب والأتمتة بمنح مكافأة مالية قدرها 10000 ليرة سورية لكل طالب في السنة الأولى يحصل على درجة 90 أو أكثر في مقرر مدخل إلى الحاسوب والبرمجة والمطلوب :

ارسم المخطط التدفقي اللازم لإدخال أسماء ودرجات الطلاب ويطبع فقط أسماء ودرجات الطلاب الذين يستحقون المكافئة .

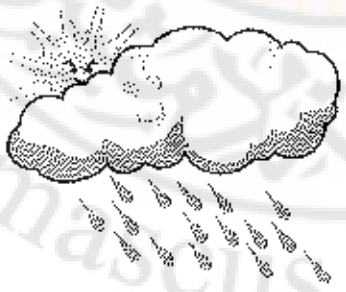
٧- ارسم المخطط التدفقي اللازم لحساب وطباعة مجموع مربعات الأعداد من 1 إلى 100 .

٨- في شركة 50 مندوب مبيعات يتقاضى كل مندوب راتباً شهريا مقداره S= 10000 ليرة سورية ويتقاضى عمولة مقدارها 4% إذا كانت المبيعات الشهرية لا تتجاوز 200000 ليرة سورية ويتقاضى عمولة مقدارها 6% إذا كانت المبيعات الشهرية لا تتجاوز 500000 , وإذا زادت المبيعات عن 500000 فان نسبة العمولة تصبح 8 % والمطلوب :

اكتب خوارزمية (المخطط التدفقي) لحساب وطباعة الدخل الشهري الكامل للموظف, علماً أن الراتب الشهري الكلي يساوي :

الراتب الشهري الكامل = الراتب الأساسي + العمولة (S1 = S + W)

==========



الفصل الثالث

أساسيات لغة C#

Basics of C# language

أساسيات لغة C#

٣-١- مقدمة

يقوم المبرمجون بكتابة برامجهم الحاسوبية بلغات برمجية مختلفة، بعضها مفهوماً مباشرة من قبل الحاسوب وبعضها الآخر يحتاج إلى خطوات وبرامج ترجمة ، ويتوافر حالياً العشرات من لغات البرمجة المستخدمة ، والتي يمكن تقسيمها إلى نوعين رئيسين:

١- لغات البرمجة منخفضة المستوى(LLL) , مثل :

- لغة الآلة Machine Language.
- لغة التجميع Assembly Language .

٢- لغات البرمجة عالية المستوى(HLL)

- لغة FORTRAN
- لغة C/C++
- لغة Java
- لغة C#

تساعد لغات البرمجة عالية المستوى المبرمجين على كتابة تعليمات قريبة من لغة الإنسان ، وقد أصبحت اللغات البرمجية العالية المستوى أكثر تقبلاً من قبل المبرمجين , وتعد اللغات C/ C++ و Java و C# من أكثر لغات البرمجية عالية المستوى قوةً وانتشاراً.

تُعد لغة C# من أشهر اللغات التي تتمتع بطابع القوة والمرونة لإنتاج أسرع البرامج وأفضلها أداءً ، وعلى الرغم من وجود العديد من لغات البرمجة الأخرى إلا أنها تفتقر إلى شمولية لغة C# وقوتها . وتتميز لغة C# بقدرتها على معالجة التطبيقات الكبيرة والمعقدة، وتتميز بالقوة في صيانة البرامج المكتوبة بها ما يوفر وقتاً في تصميم البرامج وتطويرها .

تُعد لغة #C لغة برمجة غرضية التوجه OOP طورتها شركة Microsoft ، وتستخدم هذه اللغة منصة عمل (.NET) وبالتالي تملك مجموعة واسعة من المكونات البرمجية ، ما يجعلها أداة برمجية خصبة جداً ، وباستخدام C# يمكن أن ننشئ وبسرعة تطبيقات Windows ، أو تطبيقات ويب أو تطبيقات طرفيات أو برامج شبكة أو تطبيقات قواعد البيانات أو مكتبات برمجية .

وبما أن لغة #C لغة برمجة غرضية التوجه ، فهذا يعني أن برنامج #C يتكون من مجموعة من الأغراض Objects التي تتواصل مع بعضها بعضاً في زمن التشغيل . ويتم توصيف هذه الأغراض بوساطة الصفوف Classes ويتم تعريفها عند كتابة البرنامج . وتمتلك لغة #C ميزات استثنائية لمعالجة الأخطاء وإدارة مؤتمتة للذاكرة وتدعى هذه الميزة بجمع النفايات garbage collection وكل هذا لتسهيل تطوير التطبيق .

٣-٢- تطور لغة #C

في عام 1960 صُممت لغة CPL (Combined Programming Language) ، وتم ذلك في جامعة لندن وقد استخدمت فيها بعض تعليمات لغة ALGOL-60، وفي عام 1967 انبثقت لغة BCPL من لغة CPL ، وفي عام 1972 قامت مخابر شركة AT&T الأمريكية باستنباط لغة جديدة من لغة B أُخذ منها أحسن تعليماتها وأُضيف إليها أوامر جديدة وأنواعاً جديدة من المعطيات ، وكثيراً من التوابع التي تفيد المبرمج وسُميت هذه اللغة بلغة C . ومنذ ذلك التاريخ أخذت لغة C شهرة واسعة لأنها أصبحت تنتمي إلى اللغات البرمجية عالية المستوى High Level Languages من حيث سهولة الاستخدام من ناحية ومن ناحية ثانية لأنها تنتمي إلى اللغات البرمجية المتدنية المستوى من حيث قدرة اللغة على مخاطبة مكونات الحاسوب العتادية Hardware.

بعد ذلك تم تطوير لغة البرمجة C++ التي تُعد امتداداً للغة C وقد قدمت لغة C++ العديد من الإمكانيات التي تسمو بلغة C وتعلو بها، ولكن الأهم من ذلك كله قدرات البرمجة غرضية التوجه OOP (Object Oriented Programming).

وتم تطوير لغة Java من قبل فريق من المهندسين من شركة Sun Microsystems في عام 1991 وكانت تحمل الاسم Oak لتستخدم في تضمين التطبيقات الالكترونية للاستهلاك ، وفي عام 1995 أعيدت تسمية هذه اللغة لتصبح Java كما أعيد تصميمها لتستخدم في تطوير تطبيقات الانترنت ، حيث من الممكن لتطبيقات Java أن تُضمن في صفحات HTML ثم تُحمل باستخدام مستعرضات الويب لتحقيق عملية التحريك الحية ، والتفاعل المباشر مع مستخدمي الويب ولا تنحصر قوة جافا في تطبيقات الويب بل هي لغة برمجة ذات أهداف عامة ،حيث تمتلك صفات برمجية كاملة ويمكن استخدامها في بناء تطبيقات قائمة بحد ذاتها، وعلى الرغم من أن العديد من اللغات غرضية التوجه بدأت بشكل كامل كلغات إجرائية فإن لغة Java تم تصميمها من البداية لتكون غرضية التوجه.

طورت النسخة الأولى من لغة C# من قبل فريق صغير من مبرمجي مايكروسوفت ، يقوده مهندسان بارزان هما : Andres Hejsberg و Scott Wiltamuth ، ويُعد المهندس Andres مبتكر لغة Turbo Pascal ، وقد كان قائداً للفريق الذي صمم لغة Delphi من إنتاج شركة Borland .

إن الهدف من لغة C# إيجاد لغة سهلة آمنة وذات أداء عالٍ تدعم البرمجة غرضية التوجه بهدف بناء تطبيقات ويندوز و ويب بشكل أساسي ، وتملك لغة C# عدداً قليلاً نسبياً من الكلمات المحجوزة Keywords وهذا ما يجعلها سهلة التعلم وسريعة التكيف مع احتياجاتنا البرمجية ، وقد صممت لغة C# خصيصاً لمنصة التطوير NET. ، ورغم أن البرامج المكتوبة لمنصة التطوير NET. تعمل

حالياً على الويندوز إلا أنه يُنتظر في المستقبل أن تعمل البرامج بشكل مستقل عن نظام التشغيل ويندوز . وفي الواقع توجد حالياً مشاريع مفتوحة المصدر هدفها جعل منصة التطوير NET. تعمل على نظام التشغيل Linux وقد نجحت في ذلك إلى حدٍ مقبول ، فمنصة التطوير NET. عبارة عن إطار برمجي يمكننا من خلاله تطوير تطبيقات وبرامج تعمل على نظام تشغيل ويندوز بطريقة جديدة ومختلفة ، وتسعى لتسهيل إنشاء تطبيقات تعمل على الويب بسرعة وسهولة ومرونة كبيرة .

٣- ٣- بيئة لغة C#

تُعدّ البرمجة غرضية التوجه طريقة قوية للتعامل مع المسائل البرمجية ، فقد ظهرت في كل نقطة من مراحل تطورها طريقة جديدة تساعد المبرمجين على التعامل مع البرامج متزايدة التعقيد .على الرغم من أن البرمجة البنيوية كانت تعطي نتائج ممتازة عندما تطبق على برامج متوسطة التعقيد ، إلا أنها كانت تفشل عند بعض النقاط حين يصل البرنامج إلى حد معين . ولكتابة برامج أكثر تعقيداً كانت الحاجة إلى طريقة جديدة في العمل البرمجي وتلبية لهذه الحاجة ظهرت البرمجة غرضية التوجه واختصاراً " OOP " . تأخذ البرمجة OOP أفضل ميزات البرمجة البنيوية وتضم إليها مبادئ قوية وجديدة تسمح بتنظيم البرامج بفعالية أكبر ، وتعتمد البرمجة OOP على تحليل المشكلة إلى أجزائها الأساسية بحيث يصبح كل عنصر أو جزء عبارة عن غرض object محتوى ذاتياً يضم تعليماته الخاصة والمعطيات العائدة إليه ، وأصبح المبرمج يستطيع إدارة برامج أكبر . تشترك جميع لغات البرمجة OOP بما فيها C++ و Java و C# بثلاث سمات مميزة مشتركة وهي :

١. التغليف Encapsulation(الصفوف Classes) .
٢. الوراثة Inheritance .
٣. تعدد الأشكال Polymorphism .

تُعد لغة C# من اللغات القوية جداً في مجال إنشاء التطبيقات المختلفة سواء كانت هذه التطبيقات تعمل منفردة على أجهزة الحاسوب الشخصي أو تطبيقات الإنترنت أو التطبيقات المختلفة للأجهزة المحمولة ، مثل الموبايل والمفكرات الإلكترونية وغيرها. إن لغة #C ليست مجرد مفردات برمجية وتراكيب نحوية ، وإنما هي لغة منتخبة كأفضل أداة برمجية لتطبيق مبادئ هندسة البرمجيات ، إضافة إلى أنها لغة تطوير متكاملة يمكننا استخدامها لبناء العديد من التطبيقات مثل :

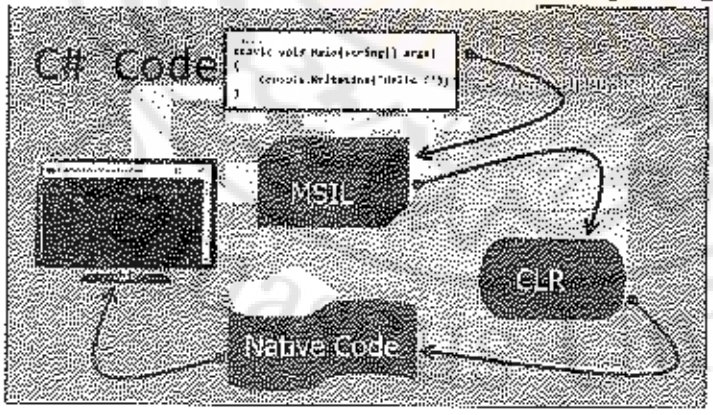
- تطبيقات سطح المكتب Desktop Applications
- التطبيقات الشبكية Client/Server Applications
- تطبيقات الويب Web-Based Applications
- التطبيقات الموزعة Distributed Applications
- تطبيقات المحمول Mobile Applications

ولغة #C هي لغة بسيطة ، غرضية التوجه، وموزعة ، ومفسرة ، وقوية ، وآمنة ، وذات بنية محايدة ، ومحمولة ، وذات أداء عالٍ ، ومتعددة المسالك وديناميكية . تشبه لغة #C لغة C++ ولغة Java المبسطة ، ما يقلل من وقت تعلمها ، وإن لغة C++ لغة هجينة (غرضية وبنيوية) بينما لغة #C غرضية التوجه OOP تماماً . وتحقق #C معظم مبادئ هندسة البرمجيات مثل إعادة الاستخدام وقابلية التعديل ، والفعالية ، والوثوقية ، والتغليف ، وجعلت #C من انتقال الفيروسات أمراً مستحيلاً ، فهي لا تسمح بإنشاء مؤشرات خارج ترميزها الخاص ، إذ أن المؤشرات الخارجية التي لا تسمح بها #C هي التي تساعد على عمل الفيروسات، فالضرر يحدث عندما يخرج الترميز عن مساحة الذاكرة الخاصة به ، حيث إن تطبيقات #C لا تتفاعل مباشرة مع المعالج أو نظام التشغيل ، فبيئة تشغيل #C تعالج مسائل الذاكرة ذاتياً بحيث لا يحتاج للقيام بتخصيص الذاكرة ،

أو تفريغ الشيفرة منها ، كذلك لا حاجة لاستخدام المؤشر Pointer والذي يعد أهم مصادر الأخطاء في C++ ، فتقدم C# حلاً جذرياً وتلقائياً لمشكلة الذاكرة التي نصادفها في برامج C++ كما تستبدل C# التوريث المعقد المتعدد في C++ بالواجهات Interfaces .

☐ زمن تشغيل اللغة الوسيطية والمشتركة

يتم تنفيذ برامج NET. ضمن بيئة مدارة managed تدعى بيئة زمن تشغيل اللغة المشتركة CLR (Common Language runtime) ، ولا تتم ترجمة البرامج المكتوبة من أجل منصة عمل NET. مباشرة إلى شيفرة الآلة ، كما هو الحال بالنسبة للبرامج المكتوبة بلغات برمجة أخرى ، وبدلاً من ذلك تتم ترجمتها إلى أوامر رقمية Numeric Instruction تقوم CLR بترجمتها في زمن التشغيل إلى أوامر بلغة الآلة ، ثم يتم تنفيذها . وتُعد هذه الأوامر جزءاً من لغة تدعى اللغة الوسيطية MSIL (Microsoft Intermediate language) وتفهم جميع لغات NET. كيف تتعامل مع الـ MSIL ، ما يمكّن برامج نظام النوع المشترك Common Type System المكتوبة بلغات NET. أخرى من استخدام وظائف مكتوبة بلغة C# (مثلاً Visual Basic.NET) ، ويبين الشكل(٣-١) مراحل الترجمة بلغة C# .



الشكل(٣-١) – مراحل الترجمة بلغة C#

٣-٤- الشكل العام لبنية البرنامج بلغة C#

يتكون أي برنامج مكتوب بلغة C# من أربعة عناصر أساسية هي التالية:

١- استدعاء المكتبات المستخدمة في البرنامج.

٢- تعريف المعطيات المستخدمة في البرنامج.

٣- تعليمات (عبارات) البرنامج.

٤- توابع(دوال)المستخدم إن وجدت فهي تكتب ضمن التابع main .

الشكل العام لبنية برنامج بلغة C# :

```
1.  using System ;
2.  namespace  Example1 {
3.  class    ClassName
4.     {
5.       static  void   Main( string args[ ] )
6.        {

      body {          Body of main( )

7.        } // end method main
8.     } // end class Name
9.  } //end namespace
```

مثال(٣-١)

لنكتب أول برنامج بلغة C# والذي يقوم بطباعة العبارة :

Welcome to C# Programming !

```
1. //This program will display a message on the screen.
2. using System;
3. namespace ConsoleApplicationy1
4. {
5. public class Welcome
6. {
7. public static void Main(string[] args)
8. Console.WriteLine("Welcome to C# Programming \n");
9. Console.ReadKey();
10.   } // end method main
11.   } // end class Name
12.  }//end namespace
```

وعند تنفيذ البرنامج السابق نحصل على المخرج التالي:

```
RESULT
Welcome to C# Programming
Press any key to continue . . .
```

في هذا المثال تم تعريف صف اسمه Welcom وهذا الصف عام public ويحمل اسم الملف ، ويبدأ الصف بالقوسين " { " ، " } " وينتهي بهما، ولا ينفذ التطبيق دون وجود التابع ( )Main وهو من نوع :

```
public static void Main( string [ ] args)
```

١- التعليقات Comments

```
//This program will display a message on the screen.
```

يبدأ هذا السطر من البرنامج بالشرطة المزدوجة (//) الدالة على أن بقية السطر عبارة عن تعليق (comment)، تضاف التعليقات إلى البرامج لتساعد المبرمج أو أي شخص آخر قد يحتاج إلى قراءة البرنامج على فهم ما الذي يفعله البرنامج، لذا

من المستحسن أن يبدأ كل برنامج في لغة #C بتعليق يوضح الغرض الذي من أجله كُتب البرنامج. وتُستخدم الشرطة المزدوجة (//) إذا كان التعليق يمتد لسطر واحد فقط single-line comment ، وهنالك نوع آخر من التعليقات يتيح لنا كتابة تعليقات تمتد إلى عدة أسطر multi-line comments ، نستطيع كتابة التعليق السابق كما يلي:

```
/*
    This program will display
    a message on the screen
*/
```

يبدأ الرمز " /* " ثم التعليق وينتهي بالرمز " */" ، ونجد أن نهاية السطر لا تعني انتهاء التعليق لذا يمكننا كتابة ما نشاء من أسطر التعليقات قبل الانتهاء بالرمز " */ " ، ويمكن اختيار نوع التعليق الذي نراه مناسباً .

٢- فضاءات الأسماء namespaces

تملك البرامج الضخمة عادة الكثير من الصفوف ، وهذا يعني أنه من الوارد جداً أن يكون لصفين الاسم نفسه ، عندها يصبح المترجم مشوشاً لا يعرف ما هو الصف الذي نتحدث عنه ، ولمنع المترجم من الالتباس نقوم بتنظيم الصفوف ضمن فضاءات أسماء ، ولكتابة الاسم الرسمي للصف نقوم بإلحاق الصف باسم فضاء الأسماء الذي يتضمن هذا الصف ، فمثلاً : إذا كان اسم فضاء الأسماء في الصف Welcome هو Example1 فإن الصف Welcomeسيحمل رسمياً الاسم :

Example1.Welcome ، ما يجعله مميزاً عن الصف Welcome الموجود في إطار منصة عمل .NET ، وهو فضاء الأسماء System.Drawing وبالتالي فإن الاسم الرسمي له هو : System.Drawing.Welcome (السطر 2) .

نستخدم في السطر 8 الصف Console ، وهو موجود في فضاء الأسماء System . ونحتاج في الحالة الافتراضية إلى الاسم الكامل للوصول إلى الصف Console أي إلى System.Console، ولكن من المتعب أن نكتب (System.) في كل مكان ، لذلك يمكننا اختصار الأمر بأن نعلم المترجم أننا نستخدم فضاء الأسماء System في برنامجنا ، وبذلك يمكننا أن نستخدم مباشرة أي صف ثم التصريح عنه ضمن فضاء الأسماء System بدون أن نكون مضطرين لوضع الاسم الرسمي كاملاً .

توضع تعليمات using في أعلى البرنامج قبل أية تصريحات أو تعليمات أخرى كما هو مبين في السطر 2 من البرنامج (using System) .

يمكن إنشاء فضاءات خاصة بنا ، ويتم ذلك باستخدام الكلمة المفتاحية namespace متبوعة باسم فضاء الأسماء الجديدة ، ثم زوج من الأقواس { } كما هو مبين في البرنامج في السطر 3 حيث فضاء الاسم الجديد هو Example1 وكل صف يتم تعريفه ضمن القوسين يكون منتمياً إلى فضاء الأسماء الجديد هذا .

يجب أن يتضمن كل برنامج C# صفاً واحداً على الأقل ، لقد قمنا في المثال السابق في السطر 5 بتعريف صف اسمه Welcome ويوضع جسم الصف ضمن قوسين أي من السطر 6 (فتح القوس) إلى السطر 11 (إغلاق القوس).

٣- التابع ReadKey()

يقوم بإيقاف الشاشة إلى حين الضغط على أي زر من لوحة المفاتيح(السطر 9) .

٤- التابع أو الطريقة Main( )

هناك دائماً في كل برنامج تابع اسمه Main ، وهو يُعد نقطة البدء في البرنامج ، وهذا التابع مميز لأن CLR تستدعيه بعد تحميل البرنامج إلى الذاكرة ،

وبمجرد أن تقوم CLR باستدعاء التابع Main ينطلق تنفيذ البرنامج وعندما يعيد التابع Main ينتهي البرنامج كما هو مبين في البرنامج السابق .

```
1. public static void Main(string[] args)
2. {
3. Console.WriteLine("Welcome to C# Programming \n");
4. Console.ReadKey();
5. } // end method main
```

ويمكن القول إن التابع Main هو كتلة الشيفرة التي تعرف CLR وإن عليها استدعاؤها عندما نبدأ بتنفيذ البرنامج ، وباعتبار أن على CLR استدعاؤه ، فمن الضروري أن نقوم بتعريف التابع Main ساكناً static ويجب أن يكون اسمه Main بحرف M كبير ، وليس صغيراً main كما هو الحال في لغة C++ ولغة Java ، لأن لغة C# تميز بين الأحرف الكبيرة والصغيرة . وبما أن التابع يمكن أن يعيد معطيات ، فمن الضروري أن يحدد كل تابع نوع المعطيات التي سيعيدها ، فإذا كان التابع لا يعيد معطيات فمن الضروري تحديد نوع الإعادة void ، وبما أن التابع Main في المثال السابق لا يعيد بيانات فإن نوع الإعادة هو void.

ويتضمن جسم التابع Main في المثال السابق سطرين من الشيفرة وهو السطر 8 و السطر 9 ، ويتم فيه استحضار التابع Main من الصف Console . أما التابع WriteLine فيقوم بطباعة سلسلة المحارف التي بين علامتي الاقتباس المزدوجة على الشاشة كما هي وهو موجود ضمن الصف Console والانتقال إلى سطر جديد ، أما إذا كتبنا التابع Write فيقوم بطباعة سلسلة المحارف دون الانتقال إلى سطر جديد ، أما متوالية الهروب \n فهي تجبر المترجم إلى

الانتقال إلى سطر جديد ، ويجب أن تنتهي كل تعليمة في برنامج C# بفاصلة منقوطة " ; " (semi colon).

إذا كتبنا السطر 8 من البرنامج السابق كما يلي :

```
Console.WriteLine(" Welcome \n to C# \n Programming \n");
```

عند تنفيذ البرنامج سوف نحصل على المخرج التالي :

```
RESULT
Welcome
 to C#
 Programming
Press any key to continue . . .
```

٣-٥- الإدخال والإخراج بوساطة تطبيقات Console

هناك حاجة لوجود طريقة لإدخال المعطيات من المستخدم وإظهار النتائج على الشاشة ، ونحتاج أيضاً إلى تحديد تنسيق الخرج ، وذلك باستخدام عرض للحقل ورموز التنسيق التي يمكن من خلالها عرض المعطيات كعملة أو أية تنسيقات أخرى للحصول على واجهة تحقق أكثر تفاعلاً مع المستخدم .

استخدمنا في المثال السابق التابع WriteLine الخاص بالصف Console لطباعة نص (سلسلة محرفية) text على الشاشة وقد قمنا بطابعة النص :

Welcome to C# Programming !

يقوم التابع WriteLine بطباعة النص والانتقال إلى سطر جديد . قد لا نرغب أحياناً بالانتقال إلى سطر جديد بعد طباعة النص وانما الاستمرار بالطباعة على السطر نفسه عندها نستخدم التابع Write .

٣-٥-١- توابع الإخراج

من المفيد أحياناً أن تتم طباعة النص على الشاشة بدون الانتقال إلى سطر جديد ، لذلك نستخدم التابع Write من الصف Console لأداء هذه المهمة .

مثال(٣-٢)

```
//This program will display a message on the screen.
using System;
namespace ConsoleApplicationy1
{
public class Welcome
{
public static void Main(string[] args)
Console.Write ("Welcome to C# Programming \n");
Console.ReadKey();
} // end method main
} // end class Name
}//end namespace
```

يقوم البرنامج بطباعة النص Welcome to C# Programming وكذلك العبارة Press any key to continue على سطر واحد كما يلي :

```
RESULT
Welcome to C# Programming Press any key to continue
```

يمكننا جعل التابع Write يتصرف كالتابع WriteLine وذلك بوضع محرف الانتقال إلى سطر جديد في نهاية السلسلة المحرفية ، حيث إن محرف الانتقال

إلى سطر جديد (\n) هو محرف خاص يميزه المترجم عن غيره ويعني " انتقل إلى السطر التالي " ويصبح السطر 8 من البرنامج كما يلي :

```
Console.Write (" Welcome to C# Programming \n");
```

يوضح البرنامج في المثال(١-٣) كيف يتم طبعة رقم وحرف .

```
1.  /* Example2  :This program will display a character and integer
    number on the screen  */
2.  using System ;
3.  class    Program2
4.    {
5.  static  void  Main( string [ ] args )
6.        {
6.   Console.Write (7 + " is an integer.\n ") ;
7.   Consolc.Write ('a' + " is charachter.\n ") ;
7.     } // end method main
8.   } // end class Name
```

عند تنفيذ البرنامج سوف نحصل على الخرج التالي :

```
RESULT
7  is an integer.
a  is a character .
Press any key to continue . . .
```

من خرج البرنامج يتضح ما يلي:

١- يتم حصر النص المطلوب ظهوره على الشاشة بين علامتي اقتباس مزدوجة " is an integer " .

٢- تتم كتابة الثوابت الرقمية بدون علامتي اقتباس مثلاً العدد 7.

```
Console.Write (7 + " is an integer.\n ") ;
```

٣- يتم حصر حرف واحد مطلوب ظهوره على الشاشة بعلامة اقتباس فردية المحرف ' a '.

```
Console.Write ('a' + " is charachter.\n ") ;
```

مثلاً : تقوم بعض اللغات كـ Basic بالانتقال إلى سطر جديد تلقائياً في نهاية كل عبارة خرج ، لكن C# لا تفعل ذلك كما أن العبارات المختلفة والموضوعة في أسطر مختلفة لا تؤدي إلى ذلك .

٣-٥-٢- متواليات الهروب Escape Sequences

نلاحظ في المثال(١-٢) أن السلسلة المحرفية في تعليمة Write تنتهي بـ \n وهي تُعلِم الصف Console أن ينتقل إلى السطر التالي ، والمحرف \n هو نوع خاص من المحارف التحكمية، و \ تسمى الشرطة الخلفية(Back slash) أو حرف هروب (Escape character) وتسمى هي والحرف الذي يليها متواليات الهروب.

متواليات الهروب \n يعني الانتقال إلى سطر جديد حيث يجبر المؤشر على الانتقال إلى بداية السطر التالي ، وهناك محارف تحكمية أخرى مثل محرف الجدولة ومحرف تمرير صفحة وغيرها، ونعرض بعض متواليات الهروب الشائعة في الجدول (٣-١) .

متوالية الهروب	الوصف
\n	الانتقال إلى بداية سطر جديد
\t	وضع مسافة أفقية بين قيم الخرج
\b	حرف التراجع back space.
\\	لطباعة شرطة خلفية
\r	حرف الإرجاع، يجبر المؤشر على الانتقال إلى بداية هذا السطر
\"	لطباعة علامة اقتباس

الجدول(٣-١) – بعض متواليات الهروب

يمكن استخدام محرف الهروب ليس فقط من أجل إخبار المترجم أن عليه التعامل مع المحارف العادية على أنها محارف تحكم ، ولكنه يستخدم أيضاً لإخبار المترجم أن عليه التعامل مع محارف خاصة محرفية كمحارف عادية . مثلاً علامات الاقتباس المزدوجة ، تُستخدم لبدء وإنهاء السلسلة المحرفية ، فإذا وضعت شرطة خلفية أمام علامة الاقتباس المزدوجة " Hello \ " C# " فإن المترجم سيتعامل معها على أنها محرف عادي وستتم طباعته على الشاشة أي طباعة(") وليس كمحرف إنهاء السلسلة المحرفية . مثلاً :

```
Console.Write (" Hello \" C# ") ;
```

هناك ثلاث مجموعات من علامات الاقتباس المزدوجة في التصريح عن السلسلة المحرفية ، تقوم الأولى والأخيرة كالمعتاد ببدء وإنهاء السلسلة المحرفية ، ولكن علامة الاقتباس المزدوجة في الوسط هي في الواقع جزء من السلسلة المحرفية ، لذلك علينا إخبار المترجم أن يتعامل معها كمحرف عادي لذلك وضعنا شرطة خلفية قبلها ، وسوف يتم طباعة النص (Hello " C#) .

عندما نريد طباعة الشرطة الخلفية كشرطة خلفية فإننا بحاجة لطريقة نخبر بها المترجم أن يتعامل معها كمحرف عادي في الحالات التي نريد فيها فعلياً أن تكون الشرطة الخلفية جزءاً من السلسلة المحرفية . تصادفنا هذه الحالة كثيراً عندما نقوم بتخزين أسماء الملفات كسلاسل محرفية . لأن المسارات إلى أسماء الملفات تتضمن شرطات خلفية لفصل أسماء المجلدات ، وتستخدم شرطة خلفية مزدوجة لإخبار المترجم أن يتعامل مع الشرطة الخلفية كمحرف عادي .

مثلاً : لتكن لدينا السلسلتين المحرفيتين التاليتين :

```
string  s1 = " C: \\ test.txt " ;
string  s2 = " C: \  test.txt " ;
```

عند طباعة هاتين السلسلتين سنحصل على الخرج التالي :

```
C: \  test.txt

C:     est.txt
```

المشكلة بالنسبة للملف الثاني هو أن المترجم يتعامل مع الشرطة الخلفية كمحرف هروب ، وبالتالي يتم تحويل حرف t الذي يليها إلى محرف جدولة tab . باختصار ، علينا وضع محرف هروب تليه شرطة خلفية أخرى إذا أردنا أن تكون الشرطة الخلفية جزءاً من السلسلة المحرفية بدلاً من أن تُستخدم كمحرف هروب . وعندما نرغب بتخزين اسم ملف ضمن متحول من النوع **string** ، علينا استخدام شرطة خلفية مزدوجة بعد أسماء المجلدات ، وإلا فإن المترجم سيعتقد أننا نحاول استخدام محرف هروب . **قد تكون الشرطات الخلفية مصدر إزعاج** ، فقد قررت مايكروسوفت استخدام الرمز @ كإشارة إلى المترجم تقول بأن السلسلة هي سلسلة محرفية لا تتضمن أية محارف هروب ، وبالتالي يصبح المثال السابق كما يلي :

```
string  s1 =   " C: \\ test.txt " ;
string  s2 = @"  C: \  test.txt " ;
```

أي أن وجود المحرف @ في بداية السلسلة المحرفية تخبر المترجم أنه لا توجد أية محارف هروب في هذه السلسة ، وبالتالي يتم التعامل مع الشرطة الخلفية كجزء من السلسلة المحرفية .

٣-٥-٣- تنسيق الخرج

يمتلك التابعان Write و WriteLine ميزة سهلة الاستعمال تتيح لنا تمرير المتحولات كوسطاء إضافية يمكن تنسيقها في الخرج ، ويبين المثال (٣-٣) كيف يتم الاستدعاء .

مثال(٣-٣)

```
1. using System ;
2. class    Program3  {
3. static  void  Main( string [ ] args )  {
4. string   name = " Waseem  younes ";
5. double  temprature = 37.5;
6. Console.WriteLine(" Name {0} \n temperature {1}" ,
                                  name ,temperature);
7. Console.WriteLine( "{0} \n {1}", "Welcome to",
                                  "C# Programming!" );
7.       } // end method main
8.   } // end class Name
```

تم التصريح في هذا المثال عن متحول من النوع string اسمه name لتخزين الاسم ، ثم عن متحول من نوع double اسمه temperature وأسندنا له القيمة 37.5 ، ثم قمنا بوضع محددي موضع (place holder)

ضمن السلسلة في التابع WriteLine ، حيث محدد الموضع الأول هو {0} وسيتم ملؤه بالمتحول الأول الذي يلي السلسلة المحرفية ، أما محدد الموضع الثاني فهو {1} وسيتم ملؤه بالمتحول الثاني الذي يلي السلسلة المحرفية ، ويظهر الخرج كما يلي :

```
RESULT
Name        Waseem  younes
temperature  37.5
Press any key to continue . . .
```

يمكن أن تتضمن محددات الموضع عرض الحقول ورموز تنسيق لتحديد كيفية عرض المتحولات . حيث يوضع عرض الحقل ضمن قوسين {...} ويفصل عن الرقم الأول بفاصلة ، ويبين المثال (٣-٤) كيف يتم تحديد عرض الحقل من أجل الخرج .

مثال(٣-٤)

```
using System ;
class    Program4  {
static  void  Main( string [ ] args )  {
int    x = 20 , y = 30  ;
Console.WriteLine(" ( X , Y ) is ( {0,2} , {1,2} )",
                                                x, y);
     } // end method main
  } // end class Name
```

```
RESULT
( X , Y ) is (20 , 30 )
Press any key to continue . . .
```

ويوضح البرنامج الجزئي التالي تنسيق المخرج :

```
Console.WriteLine("{0,6}", 123);
Console.WriteLine("{0,6}", 1234);
Console.WriteLine("{0,6}", 12);
Console.Write("{0,-6}", 123);
Console.WriteLine("--end");
```

```
RESULT
   1234
     12
123    --end
Press any key to continue . . .
```

١-تنسيقات قياسية للأرقام Standard Formats for Numbers

يتم تعريفها بواسطة أحد محددات التنسيق المتعددة ، وهي أحرف ذات أهمية خاصة، يمكن أن يكون هناك عدد صحيح موجب بعد محدد التنسيق يسمى الدقة precision ، والذي له معنى مختلف للمتغيرات المختلفة. وعندما يؤثر على عدد الخانات العشرية بعد الفاصلة العشرية ، يتم تقريب النتيجة. ويصف الجدول (٣-٢) المحددات والدقة الخاصة بها .

الرمز	الوصف	الاستخدام	المخرج
C	عملة	{0,7:C}	$ 200.00
P	نسبة مئوية	{0:P}	50.00%
E	أسي	{0:E1}	1E+3
F	فاصلة عشرية	{0:F1}	44.2
D	عشري	{0:D5}	00340
G	عام	{0:G4}	5.456E+05
X	سداسي عشر	{0:X}	86FD3

جدول(٣-٢) -- تنسيقات قياسية للأرقام

ويوضح المثال(٣-٥) التنسيقات القياسية للأرقام .

مثال(٣-٥)

```
class StandardNumericFormats {
static void Main(string [] args) {
Console.WriteLine("{0:C2}", 123.456);
//Output: $123,46
Console.WriteLine("{0:D6}", -1234);
//Output: -001234
Console.WriteLine("{0:E2}", 123);
//Output: 1,23E+002
Console.WriteLine("{0:F2}", -123.456);
//Output: -123,46
```

```
Console.WriteLine("{0:N2}", 1234567.8);
//Output: 1 234 567,80
Console.WriteLine("{0:P}", 0.456);
//Output: 45,60 %
Console.WriteLine("{0:X}", 254);
//Output: FE
}//end main
}// end class
```

٢-تنسيقات مخصصة للأرقام

يتم تعيين كافة التنسيقات غير القياسية إلى تنسيقات المستخدم (مخصص). بالنسبة للتنسيقات المخصصة يتم تعريفها مرة أخرى كمجموعة من المحددات وتختلف عن التنسيقات القياسية في أنه يمكن استخدام عدد من المحددات (في التنسيقات القياسية يتم استخدام محدد واحد فقط). ويوضح الجدول (٣-٣) عدد من المحددات .

Specifier	Description
0	Indicates a digit. If at this position of the result a digit is missing, a zero is written instead.
#	Indicates a digit. Does not print anything if at this position in the result a digit is missing.
.	Decimal separator for the respective "culture".
,	Thousands separator for the respective "culture".
%	Multiplies the result by 100 and prints the character for percent.
E0 or E+0 or E-0	Indicates an exponential notation. The number of zeroes indicates the number of signs of the exponent. The sign "+" means that we always want to represent also the number's sign while minus means to display the sign only if the value is negative.

الجدول (٣-٣) – يوضح عدد من المحددات المخصصة للأرقام

يوضح المثال(٣-٦) التنسيقات المخصصة للأرقام .

مثال(٣-٦)

```
class CustomNumericFormats  {
static void Main()  {
Console.WriteLine("{0:0.00}", 1);
//Output: 1.00
Console.WriteLine("{0:#.##}", 0.234);
//Output: .23
Console.WriteLine("{0:#####}", 12345.67);
//Output: 12346
Console.WriteLine("{0:(0#) ### ## ##}", 29342525);
//Output: (02) 934 25 25
Console.WriteLine("{0:%##}", 0.234);
//Output: %23
} //end main
} // end class
```

٣-تنسيق مكونات سلسلة التاريخ

Format String Components for Dates

عند تنسيق التاريخ، لدينا مرة أخرى فاصل بين التنسيقات القياسية والمخصصة. ونظرًا لأن التنسيقات المحددة القياسية كثيرة ، فلن نعرض سوى عدد قليل منها. يمكن التحقق من الباقي بسهولة على MSDN، ويبين الجدول(٣-٤) تنسيق مكونات التاريخ .

Specifier	Format (for English (United States) "
d	2/27/2012
D	February 27, 2012
t	17:30 (hour)
T	17:30:22 (hour)
Y or y	February 2012 (only month and year)

الجدول(٣-٤) – يوضح تنسيق مكونات التاريخ المحددة القياسية

تشبه تنسيقات مكونات التاريخ المخصصة التنسيقات المخصصة للأرقام وتحتوي على محددات تنسيق متعددة ويمكننا دمج العديد منها ، ويبين الجدول (٣-٥) تنسيق مكونات التاريخ المخصصة .

Specifiers	Format (for English (United States) "culture")
d	Day - from 1 to 31
dd	Day - from 01 to 31
M	Month - from 1 to 12
MM	Month - from 01 to 12
yy	The last two digits of the year (from 00 to 99)
yyyy	Year written in 4 digits (e.g. 2012)
hh	Hour - from 00 to 11
HH	Hour - from 00 to 23
m	Minutes - from 0 to 59
mm	Minutes - from 00 to 59
s	Seconds - from 0 to 59
ss	Seconds - from 00 to 59

الجدول (٣-٥) - يوضح تنسيق مكونات التاريخ المخصصة

عند استخدام هذه المحددات ، يمكننا إدراج فواصل مختلفة بين الأجزاء المختلفة للتاريخ ، مثل "." أو "/". فيما يلي بعض الأمثلة:

```
DateTime d = new DateTime(2012, 02, 27, 17, 30, 22);
Console.WriteLine("{0:dd/MM/yyyy HH:mm:ss}", d);
Console.WriteLine("{0:d.MM.yy}", d);
```

وعند تنفيذ البرنامج بالنسبة للملكة المتحدة UK نحصل على الخرج التالي :

Execution of these examples gives the following result for the U.K. culture:

```
RESULT
  27/02/2012    17:30:22
  27.02.12
Press any key to continue . . .
```

For example if we run the same code in the Bulgarian culture, the result will be different :

```
RESULT
  27.02.2012 17:30:22
  27.02.12
Press any key to continue . . .
```

٥-تنسيق مكونات تعداد السلسلة

التعدادات (الأنواع المدرجة) هي أنواع معطيات يمكن أن تأخذ قيمة واحدة من العديد من القيم المحتملة المحددة مسبقًا (على سبيل المثال ، الأيام السبعة من الأسبوع) ، وقت تحديد أربعة محددات معيارية .ويوضح الجدول(٣-٦) تنسيق مكونات تعداد السلسة

Specifier	Format
G or g	Represents enumeration as a string.
D or d	Represents enumeration as a number.
X or x	Represents enumeration as a number in hexadecimal numeral system and with eight digits.

الجدول(٣-٦) – يوضح تنسيق مكونات تعداد السلسة .

الأمثلة التالية توضح هذا التنسيق :

```
Console.WriteLine("{0:G}", DayOfWeek.Wednesday);

Console.WriteLine("{0:D}", DayOfWeek.Wednesday);

Console.WriteLine("{0:X}", DayOfWeek.Wednesday);
```

وعند تنفيذ البرنامج نحصل على الخرج التالي

```
RESULT
  Wednesday
  3
  00000003
Press any key to continue . . .
```

٣-٥-٤- توابع الإدخال Console Input

يمكن أن يأتي الدخل من المستخدم باستخدام لوحة المفاتيح ، أو من ملف ، أو من مقبس Socket أو منفذ تسلسلي Serial Port أو من أي جهاز دخل آخر . وتحتوي كل لغة برمجة على آلية للقراءة والكتابة بالنسبة للصف Console، والغرض منه هو التحكم بتدفق الدخل في لغة C# ، بوساطة Console.In ، وبوساطة الصف Console يمكن قراءة معطيات مختلفة ، مثل :

- text;
- other types after parsing the text;

في الواقع لقراءة تدفق الدخل القياسي نادراً ما يُستخدم Console.In مباشرة. ويوفر الصف Console طريقتين للإدخال هما :

Console.Read () و Console.ReadLine ()

سنستخدم الآن التابع ReadLine الموجود ضمن الصف Console والذي يقوم بإدخال سلسلة محرفية من المستخدم بوساطة لوحة المفاتيح . في المثال(٣-٧) التابع ReadLine الذي يتيح للمستخدم إدخال اسمه ، ويقوم البرنامج بعدها ببناء رسالة مخصصة للمستخدم يعرضها على الشاشة باستخدام التابع WriteLine والذي يطلب منه إدخال اسمه .

مثال(٣-٧)

```
using System ;
class    Program6  {
static  void  Main( string [ ] args )  {
string    input  = "  " ;
Console.WriteLine(" Enter Your Name");
 input= Console.ReadLine( );
 Console.WriteLine(" Hello," + input +" Welcome to C#
                                    Programming ! ");

    } // end method main
  } // end class Name
```

```
RESULT
 Enter Your Name
 Mamoun
 Hello , Mamoun Welcome to C# Programming !
Press any key to continue . . .
```

إدخال الأرقام :

لا يتم قراءة الأرقام من الصف Console في #C مباشرة، ولكي نقرأ رقماً يجب أن نقرأ الإدخال مسبقاً كسلسلة (باستخدام ()ReadLine) ثم نحول هذه السلسلة إلى رقم، يُطلق على عملية تحويل سلسلة إلى نوع آخر التحويل parsing. وتحتوي جميع الأنواع الأولية طرائق للتحويل. المثال(٣-٨) يقوم بإدخال رقمين صحيحين من المستخدم ويجمعهما و يطبع ناتج الجمع .

مثال(٣-٨)

```
using System ;
class    Program7  {
static  void  Main( string [ ] args )
{
   int   integer1, integer2  ;
  Console.WriteLine("    Enter first Number\n ");
  integer1 = int.Parse( Console.ReadLine() );
  Console.WriteLine("  Enter   second Number\n ");
  integer2 = int.Parse( Console.ReadLine() );
  int   sum = integer1 + integer2;
  Console.WriteLine("  sum =  {0} " , sum);
  } // end method main
  } // end class Name
```

```
RESULT
Enter first Number
66
  Enter   second Number
88
  sum =  154
Press any key to continue . . .
```

٣-٥-٥-إدخال المعطيات المختلفة

إذا كان المطلوب إدخال قيم حقيقية من قبل المستخدم نستخدم الصيغة التالية :

```
X = double.Parse( Console.ReadLine() );
```

حيث نقوم بتحويل السلسلة إلى قيمة حقيقية باستدعاء التابع Parse( ) .

أما إذا كان المطلوب إدخال محرف واحد char من قبل المستخدم نستخدم الصيغة التالية :

```
ch = char.Parse( Console.ReadLine() );
```

هناك طريقة أخرى لإدخال المعطيات المختلفة من المستخدم كما يلي :

```
Convert.ToInt(Console.ReadLine() )
```

لقد قمنا بتحويل السلسلة المحرفية إلى قيمة رقمية بواسطة التابع (Toint(الموجود ضمن الصف Console .

يوضح المثال(٣-٩) كيفية التحويل إلى أنواع المعطيات المختلفة .

مثال(٣-٩)

```
Example19.Arethmatic                                   Main(string[])
using System ;
namespace   Example19 {
class Arethmatic {
public static void Main(string   [ ]args ) {
 double   x , y ;
 char    ch ;
 Console.WriteLine("inter yuor x  &  y & ch ");
 x = double.Parse( Console.ReadLine() );
 y = double.Parse( Console.ReadLine() );
//ch = Convert.ToChar( Console.ReadLine())  ;
ch = char.Parse(Console.ReadLine());
    ch=Convert.
 Console.Writei                    ---------- ");
    Console.Wri                   ') ;
 Console.Writei                    ---------- ");
if( ch == '+')
 Console.WriteL
  //----------
  else if( ch =
 Console.Writei
  //----------
    else  if( c
```

ToBoolean
ToByte
ToChar
ToDateTime
ToDecimal
ToDouble
ToInt16
ToInt32
ToInt64

المثال(٣-١٠) يوضح طريقة الادخال لأنواع مختلفة من المعطيات .

مثال (٣-١٠)

```
class ReadingNumbers
{
static void Main(string []args)
{
Console.Write("a = ");
int a = int.Parse(Console.ReadLine());
Console.Write("b = ");
int b = int.Parse(Console.ReadLine());
Console.WriteLine("{0} + {1} = {2}", a, b, a + b);
Console.WriteLine("{0} * {1} = {2}", a, b, a * b);
Console.Write("f = ");
double f = double.Parse(Console.ReadLine());
Console.WriteLine("{0} * {1} / {2} = {3}",
                  a, b, f, a * b / f);
}// end main
}// end class
```

```
RESULT
a = 5
b = 6
5 + 6 = 11
5 * 6 = 30
f = 7.5
5 * 6 / 7.5 = 4
Press any key to continue . . .
```

٣-٦- أدوات لغة #C

يبنى البرنامج في لغة #C من مجموعة العناصر التالية :

- الحروف الانكليزية الصغيرة a, b , c, d, . . . x, y, z
- الحروف الانكليزية الكبيرة A, B, C, D, . . . X, Y ,Z
- الأرقام العربية 0, 1, 2, 2, . . . 8, 9
- رموز خاصة مثل :

< <= > >= + - * / = # % $

? ; : , . << >> ! ' "

{ } [ ] || & ( )

٣-٧- الكلمات المحجوزة في لغة #C

تستخدم لغة #C مجموعة من الكلمات المحجوزة بعضها تستخدمها لغة C++ و Java وتضيف عدداً من الكلمات الأخرى التي تجسد المبادئ الجديدة في لغة #C ، ونبين في الجدول(٣-٧) مجموعة الكلمات المحجوزة .

٣-٨-العوامل الحسابية والمنطقية والعلائقية

العامل Operator عبارة عن رمز رياضي يقوم بإجراء عملية معينة على معاملاته ويمكن أن تكون عملية حسابية أو منطقية أو عملية مقارنة ، وفي هذه الفقرة سنتعرف على أنواع العوامل التي توفرها لغة #C .

C# Keywords and contextual keywords				
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
volatile	while			
Contextual Keywords				
add	alias	ascending	async	await
by	descending	dynamic	equals	from
get	global	group	into	join
let	on	orderby	partial	remove
select	set	value	var	where
yield				

الجدول(٣-٧) مجموعة الكلمات المحجوزة

٣-٨-١-العوامل الحسابية Arithmetic Operators

تستخدم أكثر البرامج في لغة C# العمليات الحسابية ، مثل : الجمع والطرح وغيرها، ويبين الجدول (٣-٨) العمليات الحسابية ورموزها المقابلة في لغة C# .

العامل	الوظيفة	التعبير الجبري	التعبير في C#
+	جمع	B+h	B+h
-	طرح	B-h	B-h
*	ضرب	Bh	B*h
/	قسمة	B/h	B/h
%	باقي القسمة	B mod h	B%h

الجدول (٣-٨) – يوضح العمليات الحسابية

تنجز العوامل الأربعة الأولى أعمالاً مألوفة لدينا، أما عامل باقي القسمة % المسمى أيضاً المعامل modulus، يتم استعماله لحساب باقي قسمة عدد صحيح على عدد آخر، لذلك فالتعبير 20%3 يساوي 2 . تسمى هذه العوامل الحسابية بالعوامل الثنائية لأنها تعمل على قيمتين ، ويمكن استعمال أكثر من عامل في تعبير رياضي واحد. التعبير الحسابي عبارة عن مجموعة من العمليات الحسابية التي تربط الأعداد والثوابت والمتغيرات والتوابع, ويبين الجدول(٣-٩) بعض التعابير الحسابية.

الشكل الرياضي للتعبير الحسابي	الشكل البرمجي للتعبير الحسابي
$\frac{x+y}{x-y}+c$	$(x+y)/(x-y)+c$
$2\cdot r-6.5$	$2*r-6.5$
$D=b^2-4ac$	$D=b*b-4*a*c$
$y=2x^2+5x-\frac{7}{3}+2x^3$	$y=2*x*x+5*x-\frac{7}{3}+2*x*x*x$

الجدول(٣-٩) – بعض التعابير الحسابية.

مثال(٣-١١)

يوضح البرنامج في المثال(٣-١١) بعض العمليات الحسابية .

```
// Demonstrate the basic arithmetic operators.
using System ;
class BasicMath {
public static void Main(string [ ]args) {
// arithmetic using integers
Console.WriteLine("Integer Arithmetic");
int  a = 1 + 1;
int  b = a * 3;
int  c = b / 4;
int  d = c - a;
int  e = -d;
Console.WriteLine("a = {0} " , a);
Console.WriteLine("b = {0} " , b);
Console.WriteLine("c = {0} " , c);
Console.WriteLine("d = {0} " , d);
Console.WriteLine("e = {0} " , e);
// arithmetic using doubles
Console.WriteLine("\nFloating Point Arithmetic");
double  da = 1 + 1;
double  db = da * 3;
double  dc = db / 4;
double  dd = dc - a;
double  de = -dd;
```

```
Console.WriteLine("da = {0} " , da);
Console.WriteLine("db = {0} " , db);
Console.WriteLine("dc = {0} " , dc);
Console.WriteLine("dd = {0} " , dd);
Console.WriteLine("de = {0} " , de);
// Demonstrate the % operator.
int x = 42;
double y = 42.25;
Console.WriteLine("x mod 10 = " + x % 10);
Console.WriteLine("y mod 10 = " + y % 10);
} // end method main
  } // end class Name
```

```
RESULT
Integer Arithmetic
a = 2
b = 6
c = 1
d = -1
e = 1
Floating Point Arithmetic
da = 2.0
db = 6.0
dc = 1.5
dd = -0.5
de = 0.5
Modulus Operator
x mod 10 = 2
y mod 10 = 2.25
Press any key to continue . . .
```

٣-٨-٢-العوامل العلائقية (المقارنة)

Relational Operators

تقارن العوامل العلائقية قيمتين، وتكون نتيجة التنفيذ إما صحيحة True أو خاطئة False حسب نتيجة المقارنة مع القيمتين ، ويوضح الجدول (٣-١٠) العوامل العلائقية(المقارنة) وما يكافئها في لغة #C.

الرمز	المعنى	مثال
==	يساوي	a==b
!=	لا يساوي	a!=b
>	أكبر من	a>b
<	أصغر من	a<b
>=	أكبر من أو يساوي	a>=b
<=	أصغر من أو يساوي	a<=b

الجدول (٣-١٠) -العوامل العلائقية

تكون التعابير المبينة في الجدول صحيحة أو خطأ وفقا لقيم المتغيرين a و b.

فلنفرض مثلاً أن: a = 9 and b = 10

- التعبير : a == b خطأ.
- التعبير a!=b صحيح وكذلك التعبيرين a < b و a <= b ،

والتعبيرين a > b و a >= b خطأ..

٣-٨-٣-أولوية العمليات الحسابية

يبين الجدول(٣-١١) أولوية العمليات الحسابية في لغة C# .

Operator	Type
( )	الأقواس
fabs( ),pow( ) , sqrt( ) ,sin( ),cos( )	التوابع
++ - -	Unary postfix
++ - - + -	Unary prefix
* / %	multiplicative
+ -	additive
< > >= <=	relational
== !=	equality
? :	conditional
= += -= *= /= %=	assignment

الجدول(٣-١١) – أولوية العمليات الحسابية

يوضح التعبير في المثال (٣- ١٢) والمثال (٣- ١٣) و والمثال (٣- ١٤) أولوية العمليات الحسابية للحصول على النتيجة الصحيحة .

مثال(٣-١٢)

```
1.5 * 2.4 + 3.3 * 4.25 / 5.5
  \_/
   |
  3.6     + 3.3 * 4.25 / 5.5
                \_/
                 |
  3.6     +   14.025    / 5.5
                          \_/
                           |
  3.6     +                2.55
  \_____________________/
            |
           6.15
```

مثال(٣-١٣)

```
1 * 2 + 3 * 5 / 4
  \_/
   |
   2    + 3 * 5 / 4
            \_/
             |
   2    +   15    / 4
                \_/
                 |
   2    +        3
   \_____________/
          |
          5
```

```
1 + 2 / 3 * 5 - 4
    \_/
     |
1 +   0     * 5 - 4
        \_/
         |
1 +     0       - 4
\_____/
   |
   1           - 4
   \_____________/
          |
         -3
```

يبين المثال(٣-١٣) مزيج بين الأعداد الصحيحة والحقيقية .

مثال(٣-١٤)

```
7 / 3 * 1.2 + 3 / 2
  \   /
   2
    \     * 1.2 + 3 / 2
     \   /
     2.4          + 3 / 2
                      \   /
                       1
2.4          +
     \             /
      \           /
       \         /
        \       /
         \     /
          \   /
           3.4
```

تمارين

1. What values result from the following expressions?
   a. 9 / 5
   b. 695 % 20
   c. 7 + 6 * 5
   d. 7 * 6 + 5
   e. 248 % 100 / 5
   f. 6 * 3 - 9 / 4
   g. (5 - 7) * 4
   h. 6 + (18 % (17 - 12))
2. Which parentheses above are unnecessary (which do not change the order of evaluation?)

٣-٨-٤-العوامل المنطقية Logical Operators

تُستخدم العوامل المنطقية لربط أكثر من تعبير منطقي بسيط مع بعضها بعضاً، ويوضح الجدول(٣-١٢) العوامل المنطقية وما يكافئها في لغة #C .

العملية	الرمز الرياضي للعملية	رمز العملية بلغة الـ #C	مثال
و	And	&&	(a>b)&&(a>c)
أو	Or	||	(a>b)||(a>c)
النفي	not	!	!(a>b)

الجدول(٣-١٢) – يوضح العوامل المنطقية

٣-٨-٥ – المعامل الشرطي الثلاثي المحدود

Conditional Operator (?)

يوجد في لغة #C معامل خاص يسمح باختبار قيمة الشرط المنطقي واستخدام واحد من تعبيرين حسب قيمة هذا الشرط وهو المعامل (?) وله الصيغة التالية :

(Condition) ? expression – if – true : expression – if – else

يُستخدم المعامل الشرطي (?) من أجل اختبار شرط معين ويُعيد قيمة إذا تحقق الشرط أو يُعيد قيمة أخرى إذا لم يتحقق الشرط ، ويمكن كتابته بصيغتين :

```
1 :  (condition)  ?   result1 :  result2
2 :  (condition   ?   result1 :  result2 )
```

حيث:

condition – الشرط المنطقي.

result1 – القيمة المعادة إذا تحقق الشرط.

result2 – القيمة المعادة إذا لم يتحقق الشرط.

<u>مثال :</u>

```
y = ( x<5   ) ? 44 : 55 ;
y = ( x<5    ? 44 : 55 ) ;
```

اذا فرضنا أن x=3 فإن قيمة y = 44 .

٣-٨-٦-عوامل الإسناد المركبة

Compound Assignation Operators

يتوافر في لغة C# عدد من عمليات الإسناد، فبالإضافة لعملية الإسناد العادية التي تستخدم فيها الإشارة (=) ، يوجد طرق مختصرة أخرى تسمى عليات الإسناد المركبة ، ويمكن توضيحها من خلال الجدول (٣-١٣).

عملية الإسناد	عملية الإسناد المختصرة
a=a+b	a+=b
a=a-b	a-=b
a=a*b	a*=b
a=a/b	a/=b
a=a%b	a%=b

الجدول (٣-١٣) – يوضح عمليات الاسناد المركبة

٣-٨-٧-عمليات الزيادة والنقصان

Increment and Decrement Operators

إن عامل الزيادة هو (++) ويعني زيادة المتحول بمقدار واحد، بينما عامل النقصان (--) يعني إنقاص قيمة المتحول بمقدار واحد. والزيادة والنقصان يمكن أن يكون من اليسار(الأمام) أو اليمين(الخلف)، أي يمكن وضع ++ أو -- قبل المتحول (الزيادة أو النقصان من اليسار) أو يمكن وضع ++ أو -- بعد المتحول (الزيادة أو النقصان من اليمين) . من حيث المبدأ لا يوجد فرق بين الزيادة (النقصان) من اليسار أو الزيادة (النقصان) من اليمين كقيمة وإنما الاختلاف في الاستخدام ، فعلى سبيل المثال التعبير I++ يعني إضافة القيمة واحد إلى قيمة المتحول I قبل استخدام المتحول I ، في حين أن التعبير I++ يعني استخدام القيمة الحالية للمتحول I ثم إضافة واحد إلى قيمة المتحول I .

❑ الزيادة بمقدار واحد (++)

مثال

```
int A ;
A++ ; // A = A + 1 ;
  Console.WriteLine(A );
```

❑ النقصان بمقدار واحد (--)

مثال

```
int B ;
B-- ; // B= B – 1 ;
Console.WriteLine(B );
```

مثال(٣-١٥)

اكتب برنامجاً بلغة C# يقوم بتعريف ثلاثة متحولات من نوع int كما يلي :

int k = 4 , i ,j ;

- أجرِ عملية الزيادة من اليسار للمتحول k ، ثم أجرِ الزيادة من اليمين للمتحول k ، ثم اطبع قيمة k .
- اجعل i = k++ ثم اطبع قيمة i و k .
- اجعل i = ++ k ثم اطبع قيمة i و k .
- ماذا تلاحظ ؟

البرنامج :

```
using System ;

class increment {
public static void Main(string [ ]args ) {
int k=4;
++k ;
k++ ;
Console.WriteLine("k="+ k );
int i = k++;
Console.WriteLine("i="+ i + "  k = " + k );

int j = ++k;
Console.WriteLine("j="+ j + "  k = " + k );
  } // end method main
  } // end class Name
```

```
6
6 7
8 8
```

يوضح البرنامج في المثال(٣-١٦) عمليات الزيادة increment (++) والنقصان decrement (- -) ، بينما نمط المعطيات المطبق هو char أي محرف واحد وليس سلسلة محرفية .

مثال(٣-١٦)

```
// char variables behave like integers.
using System ;
class CharDemo2 {
public static void Main(String [ ]args) {
  char ch1 = 'X', ch2 = 'C';
  Console.WriteLine("ch1 contains" + ch1+"\nch2
                                contains" +ch2);
  ch1++; // increment ch1
  ch2 - - ; // decrement ch2
   Console.WriteLine("ch1 is now " + ch1+"\nch2
                                  is now "  +ch2);
 } // end method main
   } // end class Name
```

```
RESULT
ch1 contains     X
ch2 contains     C
ch1 is now       Y
ch2 is now       B
Press any key to continue . . .
```

٣-٩- المتحولات والثوابت variables and constants

من المعروف أن البرمجة ليست في طباعة بعض الأسطر على الشاشة فقط، بل هي أبعد من ذلك بكثير، لذلك من أجل كتابة برامج تقوم بتنفيذ بعض المهام وحل مسائل متنوعة، فإننا نحتاج إلى التعرف إلى مفهوم المتحولات والثوابت ومعرفة أنواع المعطيات .

يستخدم كل برنامج أنواع مختلفة من المتحولات Variables ، ويستخدم كل برنامج المتحولات بطريقة ما . ففي برنامج Windows على سبيل المثال تكون جميع المعطيات التي تمثل كل معالم النافذة ، كاللون والحجم والموضع والنص في شريط العنوان ، كل هذه المعطيات مخزنة في متحولات . **فالمتحولات هي محددات مواقع في الذاكرة تخزن المعطيات التي يستخدمها البرنامج** . وهناك أنواع مختلفة من المتحولات المعرفة في لغة #C إضافة إلى إمكانية تعريف أنواع متحولات خاصة بالمبرمج .

٣-٩-١- التصريح عن المتحولات

قبل أن نستخدم المتحولات يجب التصريح عنها ، فبهذه الطريقة يعلم المترجم ما العمليات التي يمكن إجراءها على هذه المتحولات وبالتالي يستدعي العوامل المناسبة لإنجاز هذه العمليات ، ويتم التصريح عن المتحولات من خلال تحديد نوع المعطيات للمتحول بالإضافة إلى معرّف identifier وذلك حسب الصيغة التالية :

```
Variable _type    identifier ;
```

حيث :

Variable _type : هو نوع معطيات المتحول مثل int أو float أو غيرها .

Identifier : هو اسم المتحول الذي نريد استخدامه في البرنامج .

ينتهي كل تصريح عن متحول بفاصلة منقوطة ، وعندما يصادف المترجم تصريحاً فإنه يطالب بمساحة له في الذاكرة بما يكفي لتخزينه .

- أسماء المتحولات : غالباً ما يبدؤون كتابة أسماء المتحولات بحرف صغير وكتابة حرف كبير عند بداية كل كلمة بعدها ، مثلاً : accountBalance .
- لا يمكن في لغة #C أن نستخدم متحولاً قبل أن نسند له قيمة ابتدائية ، حيث تمنع هذه القاعدة الكثير من الأخطاء البرمجية الشائعة في اللغات الأخرى .
- يمكن أن نعطي المتحول قيمة ابتدائية عند التصريح عنه أو نقوم بذلك لاحقاً ، ولكن يجب أن نعطيه قيمة ابتدائية قبل الوصول إليه .

٣-٩-٢-قواعد كتابة المتحولات Variables

كل متحول يحتاج إلى اسم يميزه عن متحول آخر وعند تسمية أي متحول يجب أن نراعي بعض القواعد تجنباً للوقوع في أخطاء برمجية عند تنفيذ البرامج بلغة #C ، ويجب اتباع القواعد التالية عند تسمية المتحولات :

١. يجب أن يبدأ اسم المتحول بحرف من حروف اللغة الانكليزية الصغيرة أو الكبيرة ، ثم تتوالى بعد ذلك الحروف أو الأرقام ويجب ألا يحتوي على أي رمز خاص سوى الشرطة التحتية (_).

٢. من الممكن أن يكون طول اسم المتحول حتى 32 محرفاً وإذا زاد عن ذلك لا يأخذ بعين الاعتبار من قبل مترجم اللغة (Compiler).

٣. لغة #C تفرق بين الأحرف الكبيرة والصغيرة لذلك يجب عدم استخدام الأحرف الكبيرة والصغيرة لاسم المتحول نفسه، لأن المترجم يفرق بينها، مثلاً، المتحول a يختلف عن المتحول A .

٤. يجب أن يكون اسم المتحول فريداً في الكتلة التي عُرف بها .

٥. يجب الا يكون كلمة محجوزة مثل for ، أو أي قيمة منطقية true, false ، أو الكلمة null أو أي كلمة من الكلمات المحجوزة في لغة #C .

٦. يفضل أن يبدأ اسم المتحول بحرف صغير ، وفي حال كان الاسم مؤلفاً من أكثر من كلمة يفضل جعل الحرف الأول من كل كلمة (ما عدا أول كلمة) حرفاً كبيراً مثل : isGoodColor .

٣-٩-٣-مجال المتحولات

إن مجال المتحول هو المنطقة التي يمكن فيها استخدام المتحول ، ويحدد مجال المتحول متى يقوم النظام بإنشاء المتحول ومتى يقوم بتحريره من الذاكرة . وإن مكان توضيع المتحول البرمجي داخل النص البرمجي هو الذي يحدد مجاله ، وذلك حسب إحدى التصنيفات التالية :

١. **المتحولات الأعضاء Members Variable** : هي معطيات أعضاء من صف ، ويتم التصريح عنها داخل الصف ولكن خارج أي تابع أو بان ، ومجال هذا النوع من المتحولات هو كامل الصف .

٢. **المتحولات المحلية Local Variables** : يتم التصريح عنها داخل الكتلة، ومجالها يمتد من نقطة التصريح عنها ولغاية نهاية الكتلة التي صُرحت فيها.

٣. الوسطاء لتابع (Parameters function) : هي متحولات يتم استخدامها في التوابع أو البواني من أجل تمرير القيم من/إلى التابع أو الباني و أن مجال المتحولات هو كامل التابع .

يمكن تهيئة المتحولات الأعضاء والمتحولات المحلية مباشرة أثناء التصريح عنها :

```
int  x=200   ;   long y=300 ;
```

وإن الوسطاء لا يمكن إسناد قيم لها مباشرة ، بل يجب تمرير قيمها من خلال النص البرمجي .ويوضح المثال(٣-١٧) يعرض مجال المتحولات المحلية ضمن الكتل .

مثال(٣-١٧)

```
1. // Demonstrate block scope.
2. using System ;
3. class Scope {
4. static void Main(string[ ]args ) {
5. int  x; // known to all code within main
6. x = 10;
7. if( x == 10) { // start new scope
8. int  y = 20; // known only to this block
9. // x and y both known here.
10.   Console.WriteLine("x and y :{0} ,{1} " , x , y);
11.   x = y * 2;
12.   }//End if
13.   // y = 100; // Error! y not known here
14.   // x is still known here.
```

```
15.   Console.WriteLine("x is " + x);
16.   } // end method main
17.   } // end class Name
```

```
RESULT
 x and y: 10  20
 x  is  40
Press any key to continue . . .
```

- المتحول x في السطر 6 يتم التصريح عنه ضمن مجال التابع Main وبالتالي يكون معرفاً حتى نهاية الـ Main وسهل الوصول إليه .
- بينما المتحول y في السطر 9 فهو معرف ضمن كتلة الـ if فقط بينما المتحول x يكون معرفاً في هذه الكتلة مع المتحول y .
- بينما في السطر 15 خارج الكتلة if يكون المتحول x معرفاً أما y فيكون غير معرفاً .

٣-٩-٤-أنواع المتحولات

كل متحول يُستخدم في برنامج بلغة #C لا بد من التصريح عنه ويتم ذلك في أي مكان من البرنامج ولكن بشرط أن يكون التصريح قبل العبارات البرمجية التي ستستخدم هذا المتحول .ويُعد نوع المتحول مهماً لعدة أسباب :

✓ الأول هو أن المتحول يحتاج أن يطلب من النظام أنشاء مساحة في الذاكرة لتخزين المتحول فيما لا يعرف ما هي المساحة التي سيطلبها بدون معرفة نوع المتحول .

✓ كما إن المترجم يحتاج **أن يعرف العمليات المقبول إنجازها** على متحول محدد بحيث يمكنه فرض القواعد الملائمة عند استخدام هذا المتحول ، فبدون معرفة النوع ليست لدى المترجم طريقة لمعرفة أي القواعد سيفرض.

هناك عدة أنواع أولية للمتحولات ضمن لغة #C :

- هناك أنواع رقمية لتمثيل الأعداد الصحيحة والفاصلة العشرية .
- هناك النوع char لتمثيل محرف واحد .
- هناك النوع bool الممثل لقيمة بوليانية وهي إما صحيح true أو خطأ false .
- أما النوع string فيمثل سلسلة من المحارف .

في لغة #C يوجد العديد من المتحولات التي يمكن استخدامها والتصريح عنها ويبين الجدول (٣-١٤) بعض أنواع المتحولات الشائعة في #C .

النوع	يُستعمل لتخزين	أمثلة عن القيم المخزنة
Char	أحرف(محارف) 2byte	char ch= ' a'
Short	أعداد صحيحة قصيرة مع إشارة 2byte	short x= 222
Int	أعداد صحيحة عادية مع إشارة 4byte	int z = 14543
Long	أعداد صحيحة طويلة مع إشارة 8byte	Long y= 76543987
Bool	تمثيل قيمة بولانية true أو false 1byte	9176 321 236.01202,6

ushort a=222	أعداد صحيحة قصيرة بدون إشارة 2byte	ushort
uint b=567432	أعداد صحيحة عادية بدون إشارة 4byte	uint
ulong c=654438765	أعداد صحيحة طويلة بدون إشارة 8byte	ulong
float n= 55.8	أعداد حقيقية قصيرة مع إشارة 4byte	float
double k = 5437.8765	أعداد حقيقية مزدوجة مع إشارة 4byte	double

الجدول (٣-١٤) – بعض أنواع المتحولات الشائعة في C#

تحتوي لغة C# على نوعين من أنماط المعطيات :

- الأنماط الأساسية Primitive Type
- الأنماط المرجعية References Type

وتستطيع الأنماط الأساسية تخزين قيمة واحدة فقط (رقم ، محرف ، قيمة منطقية ، ...) وتحتاج إلى حجم معين في الذاكرة ، أما **أنواع الأنماط الأساسية** هي :

```
bool , byte, char, short,
int, long, float, double
```

أما **الأنماط المرجعية** هي : الصفوف Classes والمصفوفات Arrays .

وفيما يلي سنذكر أهم هذه المتحولات وكيفية التصريح عنها .

١-النمط boolean

النمط boolean : هو 1Bit من المعطيات , ويأخذ قيمتين true أو false أي (1 , 0) .

Declaration: boolean b;
Assignment: b = true; b = false;

أما المعامل (cast) يستخدم للتحويل من نوع إلى آخر من المعطيات ,و يبين المثال(٣-١٨) استخدام هذا النمط .

مثال(٣-١٨)

```
// Demonstrate boolean values.
using System ;
class BoolTest  {
static void  Main( string[]args )  {
bool   b;
      b = false;
      Console.WriteLine("b is " + b);
      b = true;
       Console.WriteLine("b is " + b);
// a Boolean value can control the if statement
      if(b) Console.WriteLine("This is executed." );
      b = false;
     if(b) Console.WriteLine("This is executed." );
// outcome of a relational operator is a Boolean value
     Console.WriteLine("10 > 9 is " + (10 > 9) );
} // end method main
  } // end class Name
```

```
RESULT
 b is False
 b is True
 This is executed.
 10 > 9 is True
Press any key to continue . . .
```

٢- التصريح عن المتحولات byte , char & string

- النمط **byte** : يُستخدم للأعداد الموجبة فقط 1byte=8bit من (0 -255).
- النمط **sbyte** : يُستخدم للأعداد الموجبة الصحيحة والسالبة بحجم 1byte.
- النمط **char** : هو محرف وحيد بحجم 2bytes .

للتصريح عن متحول محرفي تُستخدم الكلمة المحجوزة char ، ويتم تخزين المحارف في متحولات من النوع char، والعبارة التالية توضح ذلك :

char ch ;

تنشئ مساحة في الذاكرة لمحرف وتسميه ch وحجمه 1Byte ، ولتخزين محرف ما في هذا المتحول نكتب :

ch = 'z' ;

ودائماً تكون المحارف الثابتة كـ 'a' و 'b' محصورة بعلامتي اقتباس مفردة.

<u>مثال:</u>

```
char  a ;   //a is  a character variable
char  b ;   //b is a character variable
    or
 char  a , b ;
```

و يمكن استعمال المتحولات من النوع char لتخزين أرقام كاملة بدلاً من محارف فمثلاً يمكننا كتابة:

ch = 2 ;

لكن نطاق القيم الرقمية التي يمكن تخزينها في النوع char يتراوح بين (127إلى 128-) لذا فإن هذه الطريقة تعمل مع الأرقام الصغيرة فقط.

تُعامل السلاسل المحرفية في لغة C# كأغراض، فهي أمثال من الصف String ، لكن طبيعتها البسيطة تضطرنا في بعض الأحيان للحديث عنها كمتحولات من أنماط أولية . فعندما نريد إدخال مجموعة محارف إلى برنامج ما بلغة #C، يجب أن نصرح عن المتحول على شكل سلسلة من المحارف String وذلك كما يلي:

String season ;

أو

String season = " Winter " ;

- String literals are part of the language.
- The string concatenation operator + is part of the language.

```
Console.WriteLine("Hello " + "World");
```

يبين المثال(٣-١٩) المتحولات من النوع char .

مثال(٣-١٩)

```
using System ;
static void  Main( string[]args )  {
     char ch1 = 'X';
```

```
class BoolTest {
      Console.WriteLine("ch1 contains " + ch1);
       ch1++; // increment ch1
      Console.WriteLine("ch1 contains " + ch1);
} // end method main
  } // end class Name
```

```
RESULT
 ch1 contains X
 ch1 contains Y
Press any key to continue . . .
```

٤- التصريح عن المتحولات الصحيحة

تمثل المتحولات الصحيحة أرقاماً كاملة أي قيماً يمكن تعدادها ، كعدد الأشخاص أو الأيام أو عدد الصفحات مثلاً ، ولا يمكن أن تكون الأعداد الصحيحة أرقاماً ذات فاصلة عشرية ولكنها يمكن أن تكون سالبة, وهنالك ثلاثة أنواع من الأعداد الصحيحة في Java ، هي :

short : عدد صحيح قصير .

int : عدد صحيح عادي .

long : عدد صحيح طويل .

وهي تحتل مساحات مختلفة في الذاكرة ، والجدول (٣-١٥) يبين هذه الأنواع والمساحة التي تأخذها في الذاكرة ومجال الأرقام التي يمكن أن تأخذها.

اسم النوع	الحجم	المجال
char	1byte	128- إلى 127
Short	2byte	-32,768 إلى 32,767
int	4byte	-32768 إلى +32798
Long	8byte	-2,147,483,648 إلى 2,147,483,647

الجدول (٣-١٥) – يبين أنواع المتحولات الصحيحة

للتصريح عن متحول صحيح من النوع **int** .

<u>مثال:</u>

```
int  a ;  //a  is an integer variable
int  b ;  //b  is an integer variable
     or
int  a , b ;
```

– للتصريح عن متحول صحيح من النوع **long** .

<u>مثال:</u>

```
long  a ; // a  is a long integer variable
long  b ; // b is a long integer variable
     or
long  a , b ;
```

يحسب البرنامج في المثال(٣-٢٠) السرعة التقريبية للضوء ميل / ثانية باستخدام المتحولات من نمط Long وذلك لـ 1000 يوم .

مثال(٣-٢٠)

```
using System ;
class Light  {
static  void  Main( string[]args )  {
   long lightspeed, days, seconds, distance;
// approximate speed of light in miles per second
 lightspeed = 186000; // ( m/s )
days = 1000;  // specify number of days here
seconds = days * 24 * 60 * 60; // (second )

// compute --distance
distance = lightspeed * seconds; //(miles )
Console.Write ("In  " + days);
Console.Write ("days light will travel about " );
Console.WriteLine(distance + " miles.");
} // end method main
  } // end class Name
```

RESULT

```
In 1000 days light will travel about
  16070400000000  miles.
Press any key to continue . . .
```

٥-التصريح عن متحول حقيقي

يتم استعمال المتحولات الحقيقية لتمثيل قيم يمكن قياسها كالأطوال أو الأوزان، ويتم تمثيل الأعداد الحقيقية عادة برقم كامل على اليسار مع فاصلة عشرية وكسر على اليمين، وهنالك ثلاثة أنواع من الأعداد الحقيقية في أنظمة التشغيل الشائعة الاستعمال ، وأشهر أنواع الأعداد الحقيقية هي من النوع double والذي يتم استعماله لمعظم توابع #C الرياضية ، ويتطلب النوع float ذاكرة أقل من النوع double ، و يوضح الجدول(٣-١٦) هذه الأنواع والحجم الذي تأخذه في الذاكرة.

النوع	الوصف	الحجم
Float	عدد حقيقي	4 byte
Double	عدد حقيقي مضاعف	8 byte
decimal	عدد عشري	12 byte

الجدول(٣-١٦) – يبين أنواع المتحولات الحقيقية

للتصريح عن متحول حقيقي من النوع **float** :

<u>مثال:</u>

```
float a ; // a is a real variable
float b ; // b is a real variable
    Or
float a , b ;
```

– للتصريح عن متحول حقيقي من النوع **double** :

<u>مثال:</u>

```
double a ; // a is a big real variable
double b ; // b is a big real variable
```

-للتصريح عن متحول حقيقي من النوع **decimal** :

مثال:

```
decimal a ; // a is a very big real variable
decimal b ; // b is a very big real variable
```

يحسب البرنامج في المثال(٣-٢١) مساحة الدائرة بالاعتماد على المتحولات الحقيقية .

مثال(٣-٢١)

```
using System ;
class Light  {
static void  Main( string [ ]args )  {
     double  Pi,  r,  a ;
     r = 10.8;// radius of circle
     Pi = 3.1416;// pi, approximately
     a = Pi * r * r;// compute area
     Console.WriteLine("Area of circle is {0} ", a);
} // end method main
  } // end class Name
```

RESULT

```
Area of circle is 366.436224
Press any key to continue . . .
```

٦- المتحولات من النوع object

المتحولات من النوع object يمكن أن تكون أي نوع من أنواع المتحولات السابقة أي :

int , float , double , char , decimal , string .....

ويتم التصريح عنها كما يلي :

```
object  x=313.22222m;
object  x=313.22222f;
object  x = 'a'
object  x = "Hello"
```

٣-٩-٥-التحويل بين المتحولات Type Casting

في بعض الأحيان من الضروري دمج مجموعة من أنماط مختلفة في عملية الحساب ، فعند تنفيذ عملية ثنائية تتضمن حدين من نمطين مختلفين فإن لغة C# تقوم بشكل تلقائي بتحويل الحد معتمدة على القواعد التالية :

١. إذا كان أحد الحدود من النمط double عندها يتم تحويل الحد الآخر إلى النمط double (الحد الآخر مثلاً من النمط int أو float).
٢. إذا كان أحد الحدود من النمط float عندها يتم تحويل الحد الأخر إلى النمط float (الحد الآخر مثلاً من النمط int أو byte).
٣. إذا كان أحد الحدود من النمط long عندها يتم تحويل الحد الآخر إلى النمط long (الحد الآخر مثلاً من النمط int أو byte).

لا يمكن إسناد قيمة إلى متحول نمطه ينتمي إلى مجال أصغر من نمط القيمة، إلا إذا قمنا بعملية تحويل للنمط (Type Casting) . حيث إن تحويل النمط هو

عبارة عن عملية تقوم بتحويل قيمة تنتمي لنمط معطيات ما إلى قيمة تنتمي لنمط معطيات آخر .

وتعرف عملية تحويل متحول ينتمي لنمط ذي مجال صغير إلى متحول ينتمي لنمط ذي مجال كبير بعملية تمديد النمط (Widening a Type)، وتعرف عملية تحويل متحول ينتمي لنمط ذي مجال كبير إلى متحول ينتمي لنمط ذي مجال صغير بعملية تضييق النمط(Narrowing a Type) . ومن الممكن إنجاز عملية تمديد النمط بشكل تلقائي وبدون القيام بعملية تحويل صريحة ، بينما يجب إنجاز عملية تضييق النمط بشكل صريح .

الصيغة العامة لعملية التحويل هي وضع النمط في أقواس ( ) ثم اتباعه باسم المتحول أو القيمة التي سيتم تحويلها , مثلاً :

```
int y = 55 ;
float x = ( float ) y ;
```

تم تحويل y من النمط int إلى النمط float .

يبين البرنامج في المثال(٣-٢٢) التحويل بين المتحولات المختلفة ، يقوم البرنامج بتحويل المتحول i من نوع int إلى byte وإسناده إلى المتحول b الذي هو من نوع byte وطباعته، والمتحول d من double إلى int إلى المتحول i وطباعته ، و المتحول d من double إلى byte إلى المتحول b وطباعته.

مثال(٣-٢٢)

```
sing System;
// Demonstrate casts.
class Conversion {
static void Main(string[] args) {
```

```
 byte  b;
 int  i = 257;
 double  d = 323.142;
Console.WriteLine("\nConversion of int to byte.");
 b = (byte) i;
Console.WriteLine("i and b : " + i + "  to " + b);
Console.WriteLine("\nConversion of double to int.");
 i = (int) d;
Console.WriteLine("d and i : " + d + "  to " + i);
Console.WriteLine("\nConversion of double to byte.");
 b = (byte) d;
Console.WriteLine("d and b  : " + d + " to " + b);
}// end main
}// end class
```

RESULT

```
Conversion of int to byte.
i and b 257  1
Conversion of double to int.
d and i 323.142  323
Conversion of double to byte.
d and b 323.142  67
Press any key to continue . . .
```

التمثيل الثنائي للمتحول i هو كالتالي : $2^8 = 256$ (10000000) حيث

يمتلك النمط الصحيح 4byte (00000000 00000000 00000001

00000000) والبايت الأول في i هو 00000000 ، والذي تم إسناده إلى المتحول b ، وهكذا أصبحت قيمة b هي 0 . في البرنامج كانت قيمة i = 257 عندها قيمة b= 1 . حيث تم تحويل المتحول d = 323.142 إلى متحول من نمط صحيح عندها i = 323 حيث تم حذف الجزء العشري . بينما تم تحويل المتحول d = 323.142 إلى متحول من نمط byte عندها b =67 .
ويجب الانتباه عند التحويل من نمط إلى أخر لكي لا نفقد جزءاً من المعطيات .

٣-٩-٦-المتحولات الثابتة Constant Variables

يمكن التصريح عن متحول بحيث تُصبح قيمته ثابتة (أي غير قابلة للتعديل) باستخدام الكلمة المحجوزة const وفي لغة C# نميز عدة أنواع من المتحولات الثابتة ، منها :

```
const float  PI     = 3.14159    ;
const int    w = 100 ;
const char   newline    = ' a'   ;
```

يحسب البرنامج في المثال(٣-٢٣) مساحة دائرة نصف قطرها r وطباعتها باستخدام المتحول من نوع const.

مثال(٣-٢٣)

```
using System ;
class Light  {
static void  Main( string [ ]args )  {
   double  const Pi = 3.1416;// pi, approximately
```

```
int const  r = 10 ;// radius of circle
 a = Pi * r * r;// compute area
  Console.WriteLine("Area of circle is {0} ", a);
} // end method main
 } // end class Name
```

```
RESULT
 Area of circle is  314.16
 Press any key to continue . . .
```

١٠-١-العمليات المنطقية على البت Bitwise

تعمل معاملات الحساب بالنظام الثنائي على مستوى البتّات بغض النظر عما تمثله المعطيات ، ويعد وجود هذه المعاملات في C# مثيراً للتساؤل ، ليس لأنها غير مفيدة ، بل أنها ضرورية جداً عند الحاجة لتحسين مستوى الأداء ، إنما الغريب وجود نمط عددي بدون إشارة في لغة C#. حيث تمثل المعطيات في الذاكرة بسلاسل من البتات تجمع عادة في وحدات من مضاعفات العدد 8 ، والعنصر الأساسي هو البايت Byte . ونجد في C# نمط معطيات تمثل إحدى بتات العدد إشارته (سالب أو موجب) . ويبين الجدول (٣-١٧) أهم المعاملات المنطقية في لغة C# .

AND (&)	OUT
00	0
01	0
10	0
11	1

OR (|)	OUT
00	0
01	1
10	1
11	1

XOR (^)	OUT
00	0
01	1
10	1
11	0

NOT (~)	OUT
0	1
1	0

الجدول (٣-١٧) – يبين أهم المعاملات المنطقية في لغة C#

يمكن تطبيق العمليات الحسابية الثنائية على الأنماط (char , long , int , short , byte) ويمكن تطبيق المعاملات (& ، | ، ^) على قيم من النمط boolean . ويبين الجدول (٣-١٨) العمليات المنطقية والازاحة .

&	اجراء عملية AND على حدين
|	اجراء عملية OR على حدين
^	اجراء عملية XOR على حدين
~	اجراء عملية NOT على حد واحد
<<	إزاحة نحو اليسار وتأخذ حدين
>>	إزاحة نحو اليمين وتأخذ حدين
>>>	إزاحة نحو اليمين بدون نشر إشارة

الجدول (٣-١٨) – يبين العمليات المنطقية والازاحة

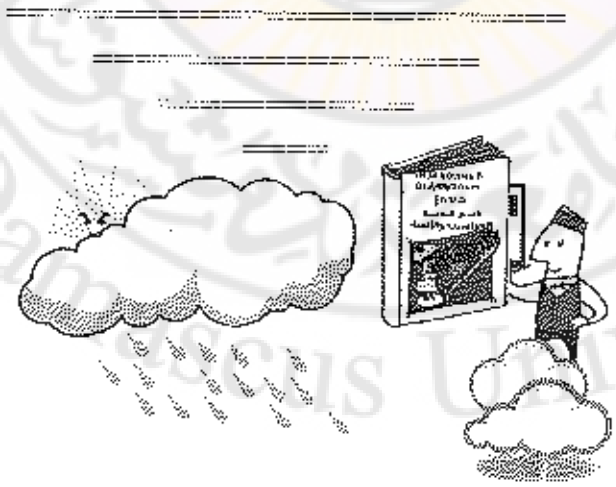
يوضح المثال (٣-٢٤) العمليات المنطقية : OR,EXOR,AND,NOT على مستوي البت Bit ، وهي مرتبة حسب الأولوية للعمليات المنطقية

مثال(٣-٢٤)

```
using System;
   class Program  {
        static void Main(string[] args) {
 // Demonstrate the bitwise logical operators.
String [ ]binary = { "0000", "0001", "0010", "0011",
                     "0100", "0101", "0110", "0111",
                     "1000", "1001", "1010", "1011",
                     "1100", "1101", "1110", "1111" };
//-------------------------------------------
int  a = 3; // 0 + 2 + 1 or 0011 in binary
int  b = 6; // 4 + 2 + 0 or 0110 in binary
int  c = a | b; //The Bitwise OR
int  d = a & b; //The Bitwise AND
int  e = a ^ b; //The Bitwise EXOR
int  f = (~a & b)|(a & ~b); //The Bitwise NOT and AND
int  g = ~a & 0x0f;
Console.WriteLine(" a = " + binary[a]);
Console.WriteLine(" b = " + binary[b]);
Console.WriteLine(" a|b = " + binary[c]);
```

```
Console.WriteLine(" a&b = " + binary[d]);
Console.WriteLine(" a^b = " + binary[e]);
Console.WriteLine("~a&b|a&~b = " + binary[f]);
Console.WriteLine(" ~a = " + binary[g]);
        }// end main
    } // end class
```

```
RESULT
  a = 0011
  b = 0110
  a | b = 0111
  a & b = 0010
  a ^ b = 0101
  ~ a & b | a &
  ~ b = 0101

Press any key to continue . . .
```



مسائل عامة

١- حدد ما إذا كانت العبارات الآتية صحيحة أم خاطئة:

- التعليقات تجبر الحاسوب على طباعة النص الذي يلي // على الشاشة عند تنفيذ البرنامج.
- تتابع الهروب \n يجبر المؤشر على الانتقال إلى سطر جديد.
- برنامج C# والذي يقوم بطباعة ثلاثة أسطر على الشاشة يجب أن يحتوي على ثلاث عبارات تستعمل `Console.WriteLine( ) ;`

٢- ما خرج العبارة الآتية:

```
Console.WriteLine("\n **\n ***\n ****\n") ;
```

٣- اكتب عبارة C# صحيحة تقوم بالآتي:

- تعريف المتغيرات x، y، z و result لتكون من النوع int.
- الطلب من المستخدم إدخال ثلاثة أرقام صحيحة.

٤- حدد ما إذا كانت العبارات الآتية صحيحة أم خاطئة :

- يجب الإعلان عن المتغيرات قبل استعمالها في البرنامج.
- يجب تحديد نوع المتغيرات عند التصريح عنها.
- لا تفرق لغة C# بين المتغيرات Number و number .

٥- ما هو الخرج الناتج عند تنفيذ العبارات البرمجية التالية المكتوبة بلغة C#:

1) 2 * 5 * 5 + 3 * 5 + 7
2) (3 * 9 * (3 + (9 * 3 / (3))))
3) 2 % 2 + 2 * 2 - 2 / 2

٦-اكتب برنامجاً بلغة الـ C# يقوم بطباعة ما يلي:

```
Welcome with Java.
This program
Is the third in Java.
```

٧- اكتب برنامجاً بلغة C# يقوم بحساب وطباعة مساحة مستطيل بعداه L و W ، بحيث يتم إدخال أبعاده من قبل المستخدم .

٨ - اكتب برنامجاً بلغة الـ C# يقوم بإدخال أي عدد صحيح من قبل المستخدم وطباعة مربع العدد ومكعبه.

٩-اكتب برنامجاً لتنفيذ العبارة الرياضية التالية ، بصيغ مختلفة باستخدام الأقواس.

$y = 6+12/6*2-1;$

$y = (6+12)/(6*2-1);$

$y = (6+12/6)*2-1;$

الفصل الرابع

بنى التحكم وحلقات التكرار

Control Structures &Loops

بنى التحكم وحلقات التكرار

٤-١- مقدمة

نحتاج في التطبيقات العملية لاتخاذ بعض القرارات تبعا لشروط معينة، ومن هنا ظهرت الحاجة لوجود طرق لجعل البرنامج قادراً على تغيير تسلسل تنفيذ التعليمات تبعا للشروط المطلوبة وهو ما يعرف بالجمل الشرطية ، وكثيرًا ما نحتاج في البرامج إلى تكرار أمر موجه للحاسوب عدداً من المرات، وتوفر لغة C# عدة وسائل تمكن المبرمج من أداء هذا التكرار ، وعادة ما تسمى هذه الوسائل " الحلقات التكرارية " ، ويوجد عدد من الحلقات التكرارية في لغة C# ، وتعد أساليب الشرط والتكرار بمثابة القلب في جسم لغات البرمجة، وبدونها لا يمكن تنظيم أي برنامج. وتوفر لغة C# للمبرمج عدداً من الأساليب والتوابع الفعالة، المتعلقة بهذا الشأن، وتمتاز هذه الأساليب بأنها أساليب بنيوية Structured أي يمكن تنظيم عمليات التحكم والتكرار فيها، بأسلوب ذاتي من بداية العمليات وحتى نهايتها دون تدخل من المبرمج أثناء هذه العمليات، للأشراف على التوجيه والتخطيط لكل خطوة من خطوات البرنامج ، وحيث إن جواب الشرط أما أن يكون صحيحاً true أو خاطئاً false ، فإن لغة C# توفر مرونة كبيرة في استخدام عدد كبير من التوابع ، وفي توجيه البرنامج بطريقة فعالة وفائقة.

٤-٢- بنى التحكم

Control Structures

إن بنى التحكم عبارة عن بنى تتحكم في تنفيذ خطوات البرنامج من خلال شرط أو مجموعة شروط ، وإن عدم استخدام بنى التحكم يؤدي إلى جعل مترجم لغة C# يقوم بتنفيذ تعليمات البرنامج بالترتيب نفسه الذي ظهر في ملف المصدر ، ويمكن استخدام بنى التحكم لتغيير مسار تنفيذ البرنامج إما بوضع تعليمات شرطية

(تنفذ عند تحقيق شرط معين فقط) أو حلقات تكرارية (تكرار مجموعة من التعليمات عدداً من المرات) ، وتزودنا لغة #C بعدد من تعليمات التحكم .

٤-٢-١-بنية التعليمة if

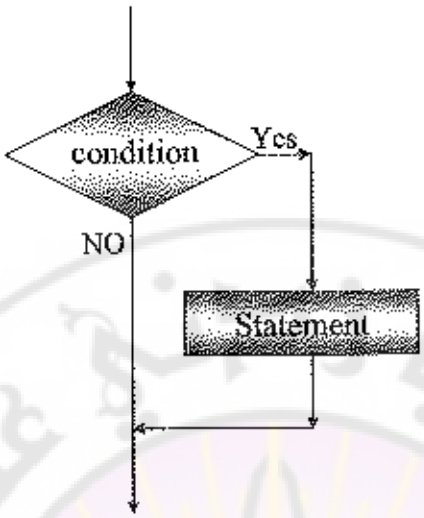
تستخدم التعليمة if في اختبار شرط ما، فإذا تحقق الشرط يتم تنفيذ تعليمة أو مجموعة من التعليمات والشكل التالي يوضح الصيغة العامة لهذه البنية في حالة تنفيذ تعليمة واحدة.

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

حيث يكون condition هو الشرط الواجب تحققه حتى يتم تنفيذ التعليمة statement ، ويمكن أن يكون هذا الشرط على شكل مقارنة بين قيمتين أو متحولين ، وهذه المقارنة يمكن توضيحها من خلال الجدول(٤-١) ، ويبين الشكل(٤-١) المخطط التدفقي لبنية التعليمة if .

الشرط	معنى الشرط
if(a==b)	هل قيمة المتغير a تساوي قيمة المتغير b ؟
if (a<b)	هل قيمة المتغير a أصغر من قيمة المتغير b ؟
if(a<=b)	هل قيمة المتغير a أصغر أو تساوي قيمة المتغير b ؟
if(a>b)	هل قيمة المتغير a أكبر من قيمة المتغير b ؟
if(a>=b)	هل قيمة المتغير a أكبر أو تساوي قيمة المتغير b ؟
if(a!=b)	هل قيمة المتغير a لا تساوي قيمة المتغير b ؟

الجدول(٤-١) – يبين جدول المقارنة للتعليمة الشرطية



الشكل(٤-١) – يبين المخطط التدفقي للتعليمة if

البرنامج الجزئي التالي يوضح كيفية تنفيذ تعليمة واحدة في حال تحقق الشرط .

Example:

```
1. if (grade >= 90)
2. Console.WriteLine(" Congratulations!\n");
3. Console.WriteLine(" Your grade is " + grade);
```

- إذا كانت grade >= 90 فتُطبع العبارة التالية Congratulations! ثم يتم طباعة العبارة Your grade is 77 على سبيل المثال .
- أما إذا كانت أصغر من 60 فإنه يتم تجاوز السطر 2 ويتم تنفيذ السطر 3 فقط .

وفي حالة تنفيذ مجموعة من التعليمات تكون الصيغة العامة لهذه البنية كما يلي :

```
if (condition)
{
    statement 1;
    statement 2;
    ..............
    statement n;
}
```

حيث : condition هو الشرط الواجب تحققه حتى يتم تنفيذ التعليمات statement 1, statement 2, ..., statement n .

البرنامج الجزئي التالي يوضح كيفية تنفيذ مجموعة من التعليمات في حال تحقق الشرط .

```
if( x= = 50 )
{
 Console.Write ("x is ");
 Console.WriteLine ("50 ") ;
}
```

يمكن أن يكون هناك أكثر من تعليمة if ويتم تنفيذها حسب ورودها ، بحيث يتم تنفيذ التعليمة الأولى أولاً ثم الثانية ... وهكذا ، والشكل التالي يوضح هذه البنية .

```
if ( condition )
  statement;

if ( condition ) {
  statement1;
  statement2;
  ...
}
```

```
example code
if ( a < 12 )
   b = 45;

if ( x.length()<10) {
   x = x + "SLAM";
   y=x.length()-3;
}
```

ويمكن أن تكون متداخلة ، بحيث يتم تنفيذ وإنهاء الـ if الداخلية وبعد ذلك يتم تنفيذ وإنهاء الـ if الخارجية ، كما هو مبين في الشكل التالي :

```
if (expression)
{
   statement1;
   Statement2;
   if (expression)
   {
      statement1;
      statement2;
   }// inner if
} // outer if
```

```
Example:
if (u > v)
{
      a = 1;
      b = 2;
    if ( u > z)
      {
        x =11;
        y = 12;
      }// الداخلية
}// الخارجية
```

البرنامج الجزئي التالي يوضح استخدام أكثر من تعلمة if ، يتحقق الشرط عندما تكون قيمة x و y أكبر من 5 .

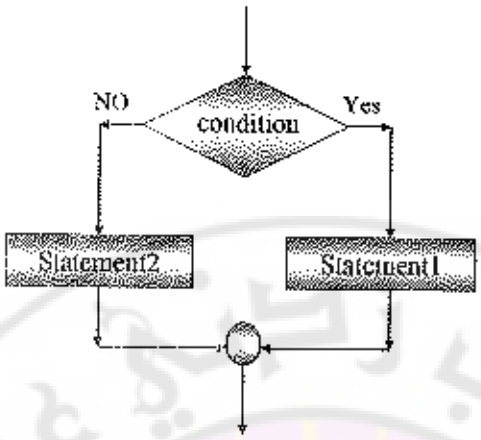
```
if ( x > 5 )
if ( y > 5 )
Console.WriteLine( "x and y are > 5" );
```

٤-٢-٢ – بنية التعليمة if / else

تستخدم التعليمة if / else في اختبار شرط ما ، فإذا تحقق الشرط يتم تنفيذ تعليمة أو مجموعة من التعليمات ، وإذا لم يتحقق الشرط فإنه يتم تنفيذ تعليمة أخرى أو مجموعة من التعليمات ، الصيغة العامة لهذه البنية كما يلي :

مثال (٤-١)

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



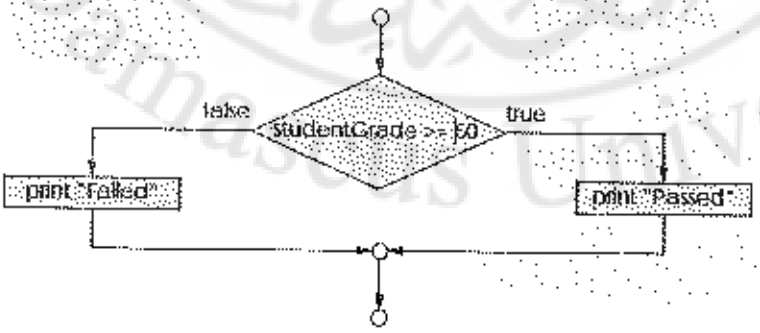
الشكل (٤-٢) - المخطط التدفقي للتعليمة if/else

فإذا تحقق الشرط condition ، يتم تنفيذ التعليمة statement 1 وإذا لم يتحقق الشرط فيتم تنفيذ التعليمة statement 2 .

البرنامج الجزئي التالي يوضح كيفية تنفيذ تعليمة واحدة في تحقق الشرط ، بحيث يطبع كلمة ناجح أو راسب فقط .

```
if ( studentGrade >= 60 )
    Console.WriteLine( "Passed" );
else
    Console.WriteLine( "Failed" );
```

إذا تحقق الشرط يطبع كلمة "Passed"، وإذا لم يتحقق الشرط يطبع كلمة "Failed" .



وعند تنفيذ أكثر من تعليمة في حال تحقق الشرط أو عدم تحقق الشرط ، فإن الصيغة العامة لهذه البنية كما يلي :

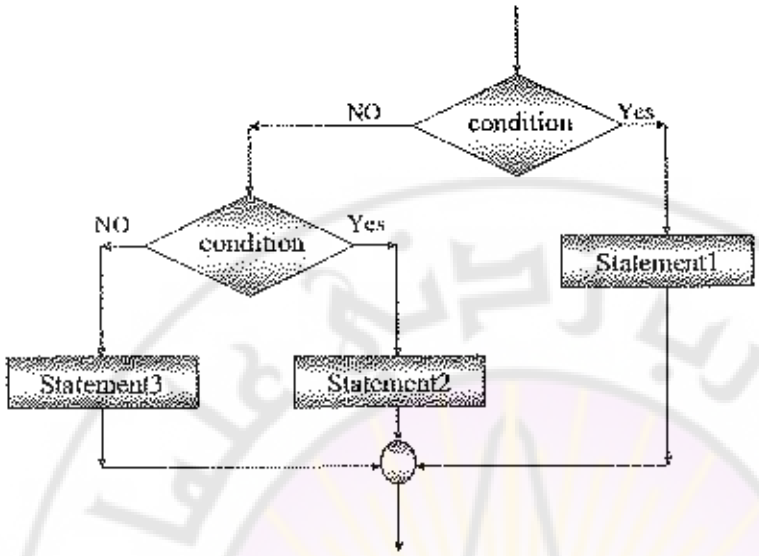
```
if(condition)
  {
    statement 1;
    statement 2;
    ...............
    statement n;
  }
else {
        statement 1;
        statement 2;
        ...............
        statement m;
      }
```

٤-٢-٣-بنية التعليمة if- else-if

تستخدم التعليمة if- else-if في اختبار الشرط الأول ، فإذا تحقق هذا الشرط فان التعليمة الأولى(أو مجموعة من التعليمات) تنفذ وإذا لم يتحقق الشرط الأول يتم اختبار الشرط الثاني ، فإذا تحقق الشرط الثاني يتم تنفيذ التعليمة الثانية (أو مجموعة من التعليمات) وإلا يتم تنفيذ التعليمة الثالثة (أو مجموعة من التعليمات)، والصيغة العامة لهذه البنية هي التالي:

```
if ( condition1 )
      statement1;
else if ( condition2 )
          statement2;
     else
          statement3;
```

ويبين الشكل (٤ – ٣) المخطط التدفقي للتعليمة if-else-if .



لشكل (٢– ٣) – يبين المخطط التدفقي لبنية التعليمة if-else-if

المثال (٤–١) يوضح التعليمة if / else والتعليمة if . حيث يتم إدخال رقم الشهر وطباعة الفصل الذي ينتمي إليه هذا الشهر .

مثال (٤–١)

```
// Demonstrate if-else-if statements.
using System ;
class IfElse {
public static void Main(string[] args ) {
int month = 4; // April
String  season;
if(month == 12 || month == 1 || month == 2)
season = "Winter";
```

```
else if(month == 3 || month == 4 || month == 5)
season = "Spring";
else if (month == 6 || month == 7 || month == 8)
season = "Summer";
else if (month == 9 || month == 10 || month == 11)
season = "Autumn";
else
season = "Bogus Month";
Console.WriteLine("April is in the "+ season + ".");
 } // end method main
} // end class Name
```

```
RESULT
 April is in the Spring.
Press any key to continue . . .
```

يقدم البرنامج في المثال(٤-٢) سلسلة من المقارنات لمتغير للتحقق فيما إذا كان أحد الأحرف الصوتية (العلة) من الأبجدية الإنجليزية. إذا لم يتم استيفاء أي شرط من الشروط ، فإنه يطبع كلمة Consonant أي ساكن .

مثال (٤-٢)

```
// Demonstrate if-else-if statements.
using System ;
class Ivowels {
char ch = 'X';
if (ch == 'A' || ch == 'a')
{
```

```
    Console.WriteLine("Vowel [ei]");
}

else if (ch == 'E' || ch == 'e')
{
  Console.WriteLine("Vowel [i:]");
}
else if (ch == 'I' || ch == 'i')
{
  Console.WriteLine("Vowel [ai]");
}
else if (ch == 'O' || ch == 'o')
{
  Console.WriteLine("Vowel [ou]");
}
else if (ch == 'U' || ch == 'u')
{
  Console.WriteLine("Vowel [ju:]");
}
else
{
  Console.WriteLine("Consonant");
}
}// end main
}// end class
```

عند ادخال الحرف O يطبع العبارة التالية :

```
RESULT
  Vowel [ou]
Press any key to continue . . .
```

عند ادخال الحرف M يطبع العبارة التالية :

```
RESULT

    Consonant
Press any key to continue . . .
```

مثال (٤-٣)

اكتب برنامجاً بلغة #C يقوم بإدخال درجة الطالب grade ويختبر هل هي A أو B أو C أو D أو F .(الادخال والإخراج عن طريق الواجهات) .

حيث : $A \geq 90$ ، $B \geq 80$ ، $C \geq 70$ ، $D \geq 60$ ، $F < 60$.

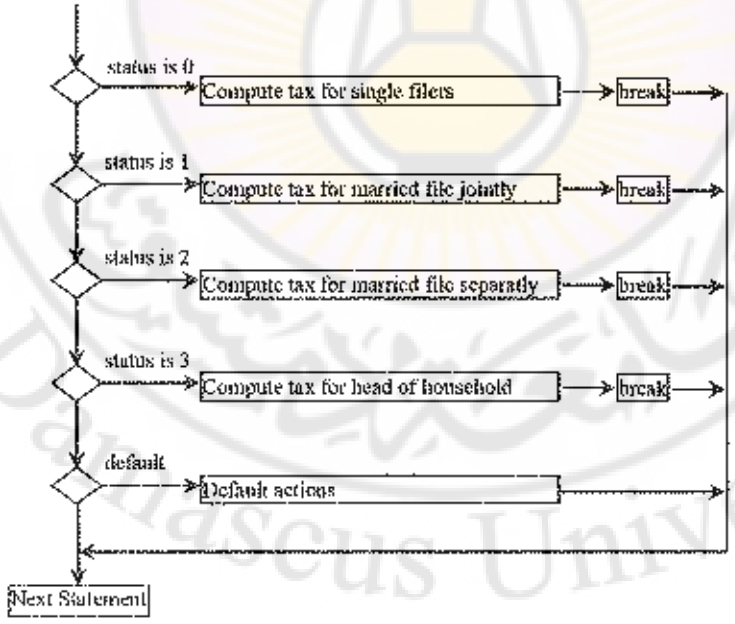
البرنامج :

```
//calculate the mark of a student
using System ;
class studentGreade {
public static void Main(string [ ] args ) {
    double grade;
    Console.WriteLine("Enter  a  grade :");
    grade = double.Parse( Console.ReadLine() );
    if (grade >= 90)
      Console.WriteLine("The mark is A\n ");
    else if (grade >= 80 )
       Console.WriteLine("The mark is B\n ");
    else if (grade >= 70 )
      Console.WriteLine("The mark is C\n ");
    else if (grade >= 60  )
      Console.WriteLine("The mark is D\n ");
    else if (grade < 60) {
       Console.WriteLine("The mark is F\n");
        Console.WriteLine("You must take this
                                  course again");
      }//end if
   } // end method main
} // end class Name
```

```
RESULT
Enter a grade :
55
The mark is F
You must take this course again.
Press any key to continue . . .
```

٤-٢-٤- بنية التعليمة switch

تستخدم هذه البنية جملة الشرط المغلقة Close Condition أي الشرط الذي يأخذ قيمة محددة (رقمية أو محرفية) ولهذا يمكن القول بأن هذه البنية تتعامل مع متحول واحد يمكن أن يأخذ في كل مرة قيمة محددة من خلال مجموعة من القيم. التعليمة switch هي تعليمة تتحكم بتدفق البرنامج ، وتبدأ بتعبير expression وتنقل التحكم إلى إحدى عبارات case بناءً على قيمة ذلك المتحول وتعمل التعليمة switch مع الأنماط الأولية: int و short و byte و char وأنماط التعداد enumeration ، وتُستخدم عبارة break من أجل الخروج من العبارة switch ، وإذا لم تحتوي عبارة case على الكلمة break فإنه سيتم تنفيذ السطر الموجود بعد نهاية العبارة case ، ويستمر التنفيذ إلى أن يتم الوصول إلى عبارة break أو الوصول إلى نهاية عبارة switch ، ويُسمح بعلامة default واحدة فقط في عبارة switch. ويبين الشكل (٤-٢) المخطط التدفقي للتعليمة switch .



الشكل (٤-٤) – يبين المخطط التدفقي للتعليمة switch

والشكل العام لبنية التحكم **switch** هو التالي:

```
switch( expression) {          // int expression
   case value 1: statement(s) ; // int value
     break;
   case value 2: statement(s);  // int value
     break;
     :
   case value n: statement(s);  // int value
     break;
   default :statement(s);
}//end switch
```

هذا الشكل يمثل الاحتمالات التي يمكن أن يأخذها المتحول variable والذي يمكن أن يكون على شكل تعبير expression، كل احتمال يتم وضعه داخل جملة باستخدام العبارة case ثم بعد ذلك كتابة الحدث الذي يمكن أن يكون تعليمة أو مجموعة تعليمات ولا بد أن ينتهي الحدث بواسطة العبارة break التي تمثل نهاية الجملة ،أما الاحتمال default يمثل الحدث الملازم للحالة خارج النطاق ، كما هو موضح في المثال (٤-٤) .

مثال (٤-٤)

```
//calculate the mark of a student
using System ;
class studentFood {
public static void Main(string [ ] args ) {
int food =0;
food = int.Parse( Console.ReadLine() );
// start switch
```

```
switch(food) {
  case 1:
    Console.WriteLine("Chicken");
    break;
  case 2:
    Console.WriteLine("Pizza");
    break;
  default:
    Console.WriteLine("Sorry, we are out");
   break;
}// end switch
} // end method main
} // end class Name
```

```
RESULT
10
Sorry, we are out
Press any key to continue . . .
```

مثال (٤-٥)

اكتب برنامجاً بلغة C# يقوم بما يلي(استخدم التعليمة switch) :

- إدخال رقم الشهر month (1,2,3,...,12).
- طباعة الفصل الذي ينتمي إليه الشهر .
- حيث أرقام الأشهر التي تنتمي للفصل كما يلي :

Winter(1,2,1,2) , Spring(3,4,5), Summer(6,7,8), Autumn(9,10,11).

```
// An improved version of the season program.
using System ;
class Switch {
public static void Main(string [ ] args ) {
int month = 4;
string season;
    switch (month) {
    case 12:
    case 1:
    case 2:
          season = "Winter";
    break;
    case 3:
    case 4:
    case 5:
          season = "Spring";
    break;
    case 6:
    case 7:
    case 8:
          season = "Summer";
    break;
    case 9:
    case 10:
    case 11:
          season = "Autumn";
```

```
    break;
default:
        season = "Bogus Month";
break;
}// end switch
Console.WriteLine("April is in the" + season +".");
  } // end method main
} // end class Name
```

```
RESULT
April is in the   Spring .
Press any key to continue . . .
```

مثال (٤-٦)

مندوب مبيعات يتقاضى راتباً شهرياً أساسيا مقداره (50000) ليرة ، ويتقاضى عمولة قدرها 4% إذا كانت المبيعات الشهرية لا تتجاوز (2000 000) ليرة ، ويتقاضى عمولة قدرها 6% إذا كانت المبيعات الشهرية لا تتجاوز (4000 000) ليرة، ويتقاضى عمولة قدرها 7% إذا كانت المبيعات الشهرية أكثر من ذلك. والمطلوب : اكتب برنامجاً بلغة #C يقوم بما يلي :

- إدخال الراتب الشهري وقيمة المبيعات الشهرية وطباعتها
- حساب وطباعة الدخل الشهري الكامل للموظف .

ملاحظة : افترض salary الراتب الشهري الأساسي ، amount المبيعات الشهرية، W العمولة ، Allsalary الراتب الشهري الكامل .

البرنامج :

```
using System ;
class salesPerson {
```

```
public static void Main(string[]args ) {
  int  S   , X ;
  double  W ,S1 ;
Console.WriteLine("inter your salary S & amount X");
 S = int.Parse( Console.ReadLine() );
 X = int.Parse( Console.ReadLine() );
Console.WriteLine("--------------------------");
if( X <= 1000 000 )
      W = X*4/100;
else if( X <= 4000 000 )
        W = X*6/100;
     else
        W = X*7/100;
 Console.WriteLine( " W= "  + W);
 Console.WriteLine("--------------------------");
  S1 = S + W ;
 Console.WriteLine("Allsalary =  " + S1);
 Console.WriteLine("-------------------------");
  } // end method main
} // end class
```

```
RESULT
salary  S & amount  X

50000    3 000 000
-------------------------------
 W=180 000 //  العمولة
-------------------------------
 Allsalary = 230 000 //  الراتب
-------------------------------
Press any key to continue . . .
```

مثال (٤-٧)

اكتب برنامجاً بلغة #C يقوم بإدخال عددين صحيحين x، y ، وإدخال الرمز ch من نوع char بحيث إذا كان الرمز المدخل(+) يقوم بجمع العددين ، وإذا كان (-) يقوم بطرح العددين ، وإذا كان (*) يقوم بضرب العددين ، وإذا كان (/) يقوم بقسمة العددين وطباعة النتيجة .

البرنامج :

```
using System ;
class Calculation {
public static void Main(string[]args )
{
  double   x , y , z;
  char   ch ;
  Console.WriteLine("inter yuor x  &  y & ch ");
  x = double.Parse( Console.ReadLine() );
  y = double.Parse( Console.ReadLine() );
  ch = char.Parse(Console.Read() ) ;
  Console.WriteLine("------------------------");
  Console.WriteLine( " x = " +x+"   y="+y) ;
  Console.WriteLine("------------------------");
  if( ch == '+')
    Console.WriteLine( x + y );
  //---------------------------
   else if( ch == '-')
       Console.WriteLine( x - y );
  //---------------------------
   else  if( ch == '*')
       Console.WriteLine( x * y );
  //----------------------------
   else if( ch == '/')
       Console.WriteLine( x  / y );
```

```
  //-----------------------------
   else
       Console.WriteLine( "Sumbol  Erorr Input ");
   Console.WriteLine( "------------------------");
  } // end method main
} // end class Name
```

```
RESULT
x  &  y & ch
7    8    *
---------------------------
x = 7   y=8
z= 56
---------------------------
Press any key to continue . . .
```

البرنامج التالي هو اختصار للبرنامج في المثال_(٤-٧) ، لقد وضعنا في بداية المقارنة تعليمة if ، بحيث يتم المقارنة مع المحرف المُدخل ، اذا كان المحرف ليس رمز عملية حسابية فإنه يُنهي تنفيذ البرنامج ويطبع رسالة :

Symbol Error Input

واستخدامنا عبارة طباعة واحدة لطباعة نتيجة العملية الحسابية .

البرنامج :

```
sing System;
class Calculation  {
public static void Main(string[] args)
{
   double x, y, z=0;
   char ch;
   Console.WriteLine("inter yuor x  &  y & ch ");
    x = double.Parse(Console.ReadLine());
    y = double.Parse(Console.ReadLine());
```

```
    ch = char.Parse(Console.ReadLine());
  if (ch =='+'|| ch =='-'|| ch =='*'|| ch =='/')
  {
    if (ch == '+')
       z = x + y;
//---------------------------
   else if (ch == '-')
       z = x - y;
//---------------------------
   else if (ch == '*')
       z = x * y;
//---------------------------
   else if (ch == '/')
        z = x / y;
 //---------------------------
   Console.WriteLine("------------------------");
   Console.WriteLine(" x " + ch + " y  =" + z );
   Console.WriteLine("------------------------");
}// end first if
  else
     {
       Console.WriteLine("Symbol  Error Input ");
      Console.WriteLine("----------------------");
        }
    } // end method main
} // end class Name
```

٤-٣- بنى حلقات التكرار Looping Structures

إن استخدام حلقات التكرار يُعد من المواضيع المهمة في البرمجة، حيث إن العديد من المسائل تحتاج إلى تكرار تعليمة أو مجموعة تعليمات أكثر من مرة، أو يتم التكرار لطالما أن شرط ما محقق، وفي لغة C# هناك ثلاث بنى للتكرار وهي :

حلقة for ، و while ، و do-while ، و foreach .

٤-٣-١- بنية الحلقة for

تُستخدم الحلقة for لتكرار مجموعة من التعليمات عدداً محدداً من المرات، وتحتوي على ثلاثة أقسام هي : التهيئة والتعبير والتحديث .

- ❑ التهيئة : هو تعبير يقوم بتهيئة متحول الحلقة initialization، لإعطاء قيمة ابتدائية لعداد(متحول) الحلقة .
- ❑ التعبير termination : يمثل شرط التوقف ويتم اختياره في بداية كل تكرار ويحدث التكرار فقط في حال كانت قيمة التعبير هي true .
- ❑ التحديث increment : يمثل مقدار الزيادة أو النقصان لعداد(متحول) الحلقة.

الصيغة العامة :

```
for ( initial   exp ;  condition  exp ;   incre/decre   exp)
        <statement>;
Or  compound statement (block) {  ...   }
```

حيث:

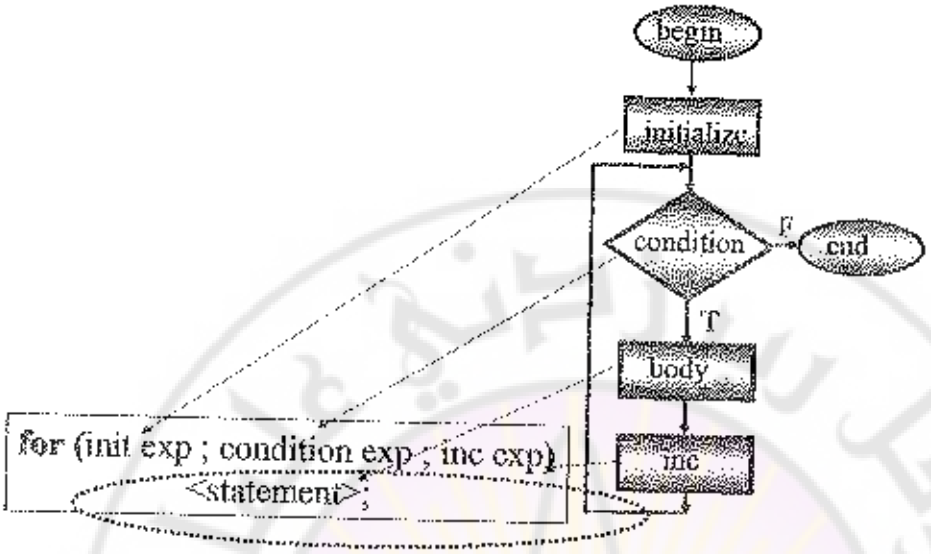
initial- exp – القيمة الابتدائية لعداد حلقة التكرار for .

condition exp – شرط حلقة التكرار وغالباً ما يحوي قيمة نهائية.

incre/decre exp – الزيادة المنتظمة لعداد الحلقة .

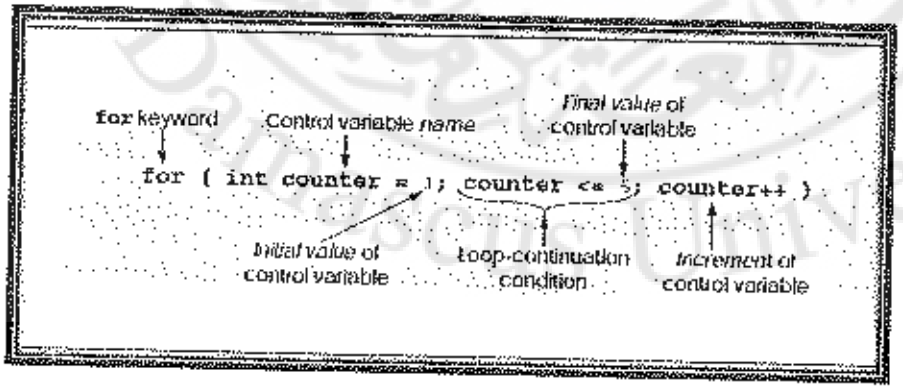
Statement – تعليمة يتم تكرار تنفيذها لعدد من المرات .

يبين الشكل(٤ – ٥) الحلقة for وما يقابلها من المخطط .



الشكل(٤ – ٥) – يبين الحلقة for وما يقابلها من المخطط

تقوم الحلقة for مبتدئة بقيمة العداد الابتدائية بتنفيذ التعليمة Statement في المرة الأولى، وفي المرة التالية تزداد القيمة الابتدائية للعداد بمقدار الزيادة ثم تنفذ التعليمة Statement مرة ثانية، وهكذا حتى يصبح الشرط condition غير محققاً لإنهاء عملية التكرار والخروج من الحلقة for . يوضح الشكل التالي حلقة for لتكرار خمس مرات لعدد من التعليمات .



يمكن أن يكون الشرط ومقدار الزيادة أو النقصان في الحلقة تعبير رياضي أو أحدهما كما يلي :

```
for ( int j = x ; j <= 4 * x * y  ; j += y / x  )
```

أمثلة على الحلقة for

توضح الأمثلة التالية الحلقة for وكيف يتم تغيير القيمة الابتدائية والشرط ومقدار الزيادة والنقصان ، كما يلي :

a) Vary the control variable from 1 to 100 in increments of 1.

```
for ( int i = 1; i <= 100; i++ )
```

b) Vary the control variable from 100 to 1 in increments of -1

```
for ( int i = 100; i >= 1; i-- )
```

c) Vary the control variable from 7 to 77 in steps of 7.

```
for ( int i = 7; i <= 77; i += 7 )
```

d) Vary the control variable from 20 to 2 in steps of -2.

```
for ( int i = 20; i >= 2; i -= 2 )
```

e) Vary the control variable over the sequence of the following values: 2, 5, 8, 11, 14, 17, 20.

```
for ( int j = 2; j <= 20; j += 3 )
```

f) Vary the control variable over the sequence of the following values: 99, 88, 77,66, 55, 44, 33, 22, 11, 0.

```
for ( int j = 99; j >= 0; j -= 11 )
```

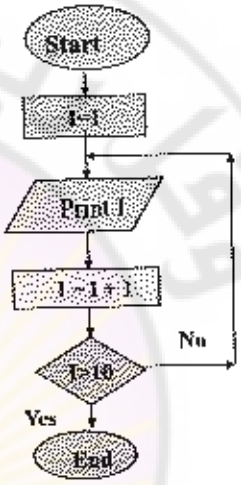
مثال (٤-٨)

اكتب برنامجاً بلغة C++ يقوم بطباعة الأعداد الصحيحة من 1 حتى 10.

(استخدم الحلقة for)

البرنامج:

```
using System ;
class Increment {
public static void Main(string[]args )
{
   for ( int i=1;i<=10 ; i++ )
     Console.Write ( " " + i  ) ;
  Console.WriteLine ( ) ;
  } // end method main
} // end class Name
```




```
RESULT
1  2  3  4  5  6  7  8  9  10
Press any key to continue . . .
```

عندما يحتوي جسم الحلقة على أكثر من تعليمة يجب وضع جسم الحلقة ضمن قوسين { ..... } ، كما يلي :

```
for (initial exp ;  condition exp ; incre exp )
 {
        Statement 1;
        ..............
        Statement n;
 }
```

مثال (٤-٩)

اكتب برنامجاً بلغة C# يقوم بحساب وطباعة المتوسط الحسابي لعشرة أعداد حقيقية يتم إدخالها من لوحة المفاتيح (استخدم الحلقة for) .

البرنامج :

```
// Add 10 numbers and print the average
using System ;
class average {
public static void Main(string [ ] args ) {
double N ,avr , sum = 0;
  for (int  i = 0; i <= 9; i++){
      N = double.Parse( Console.ReadLine() );
      sum = sum +N;
  } //End for
avr = sum / 10 ;
Console.WriteLine(" sum =" +sum+"  are= "+avr ) ;
 } // end method main
  } // end class Name
```

```
RESULT
Sum =  45
Press any key to continue . . .
```

مثال (٤-١٠)

اكتب برنامجاً بلغة C# يقوم بحساب وطباعة المتوسط الحسابي لعشرة أعداد حقيقية يتم إدخالها من لوحة المفاتيح (استخدم الحلقة for).

```
// Add 10 numbers and print the average
using System ;
class average {
public static void Main(string[]args )
{
 double N ,avr , sum = 0;
   for (int  i = 0; i <= 9; i++){
     N = double.Parse( Console.ReadLine() );
      sum = sum +N;
   } //End for
      avr = sum / 10 ;
 Console.WriteLine(" sum =" +sum+"  are= "+avr ) ;
 } // end method main
  } // end class Name
```

RESULT

```
Enter your numbers :
  2.4  5.6  6.7  5.7  7.8  5.5  6.7  4.6  6.0  7.0
average=5.8
Press any key to continue . . .
```

مثال (٤-١١)

يعمل 100عامل في معمل نسيج بشكل ساعي ، لنفرض أن H هي عدد ساعات العمل وأن R هي أجرة الساعة الواحدة وأن W هو الأجر اليومي . والمطلوب :

اكتب برنامجاً بلغة C# يقوم بما يلي :

- إدخال وطباعة عدد ساعات العمل وأجرة الساعة .
- حساب وطباعة الأجر اليومي لكل عامل .
- استخدم الحلقة for فقط .

البرنامج :

```
using System ;
class Worker {
public static void Main(string[]args)
{
  double  H , R ,W;
  for (int  i = 1; i <= 5; i++){
      Console.WriteLine(" Enter R & H " ) ;
      R = double.Parse( Console.ReadLine() );
      H = double.Parse( Console.ReadLine() );
      W = R * H ;
      Console.WriteLine(" W = {0}  " , W ) ;
  }//End for
 } // end method main
} // end class Name
```

٤-٣-٣- بنية الحلقة do / while

تُستخدم الحلقة do-while لتكرار تنفيذ تعليمة أو مجموعة من التعليمات أكثر من مرة بناءً على شرط معين كما هو الحال مع الحلقة while ولكن الفرق بينهما هو أن الحلقة while تختبر الشرط أولاً ، فإذا كان الشرط صحيحاً true عندئذ تنفذ التعليمات الموجودة داخل الحلقة ، وإلا ويتم الانتقال إلى تنفيذ التعليمات التي تلي الحلقة, أما الحلقة do-while فتنفذ التعليمات الموجودة ضمن الحلقة ثم تختبر الشرط، فإذا كان الشرط صحيحاً true يتم تكرار تنفيذ الحلقة وإلا تُوقف التكرار أي يتم تنفيذ الحلقة مرة واحدة على الأقل ، والصيغة العامة لبنية الحلقة do-while كما يلي :

```
do
  {
    Statement1;
    :
    Statement n;
    increment or decrement;
  }
while (condition);
```

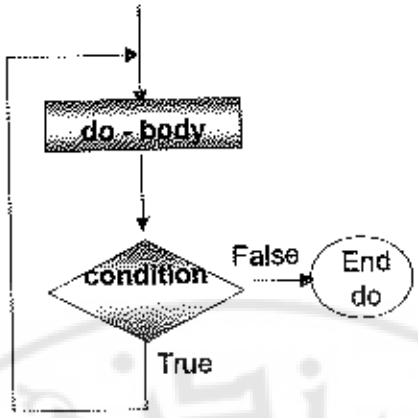
حيث:

condition – شرط تكرار الحلقة وغالباً ما يحوي قيمة نهائية.

decrement /increment – الزيادة المنتظمة في العداد أو النقصان.

Statements – مجموعة التعليمات التي يتم تكرار تنفيذها عدداً محدداً من المرات.

يبين الشكل (٤- ٧) المخطط التدفقي لبنية الحلقة do- while .



الشكل (٤ – ٧) – يبين المخطط التدفقي لبنية الحلقة do- while

مثال (٤–١٠)

اكتب برنامجاً بلغة #C يقوم بحساب مجموع الأعداد الصحيحة من 1 حتى 10 وطباعتها باستخدام الحلقة do / while.

```
/* Add the numbers from 0 to 9 and print the result
  do / while loop example .
*/
using System ;
class SumNumber {
public static void Main(string [ ] args ) {
int i=0, sum = 0;
  do
  {
    sum = sum + i; // sum += i;  accumulator
    i++;  // i = i + 1;  counter
  }
  while(i <= 9) ;  // end do /while
```

```
 Console.WriteLine("Sum = " + sum + ".\n");
  } // end method main
} // end class Name
```

```
RESULT
Sum =  45
Press any key to continue . . .
```

مثال (٤-١٦)

اكتب برنامجاً بلغة C# يقوم بإدخال 30عدداً حقيقياً وطباعة كلمة "Negative" إذا كان العدد أصغر من الصفر، ويطبع كلمة "Positive" إذا كان العدد أكبر أو يساوي الصفر ، وذلك باستخدام الحلقة do-while .

البرنامج :

```
using System ;
class Number {
public static void Main(string[]args )
{
 double number;
 int i=1;   // القيمة الابتدائية
 do
 {
  Console.WriteLine(" Enter degree ");
  number = double.Parse( Console.ReadLine() );
  if (number >= 60 )
```

```
 Console.WriteLine(" Positive = {0}", number );
 else
 Console.WriteLine(" Negative = {0}", number );
 i++;   // i = i + 1;  counter
 }// end do/while
 while(i <= 30) ; // الشرط
 } // end method main
} // end class
```

مثال (٤-١٧)

لدى شركة تجارية 30 مندوب مبيعات يتقاضى كل مندوب راتباً شهرياً أساسيا مقداره (50000) ليرة ، ويتقاضى عمولة قدرها 4% إذا كانت المبيعات الشهرية لا تتجاوز (2000 000) ليرة ، ويتقاضى عمولة قدرها 6% إذا كانت المبيعات الشهرية لا تتجاوز (4000 000) ليرة ، ويتقاضى عمولة قدرها 7% إذا كانت المبيعات الشهرية أكثر من ذلك . والمطلوب :

اكتب برنامجاً بلغة Java يقوم بما يلي :

- إدخال الراتب الشهري وقيمة المبيعات الشهرية وطباعتها لجميع المندوبين salary ,W ,Allsalary, amount
- حساب وطباعة الدخل الشهري الكامل للمندوبين .

<u>ملاحظة</u> : افترض salary الراتب الشهري الأساسي ، amount المبيعات الشهرية ، W العمولة ، Allsalary الراتب الشهري الكامل .

البرنامج :

```
using System ;
class salesPerson {
public static void Main(string[]args ) {
  int  salary   , amount ;
  double  W ,Allsalary ;
 for ( int i=1 ; i<= 3 ; i++ )
 {
   Console.WriteLine("inter yuor salary & amount");
  salary = double.Parse( Console.ReadLine() );
  amount = double.Parse( Console.ReadLine() );
  Console.WriteLine("-------------------------");
 if( amont <= 200000 )
    W=amount*4/100;
 else if( amount <= 400000 )
    W=amount*6/100;
 else
    W=amount*7/100;
 Console.WriteLine("W= {0}"  , W);
 Console.WriteLine("-------------------------");
 Allsalary = salary + W ;
 Console.WriteLine("Allsalary = {0} " , Allsalary);
 Console.WriteLine("-------------------------");
}//End for
} // end method main
} // end class Name
```

٤-٤- تعليمات التفريع Branching Statement

٤-٤-١- تعليمة التوقف break

تُستخدم تعليمة التوقف break من أجل تغيير مجرى تدفق التحكم ضمن البرنامج وتسبب التعليمة break عند تنفيذها مع الحلقات : for و while و do/while الخروج من هذه الحلقات ويتابع البرنامج بعدها التنفيذ مع أول تعليمة تلي الحلقة.

مثال (٤-١٨)

البرنامج التالي يوضح لنا كيفية استخدام التعليمة break مع بنية التكرار for.

```
using System ;
public class break1 {
public static void Main(string[]args )
{
 int x;
 for ( x = 1;  x<= 10; x++ )
 {
  if ( x == 4 )
  break ;
  Console.WriteLine("x= {0}" , x);
 }// end for
 Console.WriteLine("\nBroke out of loop at x ="+ x);
 } // end method main
} // end class
```

مثال (٤-٢١)

البرنامج التالي يحسب الفائدة المركبة ، لمستثمر يريد أن يريد أن يودع مبلغ 1000.00$في حساب توفير يحقق فائدة 5% ، إذا افترضنا أن جميع الفوائد متبقية عند الإيداع ، والمطلوب حساب الفائدة المركبة في نهاية كل عام لمدة عشر سنوات ، يتم حساب الفائدة المركبة من العلاقة التالية :

$$a = p\,(1 + r)^n$$

حيث :

P: هو المبلغ الأصلي المستثمر (أي المبلغ الرئيسي)

r : هو معدل الفائدة السنوي

n : هو عدد السنوات

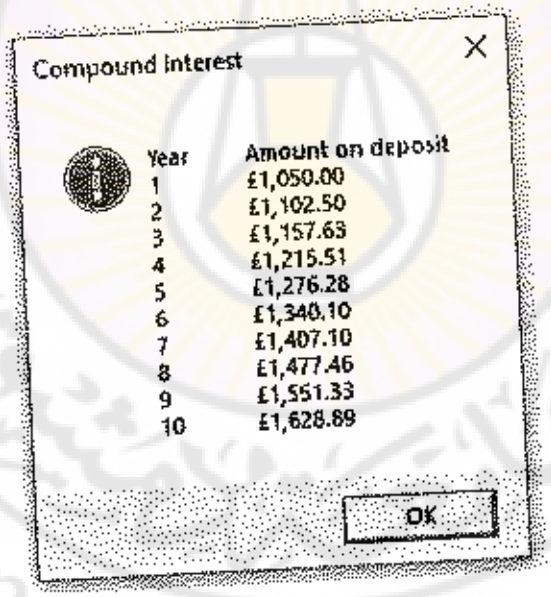
a : هو المبلغ المودع في نهاية السنة التاسعة.

ويطبع الفائدة والمبلغ في كل سنة على واجهة بيانية (نافذة) ، باستخدام MessageBox.Show .

البرنامج :

```
// Calculating compound interest.
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication3 {

 static class Program
 {
   static void Main()
  {
  decimal amount, principal = ( decimal ) 1000.00;
  double rate = .05;
  string output;
```

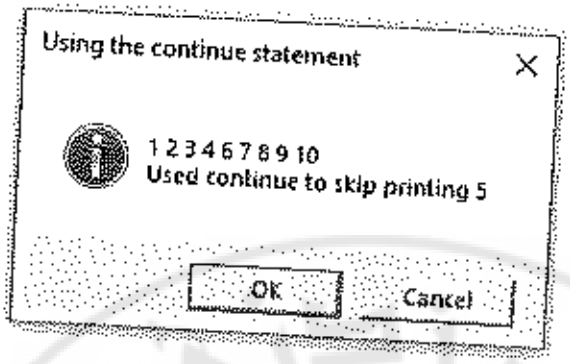
```
output = "Year\tAmount on deposit\n";
for ( int year = 1; year <= 10; year++ )
{
    amount = principal *
    ( decimal ) Math.Pow( 1.0 + rate, year );
    output += year + "\t" +
   String.Format( "{0:C}", amount ) + "\n";
 }// end for

 MessageBox.Show( output, "Compound Interest",
 MessageBoxButtons.OK, MessageBoxIcon.Information );

  } // end method Main
 } // end class Interest
}// end namespace
```



إذا عدلنا نوع الزر كما يلي :

```
MessageBoxButtons.YesNo
```

سوف نحصل على الواجهة البيانية التالية :

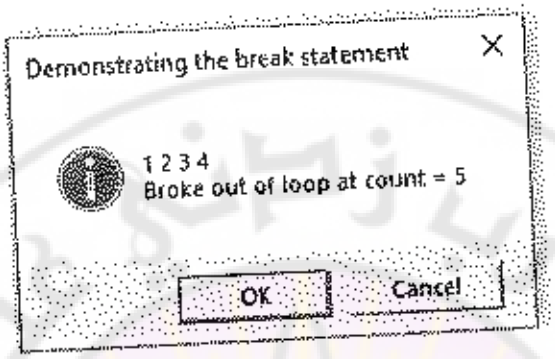


٢- البرنامج التالي يوضح لنا كيفية استخدام التعليمة break مع الحلقة for باستخدام وجهة MessageBox.Show.

البرنامج :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1 {
static class BreakTest {
  static void Main()
  {
   string output = "";
   int count;
   for ( count = 1; count <= 10; count++ )
  {
    if ( count == 5 )
     break; // skip remaining code in loop
          // if count == 5
    output += count + " ";
  } // end for loop
output +="\nBroke out of loop at count =" + count;

MessageBox.Show( output,
            "Demonstrating the break statement",
            MessageBoxButtons.OKCancel,
            MessageBoxIcon.Information);
```

```
  } // end method Main
 } // end class BreakTest
} // end class
```



٣- يقوم البرنامج التالي بإدخال عدد صحيح n موجب وحساب العاملي لهذا العدد وطباعة النتيجة .

- عند إدخال عدد سالب يقوم بطباعة العبارة

Error: n is a negative number

- عندما يكون العدد صفر أو واحد يطبع النتيجة Fact = 1
- أما إذا كان العدد أكبر من الواحد يقوم بحساب وطباعة العاملي

البرنامج :

```
using System;
class Factorail  {
public static void Main(string[] args) {
   double n, Fact = 1;
Console.WriteLine("Enter an positive integer to
                          find its factorial: ");
   n = double.Parse(Console.ReadLine());
  if (n < 0)
Console.WriteLine( "Error: " + n +
                        " is a negative number" );
else if ((n == 0) || (n == 1))
   Console.WriteLine( n + "! = " + Fact );
```

```
else
 {
  int i = 1;
  while (i <= n) {
     Fact = Fact * i;
     i++;
 }  //end while
 Console.WriteLine( n + "! = " + Fact );
 }// else
}//end main
}//end class
```

٤- يقوم البرنامج التالي بحساب وطباعة جدول الضرب على نافذة بحيث :

- نستخدم حلقات for المتداخلة .
- يقوم البرنامج بعرض العنوان في السطر الأول ثم يقوم بإضافة خطوط متقطعة (-) في السطر الثاني من الخرج .
- تعرض حلقة for الأولى الأعداد من 1 حتى 9 في السطر الثالث .
- الحلقة التالية المتداخلة بحيث يتم عرض ناتج (i * j) بعدد مرات المتحول j والذي يبدأ من 1 حتى 9 .
- استخدمت التعليمة if في الحلقة الداخلية من أجل رصف الناتج بشكل مناسب ، حيث إذا كانت قيمة الناتج مؤلفة من رقم وحيد يتم عرضه مع وضع مسافة إضافية قبله .

البرنامج :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1 {
static class MulTable
{
```

```
public static void Main( string[ ] args)
{
// Display the table heading
String output = " الضرب جدول \n  Multiplication  Table\n";
output += "-----------------------------------------\n";
// Display the number title
   output += "   | ";
   for (int j = 1; j <= 9; j++)
     output += "    " + j;
   output += "\n";
   // Print table body
   for (int i = 1; i <= 9; i++) {
     output += i + " | ";
     for (int j = 1; j <= 9; j++) {
//Display the product and align properly
       if (i * j < 10)
         output += "    " + i * j;
       else
         output += "  " + i * j;
       }
       output += "\n";
      }
      // Display result

      MessageBox.Show(output, "Demonstrating the break statement",
        MessageBoxButtons.OKCancel,  MessageBoxIcon.Information);

      }//End main
     }//End class
     }// end namespace
```

Demonstrating the break statement

جدول الضرب
Multiplication Table

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

OK Cancel

مسائل عامة

١- اكتب برنامجاً بلغة C# يقوم بحساب مجموع الأعداد الصحيحة الفردية من 10 حتى 20 وطباعتها .

٢- اكتب برنامجاً بلغة C# يقوم بحساب مجموع الأعداد الصحيحة الزوجية من 50 حتى 200, وطباعتها ، ثم ارسم المخطط التدفقي .

٣- اكتب برنامجاً بلغة C# يقوم بحساب مجموع مضاعفات العدد 5 للأعداد الصحيحة من 1 حتى 100, وطباعتها .

٤- لدينا 300 طالباً يدرسون مقرر البرمجة والمطلوب :

اكتب برنامجاً بلغة C# يقوم بما يلي :

- إدخال درجة الطالب d وطباعة كلمة ناجح" Passed " إذا كانت درجة الطالب $d \geq 60$ ويطبع كلمة راسب " Failed " إذا كانت غير ذلك .
- استخدم الحلقة while .

٥- اكتب برنامجاً بلغة C# يقوم بما يلي :

- إدخال عددين صحيحين a و b .
- إيجاد وطباعة المضاعف المشترك الأصغر للعددين المُدخلين.

٦- اكتب برنامجاً بلغة C# يقوم بما يلي:

- إدخال عددين صحيحين a و b .

- إدخال قيمة صحيحة للمتحول m بحيث إذا كانت m=1 يقوم البرنامج بإيجاد العدد الأكبر من بين العددين المُدخلين ، وإذا كانت m=2 يقوم البرنامج بحساب المتوسط الحسابي للعددين المُدخلين ، وإذا كانت m=3 يقوم البرنامج بإيجاد المضاعف المشترك الأصغر للعددين المُدخلين .
- استخدم الحلقة switch .

٧- اكتب برنامجاً بلغة C# يقوم بما يلي:

- إدخال عددين صحيحين a و b .
- إدخال قيمة محرفية(* أو / أو + أو - أو أي قيمة أخرى) .
- استخدم التعليمة switch من أجل إجراء العمليات الحسابية الأساسية (الضرب و القسمة و الجمع و الطرح) على العددين المُدخلين a و b.

٨- اكتب برنامجاً بلغة C# يقوم بإيجاد وطباعة درجة الحرارة العظمى خلال شهر تشرين الأول (31 يوم)، باستخدام الحلقة while .

٩- اكتب برنامجاً بلغة C# يقوم بإدخال عدد صحيح m ومن ثم يطبع الجملة التالية : "This number m is positive" إذا كان العدد m أكبر أو يساوي الصفر أو يطبع الجملة : "This number m is negative"إذا كان العدد m أصغر من الصفر.

١٠- اكتب برنامجاً بلغة C# يقوم بإدخال a,b,c وحساب وطباعة جذور معادلة من الدرجة الثانية:

$$ax^2+bx+c=0$$

١١- اكتب برنامجاً بلغة C# يقوم بحساب وطباعة قيمة الدالة الرياضية المعرفة كما يلي:

$$f = \begin{cases} x^2 + 3 & \text{if } 5 \le x \le 10 \\ x^3 + \dfrac{3 - x}{2} & \text{if } 10 < x \le 100 \\ \dfrac{x - 1}{25} & \text{otherwise} \end{cases}$$

١٢- اكتب برنامجاً بلغة C# يقوم بإدخال عددٍ صحيح a ويطبع "zero" إذا كان a=0 أو يطبع "one" إذا كان a=1 أو يطبع "Two" إذا كان a=2 أو يطبع "No thing" في باقي الحالات.

١٣- اكتب برنامجاً بلغة C# يقوم بجمع سلسة تبدأ بالقيمة 0.01 وتنتهي بالقيمة 1.0، وستتم زيادة الأعداد في السلسلة بمقدار 0.01 ، وذلك كما يلي :

0.01 + 0.02+ 0.03 + . . . etc

الفصل الخامس

التوابع (الطرائق)

Methods

التوابع (Methods)

٥ – ١ – مقدمة

تتجلى إحدى الفوائد الأساسية من استخدام التوابع عند كتابة البرامج بتقسيم العمليات الحسابية الطويلة والمعقدة إلى مجموعات من العمليات الصغيرة والبسيطة نسبياً، وينتج عن ذلك إمكانية إعادة استخدام هذه التوابع في العديد من البرامج بدلاً من إعادة كتابة العمليات نفسها في كل مرة. وتسمح التوابع، عند كتابتها بالطريقة المناسبة، بإخفاء التفاصيل الصغيرة للعمليات ما يسهل على المبرمج توضيح البنية العامة لبرامجه ويخفف من صعوبة تعديل هذه البرامج، وقد برهنت التجربة على أن أفضل طريقة لتطوير وصيانة برنامج كبير ، تتمثل في عملية تجزئته إلى قطع وأجزاء أصغر يمكن التحكم بها بسهولة أكبر من البرنامج الأصلي. وتوجد في لغة #C مكتبة ضخمة من التوابع تقوم بتنفيذ العمليات الرياضية، والتعامل مع السلاسل والمحارف، والإدخال والإخراج، واكتشاف الأخطاء والعديد من العمليات الأخرى المفيدة مما يسهل مهمة المبرمج الذي يجد في هذه التوابع معيناً كبيراً له في عملية البرمجة. ويمكن للمبرمج كتابة توابع تقوم بأداء عمليات يحتاج لها المبرمج في برامجه .

٥ – ٢ – فوائد استخدام التوابع

من أهم فوائد التوابع في البرمجة ، هي :

١. تساعد التوابع المخزنة في ذاكرة الحاسوب على اختصار البرنامج إذ يكتفى باستدعائها باسمها فقط لتقوم بالعمل المطلوب .

٢. تساعد البرامج المخزنة في ذاكرة الحاسوب أو التي يكتبها المستخدم على تلافي عمليات التكرار في خطوات البرنامج التي تتطلب عملاً مشابهاً لعمل تلك التوابع .

٣. تساعد التوابع الجاهزة في تسهيل عملية البرمجة.

٤. يوفر استعمال التوابع من المساحات المستخدمة في الذاكرة.

٥. كتابة برنامج في لغة C# بشكل واضح المعالم يجعل البرنامج واضحاً لكل من المبرمج والقارئ على حد سواء.

٦. التوابع تمكن المبرمج من تقسيم البرنامج إلى وحدات modules، وكل تابع في البرنامج يمثل وحدة قائمة بذاتها، ولذا نجد أن المتغيرات المعرفة في التابع تكون متحولات محلية (Local) ونعني بذلك أن المتحولات تكون معروفة فقط داخل التابع.

٧. أغلب التوابع تمتلك لائحة من الوسائط (Parameters) والتي هي أيضاً متحولات محلية .

٥-٣- نموذج التابع

Method Prototype

التابع Method، هو مجموعة من التعليمات التي تنفذ عندما يتم استدعاؤه من مكان ما في البرنامج ، وفي الواقع، والتابع هو برنامج جزئي يمكن أن ينفذ على معطيات معينة وبعد ذلك يعيد قيمة إلى التابع المستدعي.

وعندما يولد المترجم تعليمات لاستدعاء تابع ما فإنه يحتاج إلى معرفة اسم التابع وعدد وسطائه وأنواعها ونوع قيمة الإعادة ، لذا علينا كتابة نموذج أو تصريح للتابع ضمن الصف الذي يحتوي على التابع Main (قبل التابع Main أو بعده).

وعندما نصرح عن تابع ما فإننا نقوم فعلياً بتعريف مهمة هذا التابع و يبلغ المترجم عن اسم التابع وعدد وسطائه وأنواعها ونوع القيمة المعادة بواسطة التابع.

فمثلا في تصريح التابع التالي:

```
public static int sum( int x )
{
    جسم التابع
}
```

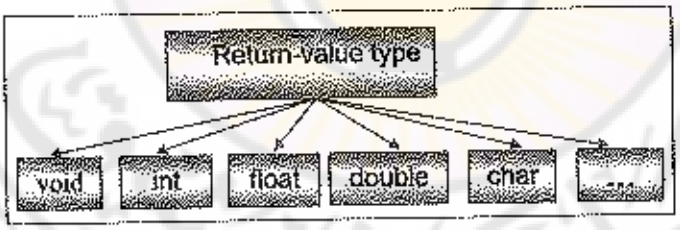
النوع int بين القوسين يخبر المترجم بأن الوسيط الذي سيتم تمريره إلى التابع سيكون من النوع int و int التي تسبق اسم التابع تشير إلى نوع القيمة المعادة بواسطة التابع. ويمكن أن تكون من نمط double أو char كما يلي :

```
public static double sum1(double x ) { . . . }
public static char sum2( char x ){ . . . }
```

أو لا يعيد قيمة وهو من نمط void

```
public static void sum( int x , double y ) {. . .}
```

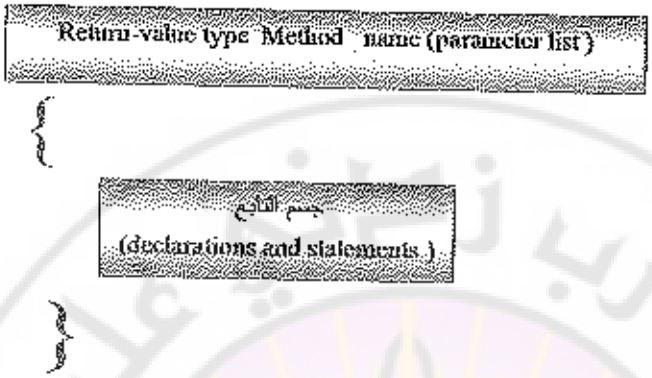
ونلاحظ أن كل تابع يملك الاسم الخاص به الذي يوضع من قبل المبرمج، وغالباً يكون اسم التابع قريباً من محتواه, وفيما يتعلق بنوع القيمة المعادة ، تحددها طبيعة المسألة المدروسة، فقد تكون من أي نوع من أنواع المعطيات المعروفة في لغة Java , كما هو موضح الشكل(٥-١) .



الشكل(٥-١) – يبين نوع القيمة المعادة

كل تابع يُستخدم في برنامج بلغة C# يجب أن يكون له تعريف داخل البرنامج (عدا التوابع الرياضية والتوابع الجاهزة الأخرى) .

لا بد من الإشارة إلى أن تعريف التابع يُعطي علماً المترجم كيف يعمل الأداء (مهمة التابع).ويبين الشكل(٥--٢)الشكل العام لتعريف التابع .



الشكل(٥--٢) الشكل العام لتعريف التابع

حيث:

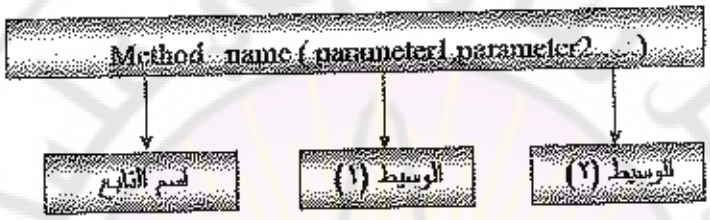
- ✓ Return-value type : نوع القيمة المعادة بوساطة التابع والذي يمكن أن يكون أي نوع من أنواع معطيات #C ، وإذا كان التابع لا يرجع أي قيمة يكون نوع إعادته void.
- ✓ Method _ name: اسم التابع والذي يتبع في تسميته قواعد تسمية المعرفات (identifiers).
- ✓ parameter list: هي لائحة الوسطاء الممررة إلى التابع والتي يمكن أن تكون خالية (void) أو تحتوي على وسيط واحد أو عدة وسطاء تفصل بينها فاصلة ويجب ذكر كل وسيط على حدة.
- ✓ declarations and statements: تمثل جسم التابع والذي يطلق عليه في بعض الأحيان كتلة (block) ، ويمكن أن تحتوي الكتلة(block) على تصريح عن المتحولات ولكن تحت أي ظرف لا يمكن أن يتم تعريف تابع داخل جسم تابع آخر.

السطر الأول في تعريف التابع يدعى المصرح declarator والذي يحدد اسم التابع ونوع المعطيات الذي يعيدها التابع وأسماء وسطائه وأنواعها .

٥ – ٤ – استدعاء التوابع

Calling Functions

كل تابع يستخدم في برنامج بلغة C# يجب أن يتم استدعاؤه من التابع الرئيسي main أو من أي تابع أخر. ويبين الشكل(٥–٣) الشكل العام للاستدعاء.



الشكل(٥–٣) – يبين الشكل العام للاستدعاء

يؤدى استدعاء التابع إلى انتقال التنفيذ إلى بداية التابع، ويمكن تمرير بعض الوسطاء إلى التابع عند استدعائه وبعد الانتهاء من تنفيذ التابع يعود التنفيذ للعبارة التي تلي استدعاء التابع في البرنامج الأصلي ، وبإمكان التابع أن يعيد قيمة إلى العبارة التي قامت باستدعائه، وإذا كان التابع لا يعيد شيئاً يجب استعمال الكلمة الأساسية void كنوع إعادة له للإشارة إلى ذلك . يوجد ثلاث طرق يمكن بها إرجاع التحكم إلى النقطة التي تم فيها استدعاء التابع ، وهي :

١. إذا كان التابع لا يرجع قيمة ، يرجع التحكم تلقائياً عند الوصول إلى نهاية التابع .

٢. باستخدام العبارة return ;

٣. إذا كان التابع يرجع قيمة فالعبارة return expression ; تقوم بإرجاع قيمة التعبير expression إلى النقطة التي استدعته .

مثلاً : لنصرح عن التابع sum يعيد قيمة من النوع int ويحتوي على وسيطين من النوع int ، ومن أجل استدعاء هذا التابع يجب أن يحتوي برنامج Java الذي يتضمن هذا التابع، على الاستدعاء التالي للتابع:

```
sum ( a , b );
```

حيث إن:

sum – اسم التابع.

a – الوسيط الأول (يجب أن يُعطى قيمة محددة أو يتم إدخاله من قبل المستخدم).

b – الوسيط الثاني (يجب أن يُعطى قيمة محددة أو يتم إدخاله من قبل المستخدم).

من غير الضروري أن تكون أسماء الوسطاء في التصريح هي نفسها المستعملة في تعريف التابع , في الواقع، المترجم يتجاهلها لكنها تكون مفيدة أحياناً للذين يقرؤون البرنامج ، ويمكن تعريف التابع sum كما يلي :

```
public static int   sum ( int x , int y )
{
  int s;
  s=a+b;
  return s;
}
```

مثال (٥-١)

لنأخذ برنامجاً يستخدم تابعاً يدعى square لحساب مربعات الأعداد من 1 إلى 10.

البرنامج :

```
using System ;

class SguareNumber {
public static void Main(string [ ] args )
{
    for (int  x = 1; x <= 10; x++){
       Console.Write ( "  {0}   " ,  square(x)  ) ;
    }// end for
  Console.WriteLine( ) ;

} // end method main
//now function definition

  public static int square( int y)
{
        return   y * y  ;
}// end function
} // end class
```

```
RESULT
1    4    9    16    25    36    49    64    81    100
Press any key to continue . . .
```

كل المتحولات المصرح عنها داخل التابع تُدعى متحولات محلية Local Variables ، وتكون معرفة فقط في التابع الحاوي عليها ، بينما المتحولات المصرح عنها في التابع الرئيسي main( ) تُدعى متحولات عامة Global Variables وتكون معرفة فقط في التابع الرئيسي. التابع الرئيسي Main( ) يمكن أن يستدعي توابع أخرى، وهذه التوابع بدورها يمكن أن تستدعي نفسها أو توابع أخرى. نعيد كتابة البرنامج السابق باستخدام تابع لا يعيد قيمة .

مثال (٥-٢)

```
using System ;
class SquareNumber {
public static void Main(string [ ] args )
{
   for (int  x = 1; x <= 10; x++){
      sguare(x) ;
   }// end for
         Console.WriteLine( ) ;
} // end method main

//now function definition
 public static void sguare( int y)
{
   Console.Write( " {0}   " , y * y  )  ;
}
} // end class
```

```
RESULT
1   4   9   16   25   36   49   64   81   100
Press any key to continue . . .
```

مثال (٥-٣)

البرنامج التالي يستخدم تابع يدعى maximum والذي يرجع العدد الأكبر بين ثلاثة أعداد صحيحة. يتم تمرير الأعداد كوسائط للتابع الذي يحدد العدد الأكبر بينهم ويرجعه للتابع Main باستخدام العبارة return ويتم تعيين القيمة التي تمت إعادتها إلى المتغير largest الذي تتم طباعته.

البرنامج :

```
using System ;

class Maximum {
public static void Main(string [ ] args )
{
   double  a, b, c;
   Console.WriteLine("Enter three integers: ") ;
   a = double.Parse( Console.ReadLine() );
   b = double.Parse( Console.ReadLine() );
   c = double.Parse( Console.ReadLine() );
   Console.WriteLine(" maximum ={0} " , maximum (a, b, c) ) ;

}// // end method  Main
        // function definition
public static double  maximum (double x, double y, double z)
{
  double  max = x;
   if (y > max)
   max = y;
```

```
    if (z > max)
    max = z;
   return   max ;
}//end function
} // end class
```

```
RESULT
Enter three integers : 22  85  17
Maximum is : 85
Press any key to continue . . .
```

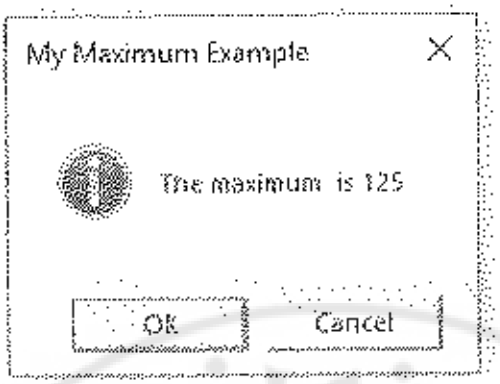
نعيد كتابة البرنامج السابق بحيث التابع من نوع void .

```
using System ;
class Maximum {
public static void Main(string [ ] args )
  {
   double  a, b, c;
   Console.WriteLine("Enter three integers: ") ;
    a = double.Parse( Console.ReadLine() );
    b = double.Parse( Console.ReadLine() );
    c = double.Parse( Console.ReadLine() );
        maximum (a, b, c)   ;
}  // end method main
            // function definition
public static void  maximum (double x, double y, double z)
{
   double  max = x ;
```

```
    if (y > x)
    max = y;
    if (z > max)
    max = z;
Console.WriteLine(" maximum= {0}  " , max) ;
}//end function
} // end class
```

نعيد كتابة البرنامج باستخدام الواجهات .

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1  {
  class Maximum {
    static void Main()
    {
      double a=77, b=125, c=25;
       MessageBox.Show(" The maximum  is " + maximum (a, b, c),
                  "My Maximum Example",
        MessageBoxButtons.OKCancel, MessageBoxIcon.Information);

      }// // end method Main
      // function definition
      public static double maximum(double x, double y, double z)
      {
        double max = x;
        if (y > max)
          max = y;
        if (z > max)
          max = z;
        return max;
      }//end function
  } // end class
}// end namespace
```



في هذه الحالة أسندنا قيم ابتدائية للأعداد الثلاثة ، وذلك بسبب إدخال القيم بوساطة تطبيقات الـ Windows.Forms ، التي سندرسها فيما بعد .

<u>ملاحظة</u>

- من غير الضروري أن تكون أسماء الوسطاء في التصريح عند استدعاء التابع هي نفسها المستعملة في تعريف التابع .
- في الواقع، المترجم يتجاهلها لكنها تكون مفيدة أحياناً للذين يقرؤون البرنامج . فمثلاً لنفترض أن الوسيطين x و y تمثلان إحداثيات نقطة على الشاشة.

[التصريح عند استدعاء التابع]: draw_dot (x, y)

- هذا التصريح كافي للمعرف لكن المبرمج قد لا يعرف أيهما الإحداثي x وأيهما الإحداثي y عند تعريف التابع إذا استخدم وسطاء تختلف عن وسطاء استدعاء التابع . لذا سيكون مفيداً لو كتبنا :

```
public static void draw_dot (int x, int  y) {  }
```

مثال (٥-٤)

البرنامج التالي يقوم بحساب مجموع الأعداد الصحيحة من 1 حتى 10 ، ومن 20 حتى 30 ومن 35 حتى 45 باستدعاء التابع sum أكثر من مرة .

البرنامج :

```
using System;
class  SumNumber {
static void Main(string[ ] args)  {
    Console.WriteLine("sum from 1 to 10 = "+ sum(1,10)) ;
    Console.WriteLine("sum from 20 to 30 = "+sum(20,30) ) ;
   Console.WriteLine("sum from 35 to 45 = " +sum(35,45)) ;
 } // end method main
          // function definition
public static int sum ( int i1 , int i2) {
       int sum1 = 0;
       for (int i=i1; i<= i2 ;i++)
       sum1 = sum1 + i ;
       return sum1 ;
   } // end  function
} // end class
```

```
RESULT
sum from 1  to 10 = 55
sum from 20 to 30 = 275
sum from 35 to 45 = 440
Press any key to continue . . .
```

٥ – ٥ – مكتبة التوابع الرياضية

Math Library Methods

تحتوي مكتبة التوابع الرياضية على العديد من التوابع التي تستخدم في تنفيذ العمليات الرياضية الحسابية، فمثلاً المبرمج الذي يرغب في حساب وطباعة الجذر التربيعي للعدد 900 قد يكتب العبارة التالية :

```
number = Math.sqrt ( 900 );
Console.WriteLine(" {0} " , number ) ;
```

عند تنفيذ هذه العبارة يتم استدعاء التابع المكتبي sqrt( ) لحساب الجذر التربيعي للعدد بين القوسين (900)، ويسمى العدد بين القوسين وسيط التابع argument وعليه فالعبارة السابقة تقوم بطباعة العدد 30 . ويأخذ التابع sqrt( ) وسيط من النوع double وتكون النتيجة قيمة من النوع نفسه وينطبق هذا على جميع التوابع الرياضية. ويوضح الجدول (٥–١) أهم التوابع الرياضية التي يمكن استخدامها من خلال مكتبة لغة Java ،ويتضمن الجدول أيضاً على وصف التوابع وأمثلة عليها . وعند استعمال التوابع الرياضية في أي برنامج بلغة C# يجب تضمين الصف Math والذي يحتوي على هذه التوابع .

- مثل :

Abs(x) ,Ceiling(x) , Cos(x) , Exp(x) , Floor(x) , Log(x) , Max(x) , Min(x),Pow(x), Sin(x) , Sqrt(x)

Methods	Description	Example
Sqrt(x)	الجذر التربيعي لـ x	Sqrt(900.0) is 30.0 Sqrt(9.0) is 3.0
Pow(a,b)	الرفع إلى قوة a^b	Pow(2.0, 7.0) is 128.0 Pow(9.0, .5) is 3.0
Exp(x)	التابع الأسي e^x	Exp(1.0) is approximately 2.7182818284590451 Exp(2.0) is approximately 7.3890560989306504
Abs(x)	القيمة المطلقة لـ x	Abs(23.7) is 23.7 Abs(0) is 0 Abs(-23.7) is 23.7
Ceiling(x)	تقريب x لأصغر عدد صحيح أكبر من x	Ceiling(9.2) is 10.0 Ceiling(-9.8) is - 9.0
Floor(x)	تقريب x لأكبر عدد صحيح أصغر من x	Floor(9.2) is 9 Floor(-9.8) is –10.0
Log(x)	اللوغاريتم الطبيعي لـ x	Log (2.718282) is 1.0
Log10(x)	اللوغاريتم العشري لـ x	Log10(1.0) is 0.0
Sin(x)	تابع الجيب لـ x	Sin(0.0) is 0.0
Cos(x)	تابع جيب التمام لـ x	Cos(0.0) is 1.0
Tan (x)	تابع الظل لـ x	Tan(0.0) is 0.0
Max(x,y)	ايجاد أكبر قيمة بين العددين x و y	Max(5,6)→ 6
Min(x,y)	ايجاد أصغر قيمة بين العددين x و y	Min(7,10)→ 7

الجدول (٥-١) - التوابع الرياضية لمكتبة لغة C#

مثال (٥-٥)

البرنامج التالي يستخدم التوابع الرياضية ضمن الصف Math ، ويقوم بحساب : القيمة المطلقة – الجذر التربيعي – الرفع إلى قوة – التوابع المثلثية – التابع الأسي واللوغاريتم .

البرنامج :

```
using System ;
class math1 {
public static void Main(string [ ] args )
{
 Console.WriteLine(" The pi number = "+Math.PI);
 Console.WriteLine(" Absolute of -15 is: = "+Math.Abs(-15));
 Console.WriteLine(" Sin(90) = "+Math.Sin((Math.PI)/2));
 Console.WriteLine(" Cos(90) = "+Math.Cos((Math.PI)/2));
 Console.WriteLine(" Smallest number = "+Math.Ceiling(9.3));
 Console.WriteLine(" Rounding value of 10.5 is = "+Math.Round(10.5));
 Console.WriteLine(" The Root Square of 36 = "+Math.Sqrt(36));
 Console.WriteLine(" The e number = " + Math.Exp(9)  );
 Console.WriteLine(" Largest Number = "+ Math.Floor(9.3));
 Console.WriteLine(" 2 Raised to the power of 4 = "+Math.Pow(2,4));
 Console.WriteLine(" Log(1000) = "+Math.Log10(1000));
  } // end method main
 } // end class
```

```
RESULT
The pi number = 3.141592653589793
 Absolute of -15 is: = 15
 Sin(90) = 1.0
 Cos(90) = 6.123233995736766E-17
 Smallest number( 9.3) = 10.0
 Rounding value of 10.5 is = 9
 The Root Square of 36 = 6.0
 The e number Exp(9) = 8103.083927575384
 Largest number( 9.3) = 9
 2 Raised to the power of 4 = 16.0
 Log(1000) = 3.0
Press any key to continue . . .
```

٥ – ٦ – التوابع بدون وسطاء

Functions with Empty Parameter Lists

في لغة #C يكتب التابع الذي لا يمتلك وسطاء إما بكتابة void بين القوسين الذين يتبعان اسم التابع أو تركهما فارغين ، فمثلاً التصريح عن التابع :

public static void print ( ) { .... }

يشير إلى أن التابع print لا يأخذ أي وسيط وهو لا يرجع قيمة ، ويبين المثال(٥-٦) الطريقة التي تكتب بهما التوابع التي لا تأخذ وسطاء . حيث التابع الأول يقوم بحساب وطباعة جداء عددين أما التابع الثاني يقوم بحساب وطباعة مجموع عددين .

مثال (٥-٦)

```
using System ;
class math1
{
   public static void  f1( ) {
   double x,y ;
    x=55;   y=44;
    Console.WriteLine( "x*y= {0}", x*y) ;
  }// end f1
  public static void  f2( ) {
     int x,y  ;
     x=5;y=7;
     Console.WriteLine("x+y= {0}",(x+y)) ;
  }// end f2
public static void Main(string [ ] args ) {
  f1 ( );
  f2 ( ) ;
  } // end method main
 } // end class
```

```
RESULT
 x * y  = 2420
 x + y = 12
Press any key to continue . . .
```

٥-٧- التحميل الزائد للتوابع

Overloading Functions

التحميل الزائد للتوابع يعني استخدام الاسم نفسه لعدة توابع، لكن كل تابع يجب أن يكون له تعريف مستقل، وعند استدعاء التابع، يبحث المترجم عن نوع وسطاء التابع وعددها لمعرفة التابع المقصود، ولكي يميز المترجم بين تابع وآخر يحمل الاسم نفسه ، يقوم بعملية تعرف بتشويه الأسماء (names mangling)، وتتألف هذه العملية من إنشاء اسم جديد خاص بالمترجم عن طريق دمج اسم التابع مع أنواع وسطائه.

مثال (٥-٧)

البرنامج التالي يقوم بتحميل التابع square بشكل زائد وذلك من أجل حساب وطباعة الجذر التربيعي لعددين من النوعين int و double على الترتيب .

البرنامج :

```
using System ;

class Overloading {
 public static int square(int x)
 {
   return  x * x  ;
 }// end funct1
 public static double square(double y)
 {
   return   y * y ;
 }// end fuct2
public static void Main(string [ ] args )
{
   Console.WriteLine( "The square of integer 7 is" + square(7) ) ;
   Console.WriteLine( "The square of double 8.5  is " + square(8.5) ) ;
```

```
  } // end method main
} // end class
```

```
RESULT
The square of integer 7 is 49
The square of double 8.5 is 72.25
Press any key to continue . . .
```

مثال (٥-٨)

البرنامج التالي يقوم بتحميل التابع abs بشكل زائد وذلك من أجل حساب وطباعة القيمة المطلقة لثلاثة أعداد من الأنواع int و double و long على الترتيب.

البرنامج :

```
// abs is overloaded three ways
using System ;
class absOverloading {
public static void Main(string [ ] args )
{
   Console.WriteLine( abs (-10) );
   Console.WriteLine( abs(-11.0) );
   Console.WriteLine( abs( -9L ) );
} // end method main
public static int abs ( int  i ) {
  Console.WriteLine("using integer abs( ) " );
  return    i< 0 ? -i : i  ;
}//End abs int
```

```
//Continued  ------------------
public static double abs ( double  d ) {
  Console.WriteLine("using double abs( ) ");
   return   d < 0.0  ? –d  : d  ;
}//End abs double
//Continued  ------------------
public static long abs( long  l ) {
   Console.WriteLine("using long abs( ) ");
   return   l<0.0 ? -l : l ;
  }//end abs Long
} // end class
```

```
RESULT
 using integer abs( )
10
 using double abs( )
11
 using long abs( )
9
Press any key to continue . . .
```

٥–٨–التمرير بالقيمة والتمرير بالعنوان

Pass by Value and Pass by Reference

لنفرض لدينا متحولين صحيحين في برنامج ونريد استدعاء تابع يقوم بتبديل قيمتي العددين ، ولنفرض أن العددين كما يلي:

```
int  x=1;
int  y=2;
```

يوجد طريقتين :

١. التمرير بالقيمة(pass-by-value):

عند تمرير وسيط بطريقة الاستدعاء بالقيمة يجري إنشاء نسخة عن قيمة الوسيط لتمرر بعدها إلى التابع المستدعى وبالتالي لا تؤثر أية تغيرات تحدث على هذه النسخة على القيمة الأصلية للمتحول ضمن التابع المنفذ لعملية الاستدعاء.

ياترى هل يقوم التابع التالي بتبديل القيمتين:

```
public static void swap (int a , int b )
{
   int temp =a ;
   a=b ;
   b=temp ;
}
```

يقوم هذا التابع بتبديل القيمتين a و b ، لكن إذا استدعينا هذا التابع كما يلي :

```
swap( x , y )  ;
```

سنجد أن قيمتي x و y لم تتغيرا ، وذلك لأن الوسطاء العادية للتابع يتم تمريرها بالقيمة وينشئ التابع متغيرات جديدة كلياً عن a و b .

٢. التمرير بالعنوان (pass-by-reference)

عند تمرير وسيط بطريقة الاستدعاء بالمرجع فإن عنوان المتحول الأصلي هو الذي يتم تمريره إلى التابع وأية تغيرات على الوسيط ستؤثر على المتحول الأصلي ، فالتمرير بالعنوان(المرجع) هو طريقة تمكن التابع ( )swap من الوصول إلى المتحولات الأصلية x و y والتعامل معها بدلاً من إنشاء متحولات (نسخ) جديدة ، لإجبار تمرير الوسيط بالعنوان **نضيف ref** إلى نوع معطيات

الوسيط في تعريف التابع وتصريح التابع . و يبين المثال (٥- ٩) كيفية كتابة التابع swap وتمرير وسطائه بالعنوان.

مثال (٥-٩)

```
using System ;
class byreference {
public static void Main(string [ ] args )
{
  int a=5 , b= 6 ;
  Console.WriteLine(" sa= {0} " , sa(a)) ;   //sa=25
  Console.WriteLine(" a= {0} " , a ) ;    // a not change , a=5
  Console.WriteLine(" sb= {0} " , sb( ref  b ) ) ; //sb=36
  Console.WriteLine(" b= {0} " , b ) ; // b is change , b=36
} // end method main
 public static int sa ( int x )
  {
        return  x = x * x  ;
  }
 public static int sb ( ref int y )
 {
        return  y = y * y  ;
  }
} // end class
```

```
RESULT
sa=25
a = 5
sb=36
b = 36
Press any key to continue . . .
```

ملاحظة :

يمكن استخدام الكلمة المفتاحية out بدلاً من ref . إن استخدام out مشابه بشكل كبير لاستخدام ref ، لكن الكلمة المفتاحية out تُعلم المترجم أن التابع الذي يتم تمرير المتحول إليه سيقوم بمهمة إعطاء قيمة ابتدائية له . لأنه عند تمرير متحول بالعنوان باستخدام الكلمة المفتاحية ref يجب إسناد قيمة ابتدائية للمتحول الذي يتم تمريره للتابع ، وسيتم توضيحه في فصل الصفوف عند إنشاء غرض من صف .

٥ – ٩ – التوابع العودية

Recursive Functions

من الميزات التي ظهرت في اللغات الحديثة تابع العودية (التكرار والتوليد) ، وتتلخص هذه الميزة بإمكانية أن يستدعي التابع نفسه ، وأن الفائدة التي يجنيه المبرمج من استدعاء التابع نفسه باستعمال تابع العودية هي أن هذا التابع يستخدم لحل بعض المسائل المرتبة ترتيباً متكرراً أصلاً كمسائل ترتيب الأرقام تصاعدياً أو تنازلياً في برنامج quicksort ، كما أن هذا النوع من التوابع يفيد في برمجة بعض مسائل الذكاء الصنعي AI(Artificial Intelligent) . وعند استخدام تابع العودية نحتاج أن نستعمل التعليمة الشرطية if لإجبار التابع على

العودة إلى مكان انطلاقه بعد استدعاء نفسه في البرنامج ، وإن لم نستعمل التعليمة if فإن التابع لن يعود .

مثال (٥-١٠)

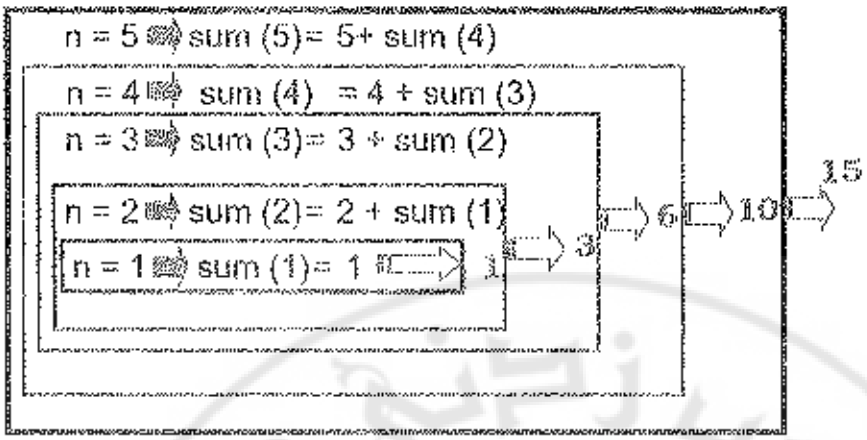
١. التابع التالي يقوم بحساب مجموع n عدد صحيح مرتب باستخدام مفهوم العودية ، لنفرض لدينا التابع التالي :

```
public static int  sum( int n ) {
   if ( n == 1 )
     return 1;
   else
    return ( n + sum( n – 1 ) )  ;
}// end function
```

هذا يعني أن Sum(n) = n + sum(n-1) ، وتتم عملية الجمع كما يلي :

- عند استدعاء هذا التابع على شكل sum(5) ، فإن الحاسوب يقوم عند الاستدعاء الأول بتنفيذ 5+sum(4) فإن التعليمة return تقوم باستدعاء التابع sum(4) مرة أخرى .
- وأما التابع sum(5) الذي استدعي في المرة الأولى فسوف يحتفظ بنسخة منه في الذاكرة على شكل تابع غير منتهي .
- وبنفس الطريقة فإن التابع sum(4) الذي استدعي في المرة الثانية سيبقى غير منتهي ، وهكذا حتى يتم استدعاء sum(1) .
- نلاحظ أن الشرط في sum(1) قد تحقق وأن sum(1) سينفذ التعليمة return (1) التي هي القيمة الذي ينتظرها التابع sum(2) والذي سينفذ التعليمة return وهكذا وحتى نصل إلى sum(5) .

يوضح الشكل(٥-٤) طريقة العودية لعملية الجمع .



الشكل(٥-٤) -- يوضح طريقة العودية لعملية الجمع

٢. ويبين التابع التالي طريقة العودية لحساب n! :

```
public static int fact( int n )
 {
   if (n == 0 || n == 1 )
     return 1 ;
    else
     return n * fact( n – 1 ) ;
}//end fact
```

مثال (٥-١١)

اكتب برنامجاً بلغة C# ، يقوم بما يلي :

- حساب n عاملي (!) باستخدام التابع العودي.
- إيجاد مجموع n عدد صحيح مرتب باستخدام مفهوم العودية .

البرنامج :

```
using System ;
class Recursion {
public static void Main(String [ ] args ) {
  int n = 5 ;
  Console.WriteLine( "Factorial "+n+" is "+fact(n) );
  Console.WriteLine("Summing "+n+" is"+sum(n) );
} // end method main
public static int  fact( int  n) {
  if (n == 0 || n == 1 )
     return 1;
  else
     return  n * fact(n-1);
}//end fact
public static int  sum( int  n) {
  if (n == 1)
     return 1;
  else
     return  n + sum(n-1);
}//end sum
} // end class
```

```
RESULT
 Factorial  5  is  120
 Summing  5  is  15
Press any key to continue . . .
```

مثال (٥-١٢)

اكتب برنامجاً بلغة #C ، يقوم بما يلي :

- حساب n عاملي (!) باستخدام التابع العودي.
- إيجاد مجموع n عدد صحيح مرتب باستخدام مفهوم العودية .

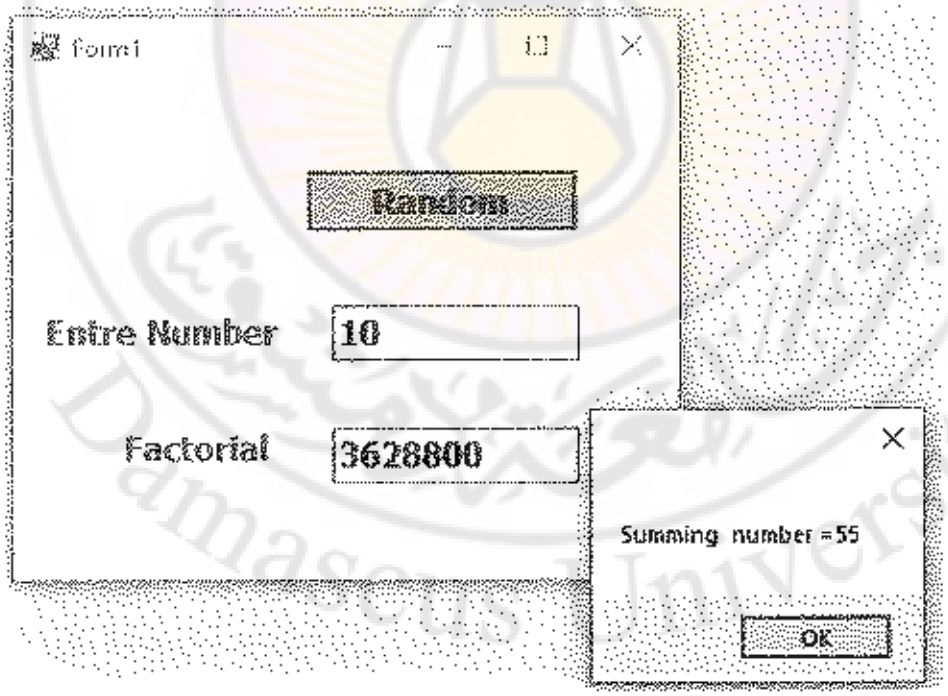
استخدم الواجهات في حساب n! ومجموع n(n-1) .

سوف نستخدم في حساب العاملي والمجاميع حلقات for بدلاً من تابع العودية .

البرنامج :

```
using System.Windows.Forms;
namespace Factorial {
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
    private void Factbutton1_Click(object sender, EventArgs e)
    {
      long fact=1, sum = 0;
      int x = int.Parse(RandtextBox1.Text);
      for( int i=1 ;i<= x ; i++ ) {
        fact = fact * i;
        sum = sum + i ;
      }
      RandtextBox2.Text = fact.ToString();
      MessageBox.Show(" Summing  number =" + sum.ToString());
    }// end Randbutton1
    private void Randlabel1_Click(object sender, EventArgs e)
    {

    }
```

```
private void FacttextBox1_TextChanged(object sender, EventArgs e)
  {

  }

 private void FacttextBox2_TextChanged(object sender, EventArgs e)
  {

  }

private void Factlabel2_Click(object sender, EventArgs e)
  {

  }
  }// end class  Form1
}//end namespace
```



٥-١٠- توليد الأرقام العشوائية

Random Number Generation

جميع لغات البرمجة توفر توابع أو طرائق أو صفوف تقوم على إنشاء أرقام عشوائية Random وقد تُستخدم هذه الارقام العشوائية في كثير من الأمور وأهمها ما يتعلق بالألعاب فمثلاً في العاب الأوراق (الشدة) كل مرة تأتي الأوراق مختلفة أي عشوائية Random، ولعبة النرد حيث يكون لحجر النرد ستة وجوه وكثير من الامور تحتاج إلى أرقام عشوائية .

لغة #C وفرت صف يقوم على إنشاء الأرقام العشوائية يسمى Random، التابع المسؤول عن إنشاء الأرقام العشوائية يسمى Next() وهو غير معرف على أنه static لذلك يتوجب علينا أولاً إنشاء غرض من الصف Random ثم الوصول إليه عن طريق الغرض. التابع Next() هو يقبل التحميل الزائد Overloading إذا قمنا باستدعائه بدون إرسال أي متحول يرجع لنا رقماً عشوائياً Random Number من 0 حتى آخر رقم موجود في لغة #C هو Max Number أي (0 إلى 2,147,483,647) .

يتم إنشاء غرض من الصف Random كما يلي :

```
Random rnd = new Random()
```

- التابع Next() هو المسؤول عن توليد الأرقام العشوائية ، وله وسيطان : **الوسيط الأول** هو الحد الأدنى وهو يمثل القيمة الابتدائية **والوسيط الثاني** هو الحد الأعلى وهو يمثل القيمة العظمى أي

```
Next( min , Max + 1 ) ;
```

- مثلاً لو أردنا توليد عدد عشوائي بين 1 و 12 :

```
rnd.Next( 1 , 13 ) ;
```

حيث يولد التابع رقماً عشوائياً بين 1 و 12 .

- التابع Next() يعيد قيمة ، إن استدعائه بهذه الطريقة خطأ ، يجب اسناده إلى متغير من نوع القيمة المعادة ، أي :

```
int   rNum  ;
rNum =  rnd.Next( 1 , 13 )  ;
```

يبين الجدول (٥-٢) التالي بعض الأمثلة :

Code	Expected output
Random x = **new** Random( ) ; Console.WriteLine(x.Next( )) ;	0 – 2,147,483,647
Random x = **new** Random( ) ; Console.WriteLine(x.Next(5)) ;	0 – 4
Random x = **new** Random(); Console.WriteLine(x.Next(1 , 5)) ;	1 – 4

الجدول (٥-٢) – بعض الأمثلة على توليد الأرقام العشوائية

مثال (٥-١٣)

المثال التالي يوضح :

- ✓ توليد رقم عشوائي من 1 حتى 7 يدل على أيام الأسبوع .
- ✓ توليد رقم عشوائي من 1 حتى 12 يدل على اسم الشهر .
- ✓ بينما العام يبقى ثابت على 2017 .

البرنامج :

```
using System;
```

```
class RandIntegers {
public static void Main(string[ ] args)
{          //    random number generator

      Random rnd = new Random();
      Random rnd1 = new Random();
      int rNum;
       int rNum1;
      rNum = rnd.Next(1, 8);
      rNum1 = rnd1.Next(1, 13);
  DateTime dt = Convert.ToDateTime(rNum + "/" + rNum1 + "/ 2017");
  Console.WriteLine(dt); // display generated value
  Console.WriteLine(dt.ToString("MMMM")); // display generated value
switch ( rNum  ) {
 case 1:
    Console.WriteLine("Sunday"); break;
case 2:
  Console.WriteLine("Monday");
  break;
case 3:
  Console.WriteLine("Tuesday");
  break;
case 4:
  Console.WriteLine("Wednesday");
  break;
case 5:
  Console.WriteLine("Thursday");
  break;
case 6:
  Console.WriteLine("Friday");
  break;
case 7:
  Console.WriteLine("Saturday");
  break;
```

```
default:
  Console.WriteLine("Sorry, we are out");
  break;
      }//end switch
   } // end method main
} // end class Name
```

```
RESULT
  04/06/2017 00:00:00
  June
  Wednesday
  ----------------
  06/10/2017 00:00:00
  October
  Friday
Press any key to continue . . .
```

بالنسبة لبرنامج يحاكي عملية رمي حجر النرد ذي ستة الوجوه ، فإننا نحتاج إلى ست قيم ضمن المجال من 1 حتى 6 ، وحتى نبين كيفية عمل الصف Random ، لنقم بكتابة برنامج يحاكي عملية رمي نرد ذي ستة الوجوه 20 مرة حيث يتم طباعة نتيجة كل عملية من عمليات الرمي .

مثال (٥-١٤)

اكتب برنامجاً بلغة الـ C# يقوم بمحاكاة عملية رمي حجر النرد ذي الأوجه الستة، 20 مرة وبحيث يطبع نتيجة كل عملية من عمليات الرمي.

البرنامج :

```
// Shifted, scaled integers produced by 1 + rand( ) % 6
using System ;
class RandIntegers {
public static void Main( string [ ] args )
  {                    // random number generator
    Random  r = new  Random( ) ;
    int face; // stores each random integer generated loop 20 times
   for ( int counter = 1; counter <= 20; counter++ )
   {  // pick random integer from 1 to 6
    face = r.Next (1, 7 );
     Console.Write ("   {0} ",  face ); // display generated value

  // if counter is divisible by 5, start a new line of output
        if ( counter % 5 == 0 )
        Console.WriteLine( );
      } // end for
    } // end method main
} // end class
```

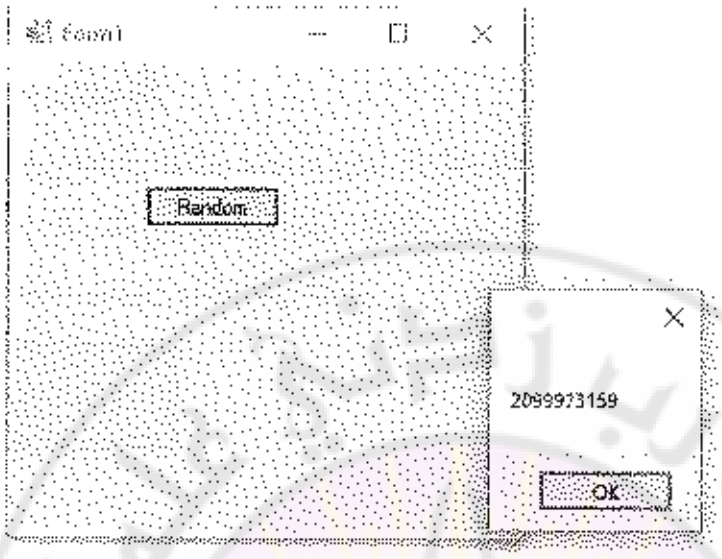
```
RESULT
   6  6  5  5  6
   5  1  1  5  3
   6  6  2  4  2
   6  2  3  4  1

Press any key to continue . . .
```

مثال (٥-١٥)

البرنامج (a) في المثال التالي يوضح توليد الأرقام العشوائية باستخدام تطبيقات ويندوز ، بينما البرنامج (b) في المثال التالي يوضح توليد الأرقام العشوائية باستخدام تطبيقات ويندوز وتدل على الشهر .

البرنامج (a)

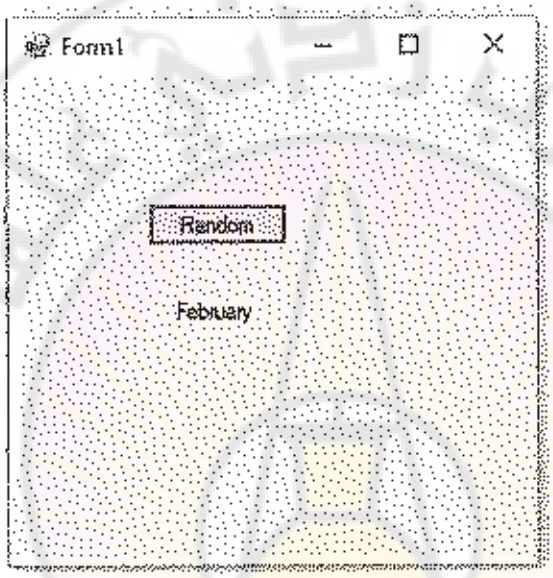
```
using System;
using System.Windows.Forms;
namespace  WindowsFormsApplication2 {
   public partial class Form1 : Form {

      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
          Random rnd = new Random();
          int s;
          s = rnd.Next();
          MessageBox.Show( s.ToString());
      }//end button
   }//end form1
}//end namespace
```

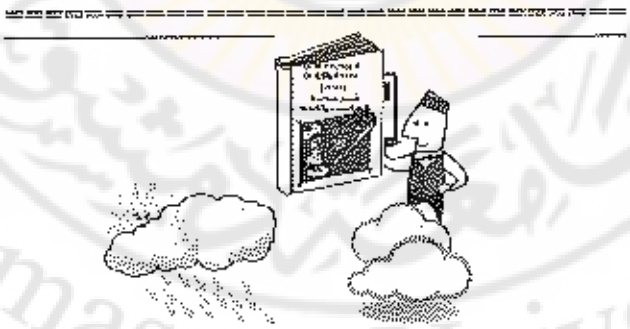


البرنامج (b)

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2 {
  public partial class Form1 : Form {
   public Form1()
    {
     InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
   {
       Random rnd = new Random();
      int s;
      s = rnd.Next(1,13);
     DateTime dt = Convert.ToDateTime(s + "/" +  s + "/ 2017");
    label1.Text = dt.ToString("MMMM");
 }//end button
```

```
private void label1_Click(object sender, EventArgs e)
        {
        }

    }//end form1
}//end namespace
```





مسائل عامة

١- اكتب برنامجاً بلغة #C يقوم بحساب العاملي (!) للأعداد من 1 إلى 5.

==========

٢- اكتب برنامجاً بلغة #C يحتوي على تابعين : التابع الأول اسمه Factorial مهمته حساب العاملي (!) لأي عدد صحيح موجب ، أما الثاني اسمه SumFactorial مهمته حساب المقدار التالي :

$$sum = n! + \frac{n!}{x!} - m!$$

التابع Main يستدعي كلاً من التابعين وطباعة قيمة المقدار Sum (التابع SumFact يستدعي التابع Factorial) .

==========

٣- اكتب برنامجاً بلغة #C يقوم بما يلي : إدخال قيم المتحولات a,b,x وهي من النمط double ويحتوي البرنامج على تابعين :

- التابع الأول اسمه FunctionY مهمته حساب المقدار التالي :

$$y = x^2 + 2x + 10$$

- أما التابع الثاني اسمه FunctionZ مهمته حساب المقدار Z والمعرف كما يلي :

$$Z = (y-a)^2 + (y+b)^2$$

- التابع main يستدعي كلاً من التابعين لطباعة قيمة Z (التابع FunctionZ يستدعي التابع FunctionY) .

٤- اكتب برنامجاً بلغة C# يقوم بإدخال أي عدد حقيقي x من قبل المستخدم . ويحتوي على ثلاثة توابع لحساب قيمة التابع الرياضي التالي :

$$yy = \begin{cases} 5x^5 + 2x + 5 \ldots\ldots if \ldots 1 \le x < 5 \\ \dfrac{x^6}{2x+1} \ldots\ldots\ldots\ldots if \ldots 5 \le x < 100 \\ 3x + 5 \ldots\ldots\ldots\ldots otherwise \end{cases}$$

يستدعي التابع Main كلاً من التوابع الثلاثة لطباعة قيمة yy .

==========

٥- اكتب برنامجاً بلغة C# يقوم بما يلي:

- إدخال أي عدد صحيح x.
- يستخدم تابعاً اسمه fun_prime لإيجاد قواسم العدد المدخل x .
- يختبر البرنامج إذا كان العدد المدخل x أولياً أو غير أولياً وذلك بناءً على خرج التابع fun_prime, وطباعة النتيجة في التابع Main .

==========

٦-اكتب برنامجاً بلغة C# يقوم بما يلي:

- إدخال أي عددين صحيحين p و k .
- يستخدم تابعين :
  - التابع الأول اسمه Factorial لحساب قيمة p! (العاملي)
  - التابع الثاني اسمه F_permutation لحساب قيمة المقدارالرياضي التالي:

$$permutation = \frac{p!}{(p-k)!}$$

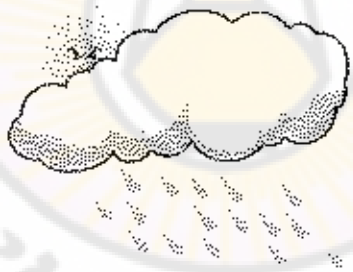
❖ التابع main يستدعي كلاً من التابعين لطباعة قيمة Permutation وطباعة النتيجة في التابع Main.

==========

٧- أنشئ تابعاً بلغة #C اسمه sumpow يقوم بحساب مجموع السلسلة $z = \sum_{i=1}^{r-10} i^r$. حيث r=5 ثم اكتب برنامجاً يقوم باستدعاء هذا التابع وطباعة النتيجة .

==========

٨- أنشئ تابعاً بلغة #C اسمه Cube يقوم بحساب حجم مكعب طول ضلعه x .ثم اكتب برنامجاً يقوم باستدعاء هذا التابع وطباعة النتيجة .



الفصل السادس

المصفوفات

Arrays

المصفوفات

٦-١- مقدمة

تنقسم المعطيات إلى معطيات محرفية (char) و صحيحة(int) و حقيقية (float) ، وتسمى هذه الأنواع بالأنواع الرئيسية للمعطيات، ولكن هناك أنواع أخرى من المعطيات تسمى بالأنواع المشتقة Derived Data Types ، منها المصفوفات Arrays ، ومن أهم فوائد استخدام المصفوفات :

- يمكن استخدام المصفوفات من أجل الترتيب التصاعدي والتنازلي للعناصر والقيم المختلفة (القيم العددية).
- يمكن استخدام المصفوفات في ترتيب الأسماء وفق الترتيب الأبجدي في النصوص الرمزية.
- يمكن استخدام المصفوفات في العمليات الحسابية كضرب المصفوفات. و في إيجاد مقلوب مصفوفة.

٦-٢- تعريف المصفوفة

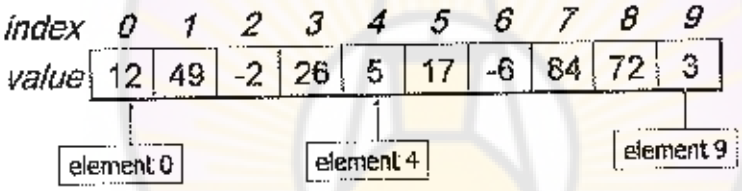
Array Definition

المصفوفة هي بنية تُستخدم للاحتفاظ بمجموعة من القيم من نفس نمط المعطيات والمصفوفة هي نوع من أنواع بنى المعطيات، لها عدد محدود ومرتب من العناصر التي تكون جميعها من النوع type نفسه، فمثلاً يمكن أن تكون جميعها صحيحة int أو حقيقية float أو محرفية char ، ولكن لا يمكن الجمع بين نوعين مختلفين في المصفوفة نفسها، وتنقسم المصفوفات إلى نوعين : مصفوفات أحادية البعد ومصفوفات متعددة الأبعاد (على سبيل المثال مصفوفات ثنائية الأبعاد).

٦- ٣- المصفوفات أحادية البعد

One-Dimensional Arrays

المصفوفة التي تتكون من سطر واحد أو عمود واحد ، تسمى مصفوفة أحادية البعد، ويمكن أن تكون المصفوفة أحادية البعد محرفية char أو حقيقية float أو صحيحة int ، أما دليل المصفوفة (index) ، فيجب أن يكون من النوع الصحيح int ، وكل مصفوفة تُستخدم في البرنامج ، لا بد من التصريح عنها, قبل استخدامها . ويبين الشكل (٤-١) مصفوفة a تحتوى على 10 عناصر من النوع int، ويمكن الوصول إلي أي من هذه العناصر بذكر اسم المصفوفة متبوعاً برقم موقع العنصر في المصفوفة محاطاً بالأقواس [ ] ، أي a[10] ، اسمها a عدد عناصره 10 .



الشكل (٤-١) - يبين مصفوفة أحادية البعد a[10] من النوع int

حيث يرمز لرقم العنصر في المصفوفة بدليل العنصر index ، ودليل العنصر الأول في المصفوفة هو 0 ولهذا يشار إلى العنصر الأول في المصفوفة a بـ a[0] والثاني a[1] والسابع a[6] وعموماً يحمل العنصر i في المصفوفة a الدليل a[i-1] ، وتتبع تسمية المصفوفات قواعد تسمية المتحولات ذاتها.

٦-٣-١- التصريح عن المصفوفة أحادية البعد

تحتل المصفوفات حجماً في الذاكرة لذا يجب على المبرمج تحديد نوع عناصر المصفوفة وعددها حتى يتسنى للمعرف تخصيص الحجم اللازم من الذاكرة لحفظ المصفوفة، وحتى تخبر المترجم بأن يخصص حجماً محدداً لنوع عناصر المصفوفة، ويتضمن التصريح عن مصفوفة أحادية البعد مرحلتين :

١- التصريح عن متحول يشير إلى مصفوفة

إن التصريح عن مصفوفة يشبه التصريح عن أي متحول آخر ، فالتصريح يحوي قسمين :

- الأول هو نوع المصفوفة
- والثاني هو اسم المصفوفة .

والشكل العام للتصريح عن المصفوفة أحادية البعد هو التالي:

```
Type [ ] anArray ;
```

حيث Type هو نمط المعطيات ، وأما القوسين [ ] فيشيران إلى أن هذا المتحول هو مصفوفة وليس قيمة واحدة .

<u>مثال :</u> String [ ] anArray1 , int [ ] anArray2 , . . .

إن التصريح عن متحول مصفوفة لن يحجز أي منطقة في الذاكرة حتى يتم فعلياً إسناد قيم إلى المصفوفة ، وإنما يتم فقطا صنع مكان تخزين من أجل المرجع على المصفوفة ، وإذا لم يملك المتحول مرجعاً إلى المصفوفة فإن قيمة هذا المتحول هي null .

٢- إنشاء المصفوفة

يتم إنشاء مصفوفة باستخدام المعامل new ، مع الصيغة التالية :

```
anArray = new  dataType [ size ] ;
```

تقوم العبارة السابقة بمهمتين : الأولى إنشاء مصفوفة باستخدام **new** dataType [size] ، والثانية إسناد مرجع للمصفوفة المنشئة حديثاً إلى المتحول anArray .

<u>مثلاً :</u>

```
anArray1 = new int [20] ;
```

- هذه العبارة تقوم بإنشاء مصفوفة وتخصيص ذاكرة محددة لها تتسع لعشرين عدداً صحيحاً ، ثم تسند مرجعاً لهذه المصفوفة إلى المتحول anArray1 .

```
anArray2 = new double [20] ;
```

- إنشاء مصفوفة وتخصيص ذاكرة محددة تتسع لعشرين عدداً حقيقيا ، ثم تسند مرجعاً لهذه المصفوفة إلى المتحول anArray2 .

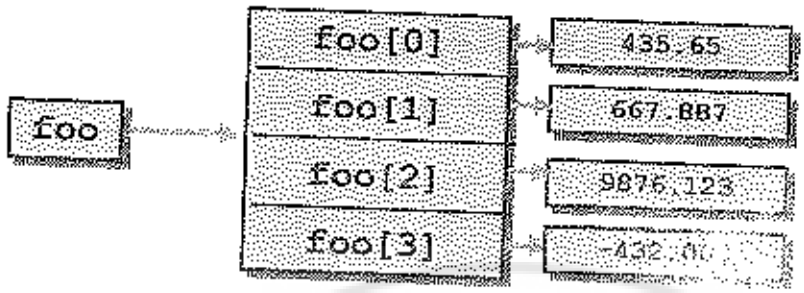
```
anArray3 = new String [20] ;
```

- إنشاء مصفوفة وتخصيص ذاكرة محددة تتسع لعشرين سلسلة محرفية ، ثم تسند مرجعاً لهذه المصفوفة إلى المتحول anArray3 .

<u>مثلاً :</u>

التصريح عن مصفوفة foo من نمط double تتكون من 4 عناصر (إنشاء مصفوفة) .

```
double foo[ ] ;
foo = new double [4] ;
```



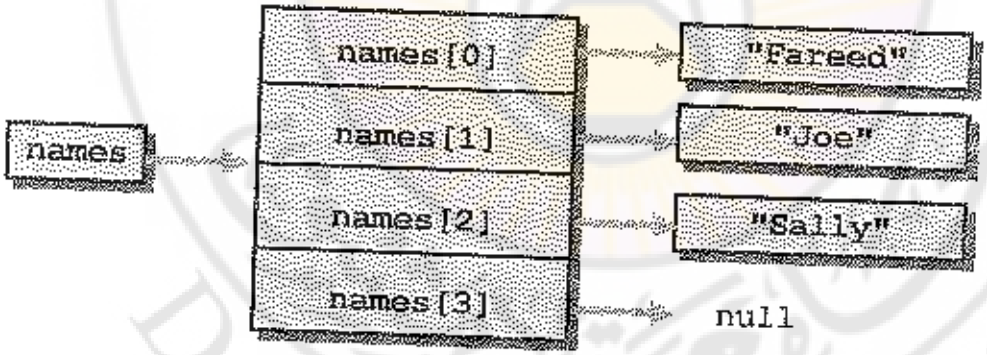
من الممكن دمج عملية التصريح عن متحول المصفوفة وإنشاء المصفوفة وإسناد مرجع للمصفوفة إلى المتحول في عبارة واحدة كما يلي :

dataType [ ] anArray = **new** dataType [size] ;

<u>مثلاً :</u>

التصريح عن مصفوفة names من نمط String تتكون من 4 عناصر (إنشاء مصفوفة) .

String [ ] names = **new String** [4];



٣- تهيئة ومعالجة المصفوفات

عند إنشاء المصفوفة يتم إسناد القيمة الافتراضية (0) من أجل أنماط المعطيات الأولية العددية والقيمة الافتراضية '\u0000' من أجل النمط char ،

ويتم إسناد القيمة الافتراضية false إلى الأنماط المنطقية . للحصول على حجم المصفوفة (عدد عناصرها) نستخدم الصيغة التالية :

Arrayname.length

تعيد العبارة Arrayname.length حجم المصفوفة . ويتم الوصول إلى عناصر المصفوفة من خلال الدليل(index) حيث يتم فهرسة المصفوفة من القيمة 0 إلى القيمة Arrayname.length .

٦-٣-٢- إسناد قيم ابتدائية لعناصر المصفوفة أحادية البعد

يمكن إعطاء قيم ابتدائية لعناصر المصفوفة أحادية البعد باتباع التصريح عن المصفوفة بعلامة المساواة (=) تليها لائحة من القيم المطلوب إسنادها لعناصر المصفوفة عندها، ويتم الفصل بين القيم بفواصل، وتحيط هذه اللائحة الأقواس الحاصرة { } ، على سبيل المثال لنأخذ التصريحات التالية لثلاث مصفوفات من أنواع مختلفة من المعطيات ، وفي الوقت نفسه إعطاء قيم ابتدائية لعناصر المصفوفة (إجراء عملية تمهيد للعناصر) عند التصريح عنها ، كما يلي:

```
1:int [ ] arry1 = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
2:Sring [ ] arry2 = { "Ahmad", "Omar", "Ali"};
3:char [ ] arry3 = { 'A', 'B', 'C', 'D', 'E'};
4:int [ ]n = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
5:int [ ]n =int[10] { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37}
```

أو

مثال (٦-١)

يقوم البرنامج التالي بإعطاء قيم ابتدائية لعناصر المصفوفة arry1 من النوع int والتي تمثل خمس درجات للطلاب في البرمجة(1) ، والمصفوفة الثانية arry2 من النوع String وتمثل أسماء الطلاب ، أما المصفوفة الثالثة arry3 من النوع char وتمثل الرمز المكافئ للدرجة .

```
using System;
namespace ConsoleApplication1 {
 class Program {
  static void Main(string[ ] args) {
        // initialize array
  int [ ]arry1 = {80,66,88,90,95} ;
  String [ ]arry2 = {"Ahmad", "Omar", "Younes","Waseem","Akram"} ;
  char [ ]arry3 = { 'B', 'D', 'B', 'A', 'A'} ;
  for (int i = 0; i < 5; i++)   // print array
 Console.WriteLine(i +1 +"  " +arry1[i] + "  " +arry2[i] + "  " + arry3[i]);
     }// end main
   }// end class
}// end namespace
```

```
RESULT
   1  80  Ahmad   B
   2  66  Omar    D
   3  88  Younes  B
   4  90  Waseem  A
   5  95  Akram   A
Press any key to continue . . .
```

لنعيد كتابة البرنامج باستخدام الواجهات .

```
using System;

using System.Windows.Forms;

namespace Www1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void Form1_Load(object sender, EventArgs e)
      {
         // initialize array
  int [ ]arry1 = {80,66,88,90,95} ;
  String [ ]arry2 = {"Ahmad", "Omar", "Younes","Waseem","Akram"} ;

  char[ ] arry3 = { 'B', 'D', 'B', 'A', 'A' };
 String output="\n" ;
 for (int i=0 ; i< 5; i++){    // print array
  int j=i+1;
output = output +j+"  " +arry1[i]+"  " +arry2[i]+"  "+arry3[i]+"\n" ;
}//end for
// Display array !
 MessageBox.Show("البرمجة مقرر في الطلاب نتيجة " + output, "My Arry
Example ",
       MessageBoxButtons.OKCancel, MessageBoxIcon.Information );
      }//end main
   }// end class
}//end namespace
```

فإذا كان المطلوب إسناد عناصر مصفوفة مهما كان حجمها بأصفار ، يمكن التصريح عن المصفوفة بالشكل التالي :

```
int [ ]anyarray ={0};
```

سيتم إسناد القيمة 0 للعنصر الأول، أما العناصر المتبقية يسند لها القيمة صفر أيضاً كوننا لم نحدد قيماً لها. إذا لم يتم تحديد عناصر المصفوفة من قبل المبرمج عند التصريح عنها، فإن المترجم يقوم بهذه المهمة من خلال عد القيم (العناصر)، الموجودة في الطرف الأيمن وحجز أماكن لعناصر المصفوفة وفق هذا العدد الناتج فعلى سبيل المثال :

```
int [ ]array ={10,20,30};
```

هذا التصريح يعني: أن عدد عناصر المصفوفة array[ ] هو ثلاثة فقط .

٦-٣-٣- إدخال وإخراج عناصر المصفوفة أحادية البعد

بالإضافة إلى إعطاء قيم ابتدائية لعناصر المصفوفة عند التصريح عنها، يمكن استخدام أمر الإدخال Console.ReadLine() ضمن حلقة التكرار ، ومن أجل طباعة عناصر المصفوفة يُستخدم أمر الإخراج وذلك أيضاً ضمن

حلقة تكرار ، عادة قبل استخدام المصفوفة يمكن تصفيرها ، وذلك بإعطاء قيم ابتدائية تساوي الصفر ، ويوضح المثال(٦ -٢) كيفية إدخال وإخراج عناصر المصفوفة احادية البعد.

مثال (٦-٢)

```
using System ;
 namespace   Example {
class IOArry {
public static void Main(string [ ] args ) {
int   [ ]n ;
n – new int [10] ;
  for (int i=0; i<10;i++) // initialize array
        n[i] = 0;
for (int i=0 ; i< 10; i++)   // print array
Console.WriteLine( i+" "+n[i]) ;
for (int i=0 ; i< n.Length; i++ )
//Enter new elements
n[i] = int.Parse( Console.ReadLine() );
for (int i=0 ; i< n.Length; i++)
Console.WriteLine( i+" " +n[i]) ;// print array

} // end method main
} // end class
 } //end namespace
```

مثال (٦-٢)

اكتب برنامجاً بلغة C# يقوم بما يلي :

- تصفير عناصر المصفوفة myArray [5] .
- إدخال 5 أعداد صحيحة وتخزينها في المصفوفة.
- طباعة عناصر المصفوفة المُدخلة.

البرنامج :

```
sing System;

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
  int[ ] myArray = new int[5];
 for (int i=0; i< myArray.Length ;i++)   // initialize array
      myArray [i] = 0;
for (int i=0 ; i< myArray.Length ; i++)   //Enter new elements
 myArray [i] =  int.Parse( Console.ReadLine() );

 for (int i=0 ; i< myArray.Length ; i++)      // print array
Console.WriteLine("myArray [" +i+"]="+ myArray [i]);

      }// end main
   }// end class
}// end namespace
```

```
RESULT
  Enter The value of myArray
 55 67 88 23 95
 myArray[0]=55
  myArray[1]=67
  myArray[2]=88
  myArray[3]=23
The value of myArray[4]=95
Press any key to continue . . .
```

ملاحظات مهمة :

١. إذا كانت القيم الابتدائية الموجودة في اللائحة أكثر من حجم المصفوفة المحددة فإن المترجم سيولد خطأ، وإذا كانت أقل سيملأ المترجم بقية العناصر أصفاراً.

٢. إذا كان المطلوب إسناد عناصر مصفوفة مهما كان حجمها بأصفار ، يمكن التصريح عن المصفوفة بالشكل التالي :

```
int  [10]anyarray = { 0 } ;
```

٣. سيتم إسناد القيمة 0 للعنصر الأول أما العناصر المتبقية يسند لها القيمة صفر أيضاً كوننا لم نحدد قيمة لها.

يقوم البرنامج (٦-٣) التالي بجمع 12 عنصراً في مصفوفة من النوع int .

مثال (٦-٣)

```
// compute the sum of the elements of the array
using System ;
class ArrayProg {
```

```
public static void Main(string [ ] args )
{
   const  int size =12;
  int [ ] a = new int [size ] {1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45};
  int total = 0;
 for (int i= 0; i< a.Length ; i++)
     total = total+ a[i];
 Console.WriteLine(" total of array element values is "+  total );
} // end method main
} // end class
```

```
RESULT
   total of array element values is   383
Press any key to continue . . .
```

❑ نلاحظ في العبارة:

```
const  int  size = 5 ;
int [ ] myarray = new int [ size ]
```

استعملنا الكلمة المفتاحية const ، يتم استعمالها في تعريف المتحول الذي لا يمكن تغيير قيمته(الثابت) في البرنامج ولذلك يجب إعطاؤه قيمة ابتدائية عند تعريفه (في البرنامج السابق تم تحديده بالقيمة 12) .

يبين البرنامج (٦-٤) التالي إدخال عناصر المصفوفة وطباعة عدد أيام الشهر (28 أو 30 أو 31) وفي مثالنا الشهر رقم (4) هو الشهر April ويقابله المصفوفة month_days[3] .

مثال (٦-٤)

```
// Demonstrate a one-dimensional array.
using System ;
 namespace   Example51 {
class Array {
public static void Main(string  [ ] args) {
int [ ]month_days;
//int [ ] month_days =int[12] { 31, 28, 31, 30, 31, 30, 31,   31, 30, 31
                                        , 30, 31 }  ;
month_days = new int[12];
month_days[0] = 31;
month_days[1] = 28;
month_days[2] = 31;
month_days[3] = 30;
month_days[4] = 31;
month_days[5] = 30;
month_days[6] = 31;
month_days[7] = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[11] = 31;
Console.WriteLine("April has " + month_days[3] + " days.");

} // end method main
} // end class
 } //end namespace
```

```
RESULT
  April has 30 days.
Press any key to continue . . .
```

البرنامج (٦-٥) التالي يقوم بحساب المتوسط الحسابي لعناصر المصفوفة المكونة من خمسة عناصر

مثال (٦-٥)

```
// Average an array of values.
using System ;
 namespace   Example52 {
class Average {
public static void Main(string [ ] args)
{
  double [ ] nums = {10.1, 11.2, 12.3, 13.4, 14.5};
  double result = 0;
  for(int i=0; i<nums.Length; i++)
  result = result + nums[i];
  Console.WriteLine("Average is " + result / 5);
} // end method main
} // end class
 } //end namespace
```

```
RESULT
  Average is 12.2999999
Press any key to continue . . .
```

٦-٣-٤-تمرير المصفوفات أحادية البعد كوسطاء للتوابع

عند تمرير مصفوفة أحادية البعد كوسيط لتابع عند استدعائه في التابع Main( ) يجب استخدام اسمها فقط بدون أقواس ، فعلى سبيل المثال ، إذا تم التصريح عن المصفوفة array على الشكل التالي :

```
int [ ] array = new int[ 20 ] ;
```

فإن الاستدعاء التالي للتابع sumArray( ) كما يلي :

```
sumArray(array , 20 ) ;
```

يأخذ التابع اسم المصفوفة وحجمها كوسيطين له ، وعند تمرير مصفوفة إلى تابع فإنه يتم تمرير حجمها أيضاً حتى يتعامل مع العدد المحدد من عناصر المصفوفة الممررة . إن استقبال المصفوفة المرسلة عند استدعاء التابع يجب أن تحتوي قائمة وسطاء التابع عند تعريفه على اسم المصفوفة مع الأقواس[ ] array ، ويجب كتابة الجزء الرأسي للتابع sumArray( ) على الشكل التالي :

```
public static void  sumArray( int  array [ ] , int  size  )
{
    Body
}
```

للدلالة على أن التابع sumArray( ) يتوقع استقبال مصفوفة من الأعداد الصحيحة حسب الوسيط a وهي تحتوي على size عنصراً . ليس من الضروري وضع حجم المصفوفة بين القوسين عند كتابة الوسيط array[20] ، وفي حال القيام بذلك فإن المترجم يقوم بتجاهله ، وعلى اعتبار أن المصفوفات يتم تمريرها بالعنوان فإن التابع المُستدعى يقوم باستخدام اسم المصفوفة array من أجل الرجوع إلى المصفوفة الأصلية الموجودة لدى التابع الذي تم الاستدعاء ضمنه

المصفوفة ، ويوضح المثال(٦ – ٦) التالي تمرير المصفوفة للتابع **sumArray** ليقوم بحساب مجموع عناصر المصفوفة وطباعته .

مثال (٦-٦)

```
// compute the sum of the elements of the array
using System ;
 namespace   Example {
class PassArray {
public static void Main(string [ ] args )
{
  int [ ] arry  = new int [10] ;
  for ( int i=0 ; i< arry.Length ; i++ ) {
  Console.WriteLine(" Enter Array  ");
  int x = int.Parse( Console.ReadLine() );
  arry[i]=x;
 } // end for
  sumArray ( arry , 10 )  ;
 } // end method main
public static void  sumArray ( int [ ] a , int  n) {
   int total = 0;
   for (int i= 0; i< n ; i++)
        total = total+ a[i];
   Console.WriteLine(" total of array element values is "+ total ) ;
 }//end function
 } // end class
} //end namespace
```

```
RESULT
  11 22 33 44 55 66 77 88 99 100
  total of array element values is 595
Press any key to continue . . .
```

يوجد العديد من الحالات التي لا نرغب فيها بتغيير قيم عناصر المصفوفة ، وعلى اعتبار أن عملية تمرير مصفوفة تتم بالعنوان فإنه من الصعب التحكم بعملية التغيير . لكن يتوفر في *C#* التعليمة const لمنع القيام بأي تغيير على قيم عناصر المصفوفة الممررة عند استخدام التعليمة const من قبل أي مصفوفة معطاة كوسيط فإن عناصرها تصبح عبارة عن قيم ثابتة ضمن جسم التابع ، ولا يمكن تغيير قيمها وأي محاولة لذلك ستؤدي إلى حدوث خطأ أثناء ترجمة البرنامج، ما يساعد المبرمج على تصحيح برنامجه بشكل لا يسمح بالقيام بعمليات التغيير غير المطلوبة ، نعدل جزء من البرنامج في المثال(٦-٦) باستخدام التعليمة const .

```
// compute the sum of the elements of the array
using System ;
 namespace   Example {
class PassArray {
public static void Main(string [ ] args )
{
  const  int size = 10;
  int [ ] arry  = new int [z] ;
  for ( int i=0 ; i< arry.Length ; i++ ) {
  Console.WriteLine(" Enter Array  ");
  int x = int.Parse( Console.ReadLine() );
  arry[i]=x;
```

```
  } // end for
  sumArray ( arry , 10 ) ;
 } // end method main
public static void  sumArray ( int [ ] a , int  n) {
   int total = 0;
   for (int i= 0; i< n ; i i+)
       total = total + a[i];
   Console.WriteLine(" total of array element values is "+ total ) ;
 }//end function
 } // end class
} //end namespace
```

٦-٣-٥-الحلقة foreach

تُقدم لغة #C نموذجاً خاصاً من حلقة for هي الحلقة foreach تُتيح لنا الوصول إلى جميع عناصر المصفوفة . حيث تقوم حلقة foreach بالتكرار عبر عناصر المصفوفة بدون استخدام عداد ، بحيث لا يمكن إضافة أو إزالة العناصر ضمن الحلقة، أي أن حالة المصفوفة يجب أن تبقى كما هي بدون تغيير أثناء الإعادة والتكرار عبر عناصرها .

وتتميز الحلقة foreach بما يلي :

- أن الحلقة foreach لا تحتاج لعداد.
- الحلقة foreach ليس لديها شرط للبداية .
- الحلقة foreach لا تتوقف حتى تعرض كل العناصر.
- الحلقة foreach لا تحتاج أن نعطيها كم عدد العناصر التي ستتعامل معها.

على سبيل المثال لدينا المصفوفة : string [ ]weekDays=new string[4]

نسند للمصفوفة قيم ابتدائية كما يلي :

```
string[ ] weekDays =new string[4] { "Sunday", "Monday",
"Tuesday" };

foreach (string day in weekDays)
 {
    System.Console.WriteLine("The day is : {0} ",day);
 }
```

تم إنشاء متغير اسمه day من نوع string (لأن المصفوفة weekDays تحتوي على أسماء الأيام أي هي من نوع string) ، تعني in أي في المصفوفة weekDays . هذا معناه من أجل كل عنصر نصي Day من المصفوفة weekDays قم بالعملية التالية :

```
System.Console.WriteLine("The day is : {0} ",day);
```

البرنامج في المثال(٦-٧)التالي يوضح كيفية طباعة عناصر المصفوفة بواسطة حلقة foreach .

مثال (٦-٧)

```
using System;
class ArrayProg {
 public static void Main(string[] args)
 {
   string[ ] weekDays = new string[7] { "Sunday", "Monday",
   "Tuesday","Wednesday", "Thursday", "Friday", "Saturday"  } ;
```

```
    foreach (string day in weekDays)
    {
        System.Console.WriteLine("The day is : {0} ", day);
    }  // end foreach
  } // end method main
} // end class
```

```
RESULT
  The day is : Sunday
  The day is : Monday
  The day is : Tuesday
  The day is : Wednesday
  The day is : Thursday
  The day is : Friday
  The day is : Saturdays
Press any key to continue . . .
```

٦-٣-٦-طرائق (توابع) الصف Array

يمتلك الصف Array عدداً من التوابع الستاتيكية (الساكنة) التي يمكن استخدامها في عمليات البحث Binarysearch والفرز Sort والوصول إلى العناصر وإنشاء نسخ فعالة من المصفوفة . يبين الجدول(٦-١) التالي بعض التوابع للصف Array . بينما يبين البرنامج(٦-٨) التالي استخدام الحلقة foreach واستخدام توابع الصف Array ، كالتابع Sort والتابع Reverse .

مثال (٦-٨)

```
using System;
class ArrayProg
 {
  public static void PrintMyArray(string[] weekDays )
   {
     foreach (string day in weekDays)
     {
       Console.WriteLine("The day is : {0} ", day);
      }
     } // end PintMyArray
public static void Main(string[ ] args)
{
  string[ ] weekDays = new string[7] { "Sunday", "Monday",
          "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };
    Console.WriteLine(" the day is ");
    PrintMyArray( weekDays);
    Console.WriteLine(" Use Reverse ");
    Array.Reverse(weekDays);
    PrintMyArray(weekDays);
    Console.WriteLine(" Use Sort ");
    Array.Sort(weekDays);
    PrintMyArray(weekDays);
     } // end method main
  } // end class
```

RESULT

The Day is

The day is : Sunday
The day is : Monday
The day is : Tuesday
The day is : Wednesday
The day is : Thursday
The day is : Friday
The day is : Saturday.

Use Reverse

The day is : Saturday
The day is : Friday
The day is : Thursday
The day is : Wednesday
The day is : Tuesday
The day is : Monday
The day is : Sunday

Use Sort

The day is : Friday
The day is : Monday
The day is : Saturday
The day is : Sunday
The day is : Thursday
The day is : Tuesday
The day is : Wednesday

Press any key to continue . . .

الطريقة (التابع)	الوظيفة
BinarySearch( )	البحث عن عنصر في مصفوفة مرتبة أحادية البعد Pos = Array.BinarySearch(src , key)
Clear( )	تهيئ أي عدد من عناصر المصفوفة بالقيمة صفر Array.Clear(src , 0 , src.Length)
Copy( )	تنسخ أي جزء من مصفوفة ما على مصفوفة أخرى Array.Copy(src , dest , dest.Length)
Sort( )	ترتب عناصر المصفوفة أحادية البعد Array.Sort(src)
Reverse( )	تعكس ترتيب العناصر في مصفوفة أحادية البعد Array.Reverse(src)
Length	خاصية ترجع طول المصفوفة Size = src.Length

الجدول(٦-١) - بعض التوابع للصف Array

مثال (٦-٩)

البرنامج التالي يستخدم الصف العشوائي Random لتوليد أسماء 12 شهراً بشكل عشوائي ضمن مصفوفة nameMonth والمطلوب :

١. أنشأ تابعاً اسمه PrintMyArray لطباعة عناصر المصفوفة nameMonth باستخدام الحلقة foreach .

٢. أدخل عناصر المصفوفة nameMonth عن طريق توليد أسماء الأشهر باستخدام الصف Random .

٣. استخدم الطريقة Sort( ) لترتيب عناصر المصفوفة .

٤. استخدم الطريقة BinarySearch( ) من أجل البحث عن الشهر April .

٥. استخدم الطريقة Reverse من أجل عكس عناصر المصفوفة .

البرنامج :

```
sing System;
class ArrayProg  {
   public static void PrintMyArray(string[ ] nameMonth)
   {
      foreach (string month in nameMonth)
      {
       Console.WriteLine("The Month is : {0} ", month);
      }
   } // end PintMyArray
public static void Main(string[ ] args)
 {
   string[ ] nameMonth= new string[12];
   Random rnd = new Random();
   int nrt ;
   for (int i = 0; i < nameMonth.Length; i++)
   {
     nrt = rnd.Next(1, 13);
     DateTime dt = Convert.ToDateTime(nrt+"/"+ nrt+"/ 2017");
     nameMonth[i] = dt.ToString("MMMM"); ;
   } // end for
  Console.WriteLine(" the Month is ");
      PrintMyArray(nameMonth);
  Console.WriteLine(" Use Reverse ");
      Array.Reverse(nameMonth);
      PrintMyArray(nameMonth);
  Console.WriteLine(" Use Sort ");
      Array.Sort(nameMonth);
      PrintMyArray(nameMonth);
```

```
    } // end method main
} // end class
```

```
RESULT
  the Month is
The Month is : October
The Month is : September
The Month is : May
The Month is : September
The Month is : August
The Month is : August
The Month is : May
The Month is : March
The Month is : May
The Month is : June
The Month is : February
The Month is : Februar
```

```
RESULT
Use Reverse
The Month is : February
The Month is : February
The Month is : June
The Month is : May
The Month is : March
The Month is : May
The Month is : August
The Month is : August
The Month is : September
The Month is : May
The Month is : September
The Month is : October
```

```
RESULT
  Use Sort
The Month is : August
The Month is : August
The Month is : February
The Month is : February
The Month is : June
The Month is : March
The Month is : May
The Month is : May
The Month is : May
The Month is : October
The Month is : September
The Month is : September
Press any key to continue . . .
```

٦-٣-٧-الكلمة المحجوزة params

تسمح الكلمة المحجوزة params بتمرير عددٍ متغيرٍ من الوسائط ذات النوع الواحد إلى تابع ما . أما بالنسبة للتابع فيستقبل هذه الوسائط على شكل مصفوفة من ذات النوع .

البرنامج في المثال(٦-١٠) التالي يحتوي على تابع اسمه Add يقبل عدداً متغيراً من الوسائط ذات النوع int ، ويقوم بحساب وطباعة مجموع عناصر المصفوفة باستخدام الكلمة المحجوزة params ، ويستخدم الحلقة foreach من اجل طباعة عناصر المصفوفة .

مثال (٦-١٠)

```
using System;
class Program {
  static void Add(params int[ ] parray)
  {
    int sum = 0 ;
    foreach (int i in parray )
        sum = sum + i;
    Console.WriteLine("Sum =" + sum);
    Console.WriteLine(" parray Elements ");
    foreach (int i in parray )
    {
        Console.Write(" {0} ", i);
    }
  }//end Add
 static void Main(string[ ] args)
 {
    Add(11, 23, 73, 40, 5);
    Console.WriteLine( );
  }//end main
} // end class
```

```
RESULT
  Sum =152
  parray Elements
  11    23    73    40    5
Press any key to continue . . .
```

٦-٤- المصفوفات ثنائية البعد

Tow-Dimensional Array

يمكن للمصفوفات في لغة #C أن تكون متعددة الأبعاد ويمكن أن يكون كل بعد بحجم مختلف ، والاستخدام الشائع للمصفوفات متعددة الأبعاد هو تمثيل الجداول Tables، والجدول (٦-٢) يحتوي على معطيات مرتبة في صورة صفوف وأعمدة، ولتمثيل الجدول نحتاج لبعدين البعد الأول يمثل الأسطر (الصفوف) والبعد الثاني يمثل الأعمدة. ويبين الشكل التالي مصفوفة مستطيلة A تحتوي على ثلاثة صفوف وأربعة أعمدة، ويتم تمثيل أي عنصر في المصفوفة A على الصورة A[i] [j] .

<u>حيث:</u>

A – اسم المصفوفة.

i – رقم السطر الذي ينتمي إليه العنصر.

j – رقم العمود الذي ينتمي إليه العنصر.

نلاحظ أن كل العناصر الموجودة في السطر الأول مثلاً يكون الدليل الأول لها هو 0 وكل العناصر الموجودة في العمود الرابع يكون الدليل الثاني لها هو 3.

	Column 0	Column1	Column2	Column 3
Row 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
Row 1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
Row 2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

الجدول (٦-٢) - يحتوي على معطيات مرتبة في صورة صفوف وأعمدة

٦-٤-١- التصريح عن المصفوفات ثنائية البعد

إن المصفوفات تحتل حجماً في الذاكرة لذا يجب على المبرمج تحديد نوع عناصر المصفوفة وعددها حتى يتسنى للمعرف تخصيص الحجم اللازم من الذاكرة لحفظ المصفوفة، والشكل العام للتصريح عن المصفوفة ثنائية البعد كما يلي:

Type [,]Array_ Name = new Type [size1, size2];

حيث إن:

Type - نوع المعطيات .

Array_Name - اسم المصفوفة، والذي يتم وضعه من قبل المبرمج مع مراعاة لقواعد تسمية المتحولات، وحيث إن كل متحول في المصفوفة يدعى عنصراً وكل عنصر في المصفوفة يجب أن يكون من نوع المصفوفة.

size1 -عدد الأسطر في المصفوفة.

size2 -عدد الأعمدة في المصفوفة.

يتم التصريح عن مصفوفة a تحتوى على x صف و y عمود كما يلي :

```
int [ , ]a = new int [ x , y ] ;
```

عند استخدام مصفوفة أحادية البعد لا بد من استخدام حلقة تكرار واحدة (for أو while أو do-while) ، بينما عند استخدام مصفوفة ثنائية البعد لا بد من استخدام حلقتي تكرار أو ما يسمى بحلقات التكرار المتداخلة Nested loops.

٦-٤-٢- إسناد قيم ابتدائية لعناصر المصفوفة ثنائية البعد

يمكن التصريح عن المصفوفة ثنائية البعد وإعطائها قيماً ابتدائية (إجراء عملية تمهيد للعناصر), كما يلي :

```
int [ , ]b = { {1,2} , {3,4} };
```

حيث:

b[1,1]=4 , b[1,0]=3 , b[0,1]=2 , b[0,0]=1

أيضاً نلاحظ في المصفوفة متعددة الأبعاد إذا تم إسناد قيم أولية لا يتوافق عددها مع حجم المصفوفة فإن المترجم سيملأ بقية العناصر أصفاراً. ويوضح البرنامج في المثال(٦-١١) كيفية إسناد قيم أولية للمصفوفات متعددة الأبعاد عند التصريح عنها.

مثال (٦-١١)

```
// initializing multidimensional arrays
using System ;
 namespace  Example56 {
class DimArry0 {
public static void Main(string [ ] args )
{
int [ , ]array1 = { {1, 2, 3}, {4, 5, 6} };
int[ , ] array2 = { { 1, 2 }, { 4 , 8 } };
```

```
    Console.WriteLine("values in array1 by row are : ");
    for (int i=0; i< 2 ; i++) {
    for (int j=0; j< 3; j++)
      Console.Write(array1[ i , j ] +" ") ;
      Console.WriteLine( );
    } //end for
  //Continued
    Console.WriteLine("values in array2 by row are : ");
    for (int i=0; i< 2 ; i++) {
    for (int j=0; j< 2; j++)
      Console.Write(array2[i,j] +" ");
      Console.WriteLine( );
    } //end for
  }//end main
 } // end class
} //end namespace
```

```
RESULT
values in array1 by row are :
1 2 3
4 5 6
values in array2 by row are :
1 2
4 8
Press any key to continue . . .
```

٦-٤-٣- المصفوفات غير المنتظمة

كل صف في المصفوفة متعددة الأبعاد هو مصفوفة بحد ذاته ، وهكذا فإنه من الممكن للصفوف أن تملك أطوال مختلفة . تعرف المصفوفة من هذا النوع بالمصفوفة غير المنتظمة . فيما يلي مثال عن إنشاء مصفوفة غير منتظمة :

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

لذلك فإن طول array[0].length يساوي 5 ، وطول array[1].length يساوي 4 ، وطول array[2].length يساوي 3 ، وطول array[3].length يساوي 2 ، وطول array[4].length يساوي 1 . إذا كانت القيم التي ستخزن في المصفوفة غير المنتظمة غير معروفة فإنه يمكن صنع مصفوفة غير منتظمة باستخدام الصيغة التالية :

```
int [ ][ ] array = new int [5][ ] ;
array [0] = new int [5] { 1,2,3,4,5} ;
array [1] = new int [4] { 2,3,4,5} ;
array [2] = new int [3] { 3,4,5} ;
array [3] = new int [2] { 4,5} ;
array [4] = new int [1] { 5} ;
```

يمكننا الآن إسناد قيم عشوائية إلى هذه المصفوفة باستخدام الحلقة التالية:

```
Random rnd = new Random( );
for (int i = 0; i < array1.Length; i++)
   for (int j = 0; j < array1[i].Length; j++)
      array1[i][j] = rnd.Next(1, 13);
```

أو بالشكل التالي :

```
for (int i = 0; i < array1.Length; i++)
   for (int j = 0; j < array1[i].Length; j++)
```

```
{
  Console.WriteLine("The Element is : {0} ",array1[i][j]);
}
```

تحتاج الصيغة new int [5][ ] إلى تحديد الدليل الأول من أجل إنشاء مصفوفة ، لأن الصيغة التالية خاطئة

```
new int [ ][ ]
```

مثال (٦-١٢)

البرنامج التالي يقوم بما يلي :

١. اسناد قيم ابتدائية لمصفوفة منتظمة array2 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .

٢. اسناد قيم ابتدائية لمصفوفة غير منتظمة array1 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .

٣. استخدام الصف Random لتوليد قيم عشوائية واسنادها للمصفوفة المنتظمة array1 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .

٤. يجب أن يكون الخرج على شكل مصفوفة ثنائية البعد .

البرنامج :

```
using System ;
class DimArry0 {
public static void Main(string [ ] args ) {
int [ ][]array1 =new int[2][];
  array1[0]=new int[3] {1, 2, 3};
  array1[1]=new int[2] {4, 5} ;
int [ , ]array2 = { { 1, 2 }, { 4 , 8 } };
```

```
Console.WriteLine("values in array1 by row are : ");
for (int i = 0; i < array1.Length; i++) {
   for (int j = 0; j < array1[i].Length; j++)
      Console.Write(array1[i][j] + " ");
      Console.WriteLine();
 } //end for
Console.WriteLine("values in array2 by row are : ");
 for (int i=0; i< 2 ; i++) {
   for (int j=0; j< 2; j++)
      Console.Write(array2[i,j] +" ");
      Console.WriteLine();
   } //end for
//--------------------
   Console.WriteLine("values in NEW array1 by row are : ");
   Random  rnd = new  Random( );
   for (int i = 0; i < array1.Length; i++)
      for (int j = 0; j < array1[j].Length; j++)
         array1[i][j] = rnd.Next(1, 13);
  // foreach (int s in array1)
   for (int i = 0; i < array1.Length; i++)  {
      for (int j = 0; j < array1[i].Length; j++)
         Console.Write("{0,2} ", array1[i][j]);
          Console.WriteLine();
    }
   }//end main
   } // end class
```

```
RESULT
Values in array1 by row are :
 1 2 3
 4 5
Values in array2 by row are :
 1 2
 4 8
Values in NEW array1 by row are :
 1  7  4
 1  5
Press any key to continue . . .
```

٦-٤-٤- إدخال وإخراج عناصر المصفوفة ثنائية البعد

يمكن إدخال عناصر المصفوفة من قبل المستخدم ، وأيضاً يمكن إخراج (طباعة) عناصر المصفوفة ثنائية البعد بوساطة حلقتين (حلقتي for ، أو while ، أو do/while ، أو خليط بينهم) . عادة قبل استخدام المصفوفة يمكن تصفيرها ، وذلك بإعطاء قيم ابتدائية تساوي الصفر ، ويوضح المثال(٦-١٣) كيفية إدخال وإخراج عناصر المصفوفة ثنائية البعد.

مثال (٦-١٣)

اكتب برنامجاً بلغة #C يقوم بما يلي :

١. إدخال درجات 10طلاب في خمسة مقررات وتخزينها في مصفوفة ثنائية البعد a[10,5] .

٢. حساب وطباعة معدل كل طالب في المقررات الخمسة .

٣. حساب وطباعة المعدل الأكبر .

البرنامج :

```
// multidimensional arrays
using System ;
namespace   Example {
class DimArryl  {
public static void Main(string [ ] args )
{
  int [ , ] Mark;
  Mark = new int[10,5] ;
  int ormax ;
  double  sum=0.0 , maxav ;
  double  [ ]avr;
  avr =new double [10];
  Console.WriteLine("Enter Mrka[ , ] values  :  ");
  for (int  i=0; i< 10 ; i++ )
    for (int  j=0; j< 5 ; j++ ){
      int x = int.Parse( Console.ReadLine() );
      Mark[i,j]=x;
    }//end for
  //Continued
    for (int  k=0; k<10; k++) {
          sum=0.0 ;
     for (int  j=0; j<5; j++) {
          sum=sum + Mark[ k , j ] ; }
          avr[ k ]=sum / 5 ;
     }//end for
    maxav = avr[0];
    ormax =1;
    for (int i2=0; i2<10; i2++)   {
         if( avr[ i2 ] > maxav ) {
            maxav = avr[ i2 ] ;
            ormax = i2+1 ;
         }//end if
     } end for
```

```
for (int i1=0; i1<10; i1++)
  Console.WriteLine(" avr ["+i1+ "]=" +avr[i1] );
Console.WriteLine(" The greatest average is " + maxav );
Console.WriteLine(" The greatest number is "+ ormax )
}//end main
} // end class
} //end namespace
```

```
RESULT
av[0]=187.5          av[1]=187
av[2]=180            av[3]=189.5
av[4]=181.5          av[5]=165
av[6]=195            av[7]=197.5
av[8]=176            av[9]=187
The greatest average is  197.5
The greatest number is   7
Press any key to continue . . .
```

٦ – ٤ – ٥ – تمرير المصفوفات ثنائية البعد كوسطاء للتوابع

عند تمرير مصفوفة ثنائية البعد كوسيط لتابع عند استدعائه في التابع Main( ) يجب استخدام اسمها فقط بدون أقواس ، فعلى سبيل المثال ، إذا تم التصريح عن المصفوفة array على الشكل التالي :

```
int [ , ] array  = int [2,3]  ;
```

فإن الاستدعاء التالي للتابع printArray( ) كما يلي :

```
printArray(array ) ;
```

وعلى اعتبار أن المصفوفات يتم تمريرها بالعنوان فإن التابع المُستدعى يقوم باستخدام اسم المصفوفة array من أجل الرجوع إلى المصفوفة الأصلية الموجودة

لدى التابع الذي تم استدعاء المصفوفة ضمنه، ويوضح المثال(٦ – ١٤) كيفية تمرير مصفوفة ثنائية البعد كوسيط لتابع .

مثال (٦–١٤)

يوضح البرنامج التالي كيفية إسناد قيم أولية للمصفوفات متعددة الأبعاد عند التصريح عنها، وكيفية تمرير المصفوفات كوسائط للتوابع وطباعتها ، ثم استخدام التابع printArray( ) لطباعة عناصر المصفوفة .

البرنامج :

```
// initializing multidimensional arrays
using System ;
class DimArry {
public static void Main(string [ ] args )
{
 int[ ][ ] array1 = new int[2][];// {{1, 2, 3}, {4, 5, 6}};
  array1[0] = new int[3] { 1, 2, 3 };
  array1[1] = new int[3] { 4, 5, 6 };

  int[ ][ ] array2 = new int[2][ ]; // { { 1, 2 } , { 4 } };
  array2[0] = new int[2] { 1, 2 };
  array2[1] = new int[1] { 4 };
  Console.WriteLine("values in array1 by row are : ");
   printArray( array1 ) ;
//Continued
  Console.WriteLine("values in array2 by row are : ");
  printArray(array2);
  }//end main
 public static void printArray(int [ ][ ] array) {
  for (int i=0; i< array.Length ; i++) {
    for (int j=0; j< array[ i ].Length ; j++)
```

```
  Console.Write(array[i][j] +" ");
  Console.WriteLine();
 } //end for
}//end function
} // end class
```

```
RESULT
values in array1 by row are :
 1   2   3
 4   5   6
values in array2 by row are :
 1   2
 4
Press any key to continue . . .
```

مثال (٦-١٥)

اكتب برنامجاً بلغة #C يقوم بما يلي :

- إدخال 12عدداً صحيحاً ، ويتم تخزينها في مصفوفة ثنائية البعد a[3,4]
- طباعة عناصر المصفوفة a .
- حساب المتوسط الحسابي لعناصر المصفوفة وطباعته .

استخدم التوابع في البرنامج كما يلي :

- التابع printArray( ) لطباعة عناصر المصفوفة .
- التابع printArray1( ) لحساب مجموع عناصر المصفوفة وحساب المتوسط الحسابي .

البرنامج :

```
// multidimensional arrays
using System ;
namespace  Example {
class DimArry2 {
public static void Main(string [ ] args )
{
  int [ , ] Arry;
  Arry = new int[3,4] ;
  Console.WriteLine("Enter Arry [ , ] values : ");
  for (int i=0; i<3; i++ )
    for (int j=0; j<4; j++ ){
        int x = int.Parse( Console.ReadLine() );
    Arry[i,j]=x;
  }// end for
  printArray(Arry);
  printArray1(Arry);
 }//end main
//Continued
public static void printArray ( int [ , ] a) {
  for (int i=0; i<3; i++){
  for (int j=0; j<4; j++)
      Console.Write ( a[ i , j ] +"  " );
    Console.WriteLine( );
  }//end for
}//End printArray
public static void printArray1(int [ , ] a) {
   int sum=0;
    for (int i=0; i<3; i++)
      for (int j=0; j<4; j++)
          sum = sum + a[ i , j ] ;
   double  avr = sum/12;
   Console.WriteLine(" avr = "+  avr );
  }//end printArray1
```

```
} // end class
} //end namespace
```

```
RESULT
Enter a[ , ] values  :
11 22 33 44 55 66 77 88 99 12 13 14
Print Array
11 22 33 44
55 66 77 88
99 12 13 14
Avr = 44

Press any key to continue . . .
```

مثال (٦-١٦)

اكتب برنامجاً بلغة #C يقوم بما يلي :

١. إدخال عناصر المصفوفة a[3,3] وهي مصفوفة ثنائية البعد ذات أعداد حقيقية .

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{21} & a_{21} & a_{21} \end{bmatrix}$$

٢. طباعة عناصر المصفوفة a .

٣. حساب المتوسط الحسابي للأعداد التي تقع فوق القطر الرئيسي للمصفوفة وطباعته .

ملاحظة :

استخدم التوابع في كتابة البرنامج .

البرنامج :

```
// multidimensional arrays
using System ;
 namespace   Example  {
class DimArry3  {
public static void Main(string [ ] args )
{
  int [ , ] A;
  A = new int[3,3] ;
  Console.WriteLine("Enter A[ , ] values  :  ");
  for (int  i=0; i<3; i++ )
    for (int  j=0; j<3; j++ ){
      int x = int.Parse( Console.ReadLine() );
      A[i,j]=x;
   }//end for
  printArray1(A);
  printArray(A);
  }//end main
public static void printArray ( int [ , ] a )
{
    int sum=0 ;
    int n = 0 ;
    for (int i=0; i<3; i++) {
       for (int j=0; j<3; j++)
          if( i<= j ) {
          sum=sum + a[ i , j ] ;
           n = n+1 ; }
   }// end for
   double   avr = sum / n ;
   Console.WriteLine("\n avr = "+  avr ) ;

 }//end printArray1
public static void printArray1(int [ , ] a) {
          Console.WriteLine( " Print Array");
     for (int i=0; i<3; i++){
```

```
        for (int j=0; j<3; j++)
            Console.Write ( a[ i , j ] +" ");
      Console.WriteLine();
      }//end for
    }//end printArry
  } // end class
} //end namespace
```

```
RESULT
Enter a[ , ] values  :
 11 22 33 55 66 77 99 12 13
-----------------
 11   22   33
 55   66   77
 99   12   13
-----------------
Avr = 44

Press any key to continue . . .
```

مثال (٦-١٦)

اكتب برنامجاً بلغة #C يقوم بتصحيح مؤتمت لامتحان مقرر البرمجة لخمسة طلاب بحيث يكون عدد الأسئلة المؤتمتة عشرة ويكون عدد الأجوبة المحتملة لكل سؤال أربعة (A,B,C,D) والمطلوب :

١. إدخال عدد الطلاب وعدد الأسئلة، بحيث يتم تخزينها في مصفوفة محرفية ثنائية البعد answers [5,10]، أو إسناد قيم ابتدائية للمصفوفة تمثل أجوبة الطلاب .

٢. إدخال (أو إسناد) الـ Key ، وهو مصفوفة محرفية أحادية البعد تمثل الأجوبة الصحيحة key[10].

٣. أنشئ تابعاً اسمه Results بوسيطين الوسيط الأول مصفوفة ثنائية البعد والوسيط الثاني مصفوفة أحادية البعد ، يقوم بحساب مجموع عدد الأسئلة الصحيحة لكل طالب .

٤. أنشئ تابعاً اسمه MaxResults يقوم بحساب أعلى درجة حصل عليها أحد الطلاب .

٥. التابع Main يستدعي التابعين ويطبع درجات الطلاب وأعلى درجة .

البرنامج :

```
using System;
class Program  {
static void Main(string[] args)
{
  /* char [ , ]answers = { { 'A' , 'A' , 'D' , 'A' , 'C' , 'A' , 'A' , 'B' , 'B' , 'A' },
                          { 'A' , 'D' , 'C' , 'A' , 'C' , 'D' , 'D' , 'B' , 'C' , 'A' },
                          { 'A' , 'B' , 'D' , 'C' , 'C' , 'A' , 'D' , 'C' , 'B' , 'C' },
                          { 'A' , 'B' , 'D' , 'A' , 'C' , 'A' , 'D' , 'B' , 'B' , 'A' },
                          { 'A' , 'C' , 'A' , 'C' , 'A' , 'D' , 'D' , 'C' , 'A' , 'A' },
                 }; */
   char ch;
   char [ , ]answers=new char[5,10] ;
   Console.WriteLine("Enter answers ");
   for ( int i=0 ; i< 10 ; i++ )
     for ( int j=0 ; j< 5 ; j++ ){
       ch = char.Parse( Console.ReadLine() );
       answers[i , j]=ch ;
   } //end loop
   char [ ] keys = {'A','B','D','C','C','A','D','B','B','A' };
    Results (  answers , keys ) ;
  }//end main
```

```
public static void  Results ( char[ , ]answers,char [ ]keys ) {
  int [ ] correctCount1= new int[5] ;
  for ( int i=0 ; i< 5 ; i++ ){
  int correctCount =0 ;
    for (int j = 0; j < 10; j++)
   {
      if(answers [i,j] == keys[j])
       correctCount++ ;
   }//end for j
  correctCount = correctCount *10 ;
 correctCount1[ i]=correctCount ;
   Console.WriteLine("student"+(i+1)+" 's correct Count is "
                                                  +correctCount  ) ;
  } //end for i
  Console.Write("Max correct Count Of  student="
                                  +MaxResults(correctCount1) );
public static int MaxResults(int[ ] max)
 {
   int xmax=max[0] ;
   for ( int i=0 ; i< max.Length ; i++ ){
    if( max[i] > xmax )
    xmax = max[i] ;
   }// end for
  return xmax ;
```

```
RESULT
 student 1 's correct  Count is  70
 student  2 's correct  Count is  50
 student  3 's correct  Count is  80
 student  4 's correct  Count is  90
 student  5 's correct  Count is  40
  Max correct Count Of student = 90

Press any key to continue . . .
```

مسائل عامة

١- اكتب برنامجاً بلغة #C يقوم بما يلي :

- إدخال 12 عدداً صحيحاً وتخزينها في مصفوفة ثنائية البعد a [4,3] .
- إدخال 6 أعداد صحيحة وتخزينها في مصفوفة ثنائية البعد b [3,2] .
- إجراء عملية ضرب المصفوفتين a [4,3] و b [3,2] وتخزين النتيجة في المصفوفة c [4][2] .
- طباعة عناصر المصفوفات الثلاثة a [4,3] و b [3,2] و c [4,2].

==========

٢- اكتب برنامجاً بلغة #C يقوم بما يلي :

- إدخال 9 أعداد صحيحة وتخزينها في مصفوفة ثنائية البعد a [3,3] .
- حساب مجموع ومتوسط الأعداد التي تقع على وفوق القطر الرئيسي للمصفوفة.
- طباعة عناصر المصفوفة المدخلة على شكل مصفوفة مربعة.
- طباعة مجموع ومتوسط الأعداد الذين تم إيجادهما .

==========

٣- اكتب برنامجاً بلغة #C يقوم بما يلي:

- إدخال 5 أعداد صحيحة وتخزينها في مصفوفة أحادية البعد n [5] .
- حساب مربعات ومكعبات الأعداد المدخلة وتخزينها في المصفوفتين s[5] و c[5] على الترتيب.
- طباعة عناصر المصفوفات الثلاثة على أسطر مستقلة.

==========

٤- اكتب برنامجاً بلغة C# يقوم بما يلي :

- إدخال 25 عدداً صحيحاً من لوحة المفاتيح وتخزينها في مصفوفة ثنائية البعد y [5,5] .
- إيجاد العدد الأصغر من بين جميع الأعداد الواقعة على وفوق القطر الرئيسي.
- طباعة عناصر المصفوفة المدخلة على شكل مصفوفة مربعة.
- طباعة قيمة العدد الأصغر الذي تم إيجاده.

==========

٥- اكتب برنامجاً بلغة C# يقوم بما يلي :

- إدخال 25 عدداً حقيقياً من لوحة المفاتيح وتخزينها في مصفوفة ثنائية البعد x [5,5] .
- حساب المتوسط الحسابي لجميع الأعداد الواقعة تحت القطر الثانوي.
- طباعة عناصر المصفوفة المدخلة على شكل مصفوفة مربعة.
- طباعة قيمة المتوسط الحسابي الناتج.

==========

٦- اكتب برنامجاً بلغة C# يقوم بما يلي :

- إدخال 16 عدداً حقيقياً من لوحة المفاتيح وتخزينها في مصفوفة ثنائية البعد x [4][4] .
- حساب المتوسط الحسابي لعناصر المصفوفة الواقعة في السطر الأول.
- إيجاد العنصر الأكبر من بين عناصر المصفوفة الواقعة في العمود الأول.
- طباعة عناصر المصفوفة المدخلة على شكل مصفوفة مربعة.
- طباعة النتائج التي تم التوصل إليها من خلال الطلبين الثاني والثالث.

٧- اكتب برنامجاً بلغة #C يقوم بما يلي :

❖ يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20].

❖ ويحتوي البرنامج على تابعين:

➢ التابع الأول اسمه SumAve( ) مهمته حساب المتوسط الحسابي لعناصر المصفوفة.

➢ أما التابع الثاني اسمه EvenArry مهمته حساب مجموع الأعداد الزوجية في المصفوفة .

❖ يستدعي التابع Main التابعين ويطبع المتوسط ومجموع الأعداد الزوجية .

==========

٨- اكتب برنامجاً بلغة #C يقوم بما يلي :

❖ يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20].

❖ ويحتوي البرنامج على تابعين:

➢ التابع الأول اسمه MaxArry( ) مهمته حساب العنصر الأكبر للمصفوفة .

➢ أما التابع الثاني اسمه OddArry مهمته حساب مجموع الأعداد الفردية في المصفوفة .

❖ يستدعي التابع Main التابعين ويطبع العنصر الأكبر ومجموع الأعداد الفردية .

==========

٩- اكتب برنامجاً بلغة #C يقوم بما يلي :

❖ يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20].

❖ ويحتوي البرنامج على تابعين:

- التابع الأول اسمه SumAve( ) مهمته حساب المتوسط الحسابي لعناصر المصفوفة.
- أما التابع الثاني اسمه EvenArry مهمته حساب مجموع الأعداد الزوجية في المصفوفة .

❖ التابع Main يستدعي التابعين ويطبع المتوسط ومجموع الأعداد الزوجية

==========

١٠- اكتب برنامجاً بلغة #C يقوم بما يلي :

❖ يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20].

❖ ويحتوي البرنامج على تابعين:

- التابع الأول اسمه MaxArry( ) مهمته حساب العنصر الأكبر للمصفوفة .
- أما التابع الثاني اسمه OddArry مهمته حساب مجموع الأعداد الفردية في المصفوفة .

❖ التابع Main يستدعي التابعين ويطبع العنصر الأكبر ومجموع الأعداد الفردية .

==========

١١- أكتب برنامجاً بلغة #C يقوم بما يلي:

❖ إدخال 10 أعداد صحيحة وتخزينها في مصفوفة أحادية البعد a[10].

❖ يستخدم تابع من النوع void اسمه sort لترتيب الأعداد المدخلة بشكل تصاعدي.

❖ التابع main يستدعي التابع sort ويطبع طباعة الأعداد المدخلة بعد عملية الترتيب .

==========

١٢-اكتب برنامجاً بلغة #C يقوم بما يلي:

- ❖ إدخال 9 أعداد صحيحة وتخزينها في مصفوفة ثنائية البعد[3,3]x .
- ❖ طباعة عناصر المصفوفة [3,3]x على شكل مصفوفة مربعة .
- ❖ يستخدم تابع من النوع void لإيجاد كافة الأعداد الأولية من بين عناصر المصفوفة المُدخلة ، ومن ثم طباعة هذه الأعداد وأيضاً طباعة عددها.
- ❖ يستدعي التابع Main التابع لطباعة النتائج .

==========

١٣- اكتب برنامجاً بلغة #C يقوم بما يلي:

- يُدخل أي عدد صحيح n .
- يستخدم تابع اسمه factiral ، مهمته حساب العاملي (!) لأي عدد صحيح .
- يقوم بحساب قيمة المقدار الرياضي التالي:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

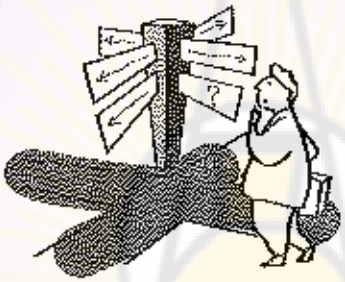
- يطبع النتيجة في التابع الرئيسي main

==========

الفصل السابع

القسم العملي

جلسات مخبر البرمجة (1)



الجلسة الأولى

بنية البرنامج -الإدخال والإخراج

الهدف من الجلسة :

١. التعرف على بنية البرنامج في لغة C# ، وكيفية إخراج المعطيات على الشاشة.

٢. التعرف على تقنيات الإدخال من لوحة المفاتيح ومعرفة جميع أنواع المتحولات (المتحولات الصحيحة والحقيقية والمحرفية...الخ)، وكيفية تعريفها و التعامل معها.

برامج الجلسة :

Q1- لدينا برنامج بلغة C# يحتوي على صف اسمه Welcome يقوم بطباعة العبارة التالية على سطر واحد : (Welcome to C# Programming).

المطلوب:

أ- نفذ البرنامج التالي :

```
1. using System;
2. namespace Example1 {
3.  class Welcome {
4.   static void Main(string[] args)
5.   {
6.    Console.WriteLine(" Welcome to C#
                                Programming \n");
7.   }//end main
8.  } //end class
9. }// end namespace
```

ب- أعد كتابة البرنامج السابق لطباعة العبارة التالية :

```
Welcome To C#  programming
```

على الشكل التالي :

```
Welcome
 To
Java
programming
```

١. باستخدام العبارة Console.WriteLine( )

٢. باستخدام محرف الهروب \n

٣. استخدم محرف الهروب \t بدلا من \n ، ماذا تلاحظ ؟

Q2 - يمتلك التابعان Write و WriteLine ميزة سهلة الاستعمال تتيح لنا تمرير المتحولات كوسطاء إضافية يمكن تنسيقها في الخرج ، والمطلوب :

أ- نفذ البرنامج التالي :

```
1. sing System;
2. class myProgram {
3. static void  Main( string [ ] args )
4. {
5.    string  name = " Waseem  younes ";
6.    double  temprature = 37.5;
7.    Console.WriteLine(" Name {0} \n temperature  {1} ",
                                    name, temprature);
8.    int x = 20, y = 30;
9.  Console.WriteLine(" ( X , Y ) is ( {0,2}  , {1,2} )", x, y);
10.Console.WriteLine("{0}\n{1}", "Welcome to",
                                    "C# Programming!");
11.Console.ReadKey( );
12.    } // end method main
13.  } // end class
```

ب- ناقش الخرج الناتج .

ت- اضف السطر التالي في البرنامج السابق :

```
Console.WriteLine(" X  is  {0:P} \n Y is  {1:p} ", x, y);
```

ناقش الخرج الناتج .

ث- اضف السطر التالي في البرنامج السابق :

```
Console.WriteLine(" X  is  {0:X} \n Y is  {1:X} ", x, y);
```

ناقش الخرج الناتج .

==========

Q3- البرنامج التالي يحتوي على صف اسمه Program يقوم بإدخال الاسم name (اسمك) من نمط string وطباعة العبارة التالية :

```
Hello  Waseem , Welcom to C# Programming !
```

المطلوب:

أ- نفذ البرنامج التالي :

```
using System;
class Program {
static void Main(string[] args)
{
  string   input = " ";
  Console.WriteLine(" Enter Your Name");
   input= Console.ReadLine( );
   Console.WriteLine(" Hello , " + input +
                        " Welcom to C# Programming ! ");
    } // end method main
 } // end class
```

==========

Q4 - البرنامج التالي يحتوي على صف اسمه Sum يقوم بإدخال العددين , integer1 integer2 ، من نمط int وحساب مجموع العددين وطباعة النتيجة .

المطلوب:

أ- نفذ البرنامج التالي :

البرنامج :

```
using System;
class Program4 {
static void Main(string[ ] args)
{
   int  integer1, integer2  ;
   string x;
   Console.WriteLine("   Enter first Number\n ");
   integer1 = int.Parse( Console.ReadLine() );
   Console.WriteLine("  Enter   second Number\n ");
   integer2 = int.Parse( Console.ReadLine() );
   int sum = integer1 + integer2;
   Console.WriteLine("  sum =  {0} " , sum);
 } // end method main
 } // end class Name
```

ج- أعد كتابة البرنامج ليقوم بإدخال وطباعة مجموع عددين من نوع double

==========

ح- أعد كتابة البرنامج ليقوم بإدخال وطباعة السلسلة المحرفية باستخدام

Convert.ToString

Welcome To Java programming

==========

الأسئلة

١. حدد ما إذا كانت العبارات الآتية صحيحة أم خاطئة:

- التعليقات تجبر الحاسوب على طباعة النص الذي يلي // على الشاشة عند تنفيذ البرنامج.
- محرف الهروب \n يجبر المؤشر للانتقال إلى سطر جديد.
- برنامج #C والذي يقوم بطباعة ثلاثة أسطر على الشاشة يجب أن يحتوي على ثلاث عبارات تستعمل WritLine.

٢. ما هو الخرج من العبارة الآتية:

```
Console.WriteLine(" \n **\n ***\n ****\n ") ;
```

==========

الجلسة الثانية

أنواع المتحولات –المتحولات المحلية والعامة – الزيادة والنقصان

الهدف من الجلسة :

١. معرفة كيفية التحويل بين المتحولات المختلفة لأنماط المعطيات ، وكتابة التحويلات اللازمة .
٢. معرفة مجال المتحولات المحلية والعامة في البرنامج .
٣. المقارنة بين الزيادة من اليمين (الأمام) واليسار (الخلف) وكذلك النقصان . .

برامج الجلسة :

Q1- يبين البرنامج التالي التحويل بين المتحولات المختلفة ، ويقوم البرنامج بتحويل المتحول i من نوع int إلى byte وإسناده إلى المتحول b الذي هو من نوع byte وطباعته، والمتحول d من double إلى int إلى المتحول i وطباعته ، و المتحول d من double إلى byte إلى المتحول b وطباعته.

والمطلوب :

أ- نفذ البرنامج التالي :

```
sing System;
   // Demonstrate casts.
class Conversion {
static void Main(string[ ] args)
{
    byte  b;
    int  i = 257;
    double  d = 323.142;
   Console.WriteLine("\nConversion of int to byte.");
```

```
    b = (byte) i;
    Console.WriteLine("i and b : " + i + "  to " + b);
    Console.WriteLine("\nConversion of double to int.");
    i = (int) d;
    Console.WriteLine("d and i : " + d + "  to " + i);
    Console.WriteLine("\nConversion of double to byte.");
    b = (byte) d;
    Console.WriteLine("d and b  : " + d + " to " + b);
   }// end main
}// end class
```

ب – أجر بعض التعديلات على تحويل المتحولات بقيم مختلفة وأضف بعض الأنواع الأخرى.

==========

Q2- البرنامج التالي يعرض مجال المتحولات المحلية ضمن الكتل ، حيث المتحول x عام بينما المتحول y هو خاص ضمن الكتلة if .

و المطلوب :

أ- نفذ البرنامج التالي :

```
1. // Demonstrate block scope.
2. using System ;
3. class Scope {
4. static void Main(string [ ] args )
5. {
6.   int  x; // known to all code within main
7.   x = 10;
8.   if( x == 10) { // start new scope
9.   int  y = 20; // known only to this block
10.  // x and y both known here.
```

```
11.  Console.WriteLine("x and y: {0} ,{1} ",x, y);
12.   x = y * 2;
13.   }//End if
14.   // y = 100; // Error! y not known here
      // x is still known here.
15.   Console.WriteLine("x is " + x);
16.  } // end method main
17.} // end class Name
```

- هل المتحول x في السطر 6 عام أم خاص ضمن الكتلة ؟ ولماذا ؟
- هل المتحول y في السطر 9 عام أم خاص ضمن الكتلة ؟ ولماذا ؟
- إذا أسندنا للمتحول y القيمة 100 كما هو مبين في السطر 15 ماذا يحدث ؟ ولماذا؟
- إذا بدلنا السطر 14 كما يلي :

```
Console.WriteLine ("x is " + x +" y is "+y);
```

ماذا يحدث ؟ ولماذا ؟

Q3- البرنامج التالي يبين أهم العمليات الحسابية الأساسية : مثل % , * , - , + ، وذلك لمتحولات من نوع Integer Arithmetic ومن نوع Floating Point Arithmetic

والمطلوب :

نفذ البرنامج التالي :

```
// Demonstrate the basic arithmetic operators.
using System ;
class BasicMath {
public static void Main(string [ ]args)  {
// arithmetic using integers
```

```
Console.WriteLine("Integer Arithmetic");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c - a;
int e = -d;
Console.WriteLine("a = {0} " , a);
Console.WriteLine("b = {0} " , b);
Console.WriteLine("c = {0} " , c);
Console.WriteLine("d = {0} " , d);
Console.WriteLine("e = {0} " , e);
// arithmetic using doubles
Console.WriteLine("\nFloating Point Arithmetic");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc - a;
double de = -dd;
Console.WriteLine("da = {0} " , da);
Console.WriteLine("db = {0} " , db);
Console.WriteLine("dc = {0} " , dc);
Console.WriteLine("dd = {0} " , dd);
Console.WriteLine("de = {0} " , de);
// Demonstrate the % operator.
int x = 42;
double y = 42.25;
Console.WriteLine("x mod 10 = " + x % 10);
Console.WriteLine("y mod 10 = " + y % 10);
} // end method main
} // end class
```

Q4- البرنامج التالي يبين الزيادة من اليمين(الخلف) واليسار (الأمام) ويقوم بطباعة قيم المتحولات قبل وبعد الزيادة.

والمطلوب :

١. نفذ البرنامج التالي :

```
using System;
class Increment {
public static void Main(string[] args)
 {
     int k = 4;
     ++k;
     k++;
     Console.WriteLine("k=" + k);
     int i = k++;
     Console.WriteLine("i=" + i + "  k = " + k);

     int j = ++k;
     Console.WriteLine("j=" + j + "  k = " + k);
   } // end method main
} // end class Name
```

==========

ب-أعد كتابة البرنامج السابق من أجل النقصان من اليمين(الخلف) واليسار (الأمام) ويقوم بطباعة قيم المتحولات قبل وبعد النقصان .

==========

الأسئلة

١. أكتب عبارات بلغة البرمجة #C تقوم بالآتي:

a. تعريف المتغيرات x، y، z و result لتكون من النوع int.

b. الطلب من المستخدم إدخال ثلاثة أرقام صحيحة.

٢. حدد ما إذا كانت العبارات الآتية صحيحة أم خطأ:

a. يجب الإعلان عن المتغيرات قبل استعمالها في البرنامج.

b. يجب تحديد نوع المتغيرات عند الإعلان عنها.

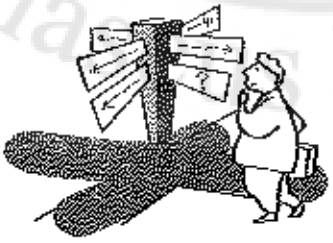
c. لا تفرق #C بين المتغيرين Number و number .

٣. اكتب برنامجاً يطبع على الشاشة أي معلومات شخصية عنك تريدها كـ (اسمك ، رقمك الجامعي ، تاريخ ميلادك ، عنوانك ....) بشكل مُنسق ؛ مستخدماً محارف الهروب (\n) و (\t).

٤. اكتب برنامجاً يقوم بحساب مربع عدد حقيقي يُدخله المستخدم من لوحة المفاتيح ؛ ثم يطبع النتيجة على الشاشة .

٥. اكتب برنامجاً لتنفيذ العبارة الرياضية التالية ، بصيغ مختلفة باستخدام الأقواس.

```
y = 6+12/6*2-1;
y = (6+12)/(6*2-1);
y = (6+12/6)*2-1;
```



الجلسة الثالثة

(بنى التحكم)

الهدف من الجلسة :

١. التعرف على استخدام البنى الشرطية (if) ، و العبارة (if...else)

٢. التعرف على استخدام البنية switch

برامج الجلسة :

Q1- البرنامج التالي يوضع التعليمة if / else والتعليمة if ، حيث يتم إدخال رقم الشهر وطباعة الفصل الذي ينتمي إليه هذا الشهر .

والمطلوب :

a) نفذ البرنامج التالي :

```
// Demonstrate if-else-if statements.
using System ;
class IfElse {
public static void Main(string [ ] args )
{
 int month = 4; // April
 String  season;
 if(month == 12 || month == 1 || month == 2)
 season = "Winter";
 else if(month == 3 || month == 4 || month == 5)
 season = "Spring";
 else if (month == 6 || month == 7 || month == 8)
  season = "Summer";
 else if (month == 9 || month == 10 || month == 11)
  season = "Autumn";
 else
```

```
  season = "Bogus Month";
 Console.WriteLine("April is in the " + season + ".");
 } // end method main
} // end class Name
```

---

b) أعد كتابة البرنامج السابق لإدخال رقم الشهر من قبل المستخدم وطباعته.

c) أعد كتابة البرنامج السابق باستخدام التعليمة Switch حيث يتم إدخال رقم الشهر وطباعة الفصل الذي ينتمي إليه هذا الشهر .

Q2- البرنامج التالي يقوم بإدخال عددين حقيقيين x , y ، وإدخال الرمز ch من نوع char بحيث إذا كان الرمز المدخل هو(+) يقوم بجمع العددين ، وإذا كان (−) يقوم بطرح العددين ، وإذا كان (*) يقوم بضرب العددين ، وإذا كان (/) يقوم بقسمة العددين وطباعة النتيجة .

والمطلوب :

a) نفذ البرنامج التالي :

```
using System ;
class Arethmatic {
public static void Main(string  [ ] args )
{
  double  x , y ;
  char  ch ;
  Console.WriteLine("inter yuor x  &  y & ch ");
  x = double.Parse( Console.ReadLine() );
  y = double.Parse( Console.ReadLine() );
  ch = char.Parse(Console.ReadLine() ) ;
  Console.WriteLine("----------------------------");
```

```
 Console.WriteLine( " x = " +x+"   y="+ y) ;
 Console.WriteLine("-----------------------------");
 if( ch == '+')
  Console.WriteLine( x + y );
 //---------------------------
  else if( ch == '-')
  Console.WriteLine( x - y );
 //---------------------------
  else if( ch == '*')
  Console.WriteLine( x * y );
 //---------------------------
  else if( ch == '/')
  Console.WriteLine( x / y );
 //---------------------------
  else
   Console.WriteLine( "Sumbol Erorr Input ");
   Console.WriteLine( "---------------------------");
 } // end method main
} // end class Name
```

==========

b) أعـــد كتابـــة البرنـــامج الســـابق باســـتخدام عبـــارة `Console.WriteLine()` واحدة فقط لطباعة النتيجة كما يلي :

`z = x + y` حيث z هو المجموع أو أي عملية حسابية أخرى ،

واستخدام عبارة ثانية `Console.WriteLine()` لطباعة رسالة خطأ إذا كانت ch غير صحيحة .

c) أعد كتابة البرنامج السابق باستخدام التعليمة Switch حيث يتم إدخال رقم الشهر وطباعة الفصل الذي ينتمي إليه هذا الشهر .

الأسئلة

١- اكتب برنامجاً بلغة C# يقوم بإدخال عدد حقيقي من لوحة المفاتيح ؛ بحيث يطبع كلمة (Positive) إذا كان العدد موجباً ، ويطبع عبارة (Negative) إذا كانت العدد سالباً ، أما إذا كان مساوياً للصفر يطبع عبارة (Zero) ؛ و يطبع الناتج على الشاشة.

٢- اكتب برنامجاً بلغة C# لإيجاد جذور معادلة من الدرجة الثانية :

$$ax^2 + bx + c = 0$$

حيث يطلب من المستخدم ادخال الثوابت a , b , c من لوحة المفاتيح .

٣- اكتب برنامجاً بلغة C# لإدخال ثلاثة أعداد حقيقية من لوحة المفاتيح وإيجاد العدد الأكبر (Max number) .

٤- اكتب برنامجاً بلغة C# يقوم بإدخال درجة الطالب Degree ويقوم بطابعة كلمة ناجح Passed إذا كانت درجة الطالب Degree ≥ 60 أو يطبع كلمة راسب Failed .

الجلسة الرابعة

(بنية حلقة التكرار for)

الهدف من الجلسة :

١. التعرف على استخدام بنية الحلقة (for) .

٢. استخدام العبارة (break) في حلقة التكرار (for) .

٣. استخدام العبارة (continue) في حلقة التكرار (for).

برامج الجلسة :

Q1- البرنامج التالي يطبع الأعداد الصحيحة من 1 إلى 10.

والمطلوب :

a) نفذ البرنامج التالي :

```
using System ;
class Increment {
public static void Main(string   [ ] args ) {

for ( int i=1 ; i<=10 ; i++ )
  {
    Console.WriteLine(" i = {0}" ,i  ) ;
  }//End for
 } // end method main
} // end class Name
```

b) عدّل البرنامج السابق لطباع الأعداد من مضاعفة العدد 3 .

c) عدّل البرنامج السابق لطباع الأعداد التي هي أصغر من العدد 6 باستخدام العبارة break للخروج من الحلقة .

d) عدّل البرنامج السابق لطباع الأعداد من 1 إلى 10 ما عدا العدد 6 باستخدام العبارة continue في الحلقة .

e) عدّل البرنامج السابق لحساب وطباعة مجموع الأعداد من 1 إلى 10 .

f) عدل البرنامج السابق لحساب وطباعة المتوسط الحسابي للأعداد من 1 إلى 10 .

g) عدل البرنامج السابق لحساب وطباعة المتوسط الحسابي لعشرة أعداد يتم إدخالها من قبل المستخدم .

Q2- البرنامج التالي يقوم بإدخال عدد ساعات العمل وأجرة الساعة لـ 10 عمال في معمل نسيج ، لنفرض أن H هي عدد ساعات العمل وأن R هي أجرة الساعة الواحدة وأن W هو الأجر اليومي ويقوم بحساب وطباعة الأجر اليومي لكل عامل .

والمطلوب :

a) نفذ البرنامج التالي :

```
using System;
class Worker {
public static void Main(string[] args)
{
   double H, R, W;
   for (int i = 1; i <= 10; i++)  {
     Console.WriteLine(" Enter R & H ");
      R = double.Parse(Console.ReadLine());
       H = double.Parse(Console.ReadLine());
       W = R * H;
       Console.WriteLine(" W = {0} ", W);
```

```
    }//End for
  } // end method main
} // end class
```

b) عدل البرنامج السابق لطباعة الرواتب التي هي أكبر من 50000 .

c) عدل البرنامج السابق لطباعة الرواتب التي هي أقل من 50000 .

d) عدل البرنامج السابق لطباعة الرواتب التي هي أكبر من 50000 وأقل من 10000.

الأسئلة

١- اكتب برنامجاً يقوم بإدخال علامتي مقرر البرمجة (١) في النظري و العملي لطلاب السنة الأولى تقنيات حاسوب ؛ ثم يحسب ما يلي: عدد الطلاب الناجحين ، و عدد الطلاب الراسبين ، و يطبع النتائج على الشاشة. حيث درجة النظري (60/100) أما العملي فهي (40/100) ، بحيث إذا تجاوز المجموع عن 100يُعطي رسالة خطأ .

٢- اكتب برنامجا بلغة #C لطباعة الأشكال التالية باستخدام الحلقات (اكتب لكل شكل برنامج خاص به):

```
1                1
22              2
333            3
4444          4
55555        5
666666
```

```
1
22
333
4444
55555
```

الجلسة الخامسة

(بنى حلقة التكرار do/while & while)

الهدف من الجلسة :

١. التعرف على استخدام بنية الحلقة (while) .

٢. التعرف على استخدام بنية الحلقة (do/while) .

٣. استخدام العبارتان (break/continue) في حلقة التكرار (while).

برامج الجلسة :

Q1- البرنامج التالي يقوم بإدخال 10 درجات ويطبع درجات الطلاب الناجحين وكلمة ناجح Passed ، ويطبع درجات الطلاب الراسبين وكلمة راسب Failed .

والمطلوب :

a) نفذ البرنامج التالي :

```
using System ;
class Student {
public static void Main(string [ ] args )
{
 double d;
 int i=1;  // الابتدائية القيمة
 while(i <= 10)  // الشرط
  {
 Console.WriteLine(" Enter degree ");
 d = double.Parse( Console.ReadLine() );
  if ( d >= 60 )
 Console.WriteLine(" Passed = {0}" ,d);
```

```
 else
  Console.WriteLine(" Failed = {0}" ,d);
   i++;      // i = i + 1;  counter
  }//end while
 } // end method main
 } // end class Name
```

b) عدل البرنامج السابق باستخدام الحلقة do/while.

c) عدل البرنامج السابق لطباعة الطلاب الناجحين فقط والتي درجتهم أكبر من 80 .

d) ما الفرق بين حلقة التكرار while و do/while .

e) عدل البرنامج السابق لطباعة الطلاب الناجحين فقط و الخروج من الحلقة عند أول درجة أكبر أو تساوي 80 باستخدام العبارة break.

f) عدل البرنامج السابق لطباعة الطلاب الناجحين فقط ولا يطبع الطلاب الذين حصلوا على الدرجة 60 باستخدام العبارة continue في الحلقة .

==========

Q2- يقوم البرنامج التالي بإدخال عدد صحيح n موجب وحساب العاملي لهذا العدد n وطباعة النتيجة .

والمطلوب :

a) نفذ البرنامج التالي :

```
using System;
class Factorail  {
static void Main(string[] args)
{
  double n, Fact = 1;
```

```
  Console.WriteLine("Enter an positive integer to
                              find its factorial: ");
  n = double.Parse(Console.ReadLine());
  if (n < 0)
    Console.WriteLine( "Error: " + n + " is a
                                negative number" );
  else if ((n == 0) || (n == 1))
        Console.WriteLine( n + "! = " + Fact );
  else
  {
    int i = 1;
    while (i <= n)
     {
       Fact = Fact * i;
       i++;
     }
    Console.WriteLine( n + "! = " + Fact );
   }//end else
  }//end main
}//end class
```

Q2- يقوم البرنامج التالي بإدخال وحساب مجموع عدد غير محدد من الأرقام الصحيحة . وعند ادخال القيمة 0 (صفر) يشير إلى نهاية البرنامج .

والمطلوب :

a) نفذ البرنامج التالي :

```
using System ;
class Student {
public static void Main(string [ ] args ) {
    int data;
    int sum = 0;
    Console.WriteLine(" Enter the first No. ");
    data = int.Parse(Console.ReadLine());
```

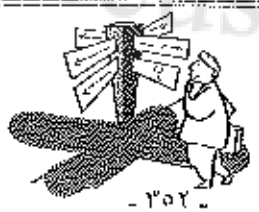
```
//--------  start while ---------------
 while(data != 0)  // الشرط
  {
      sum = sum + data;
 Console.WriteLine(" Enter integer No. ");
 data = int.Parse(Console.ReadLine());
 }// end while ---------------------

 Console.WriteLine(" Sum = {0}" ,sum);

  } // end method main
} // end class Name
```

b) عدل البرنامج السابق باستخدام الحلقة do/while.

الأسئلة

١. اكتب برنامجاً بلغة #C يقوم بإدخال 50 عدداً صحيحاً وحساب وطباعة المتوسط الحسابي للأعداد الصحيحة الفردية .

٢. اكتب برنامجاً بلغة #C يقوم بإدخال محرف من لوحة المفاتيح ، بحيث يستخدم حلقة التكرار while والتي من خلالها يُطلب من المستخدم الاستمرار بإدخال محرف ما وطباعته ويتم التوقف عن التكرار عندما يقوم المستخدم بإدخال المحرف Y أو المحرف N .



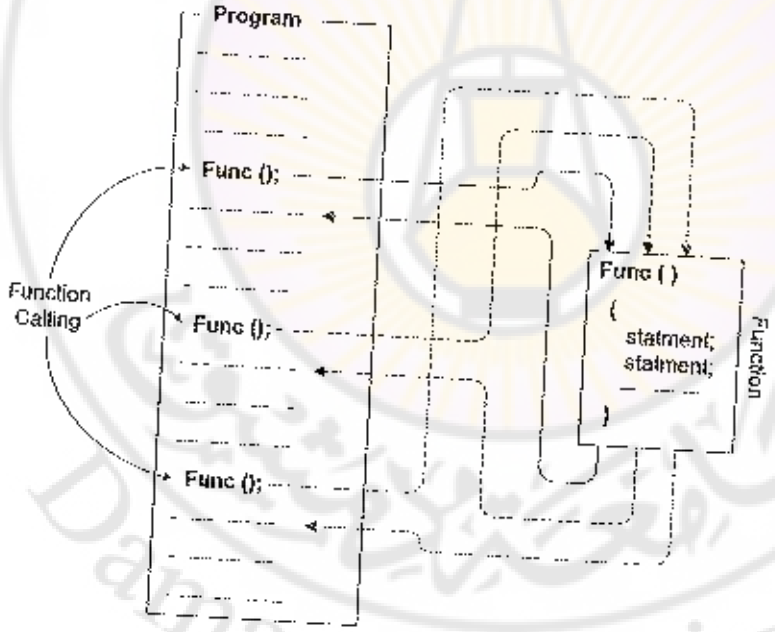
الجلسة السادسة

(التوابع (١))

الهدف من الجلسة :

١. التعرف على التوابع وكيفية تصريحها ، و تعريفها ، و طُرق استدعائها.
٢. التعرف على التوابع التي لا تعيد قيمة معطيات (أي استخدام void) .
٣. التدريب على استخدام التوابع الرياضية الموجودة في مكتبة #C .

الشكل التالي يبين كيفية انتقال التنفيذ من التابع الرئيسي ( )Main إلى التابع ( )Func



برامج الجلسة :

Q1 – يتضمن البرنامج التالي تابعاً يدعى myMethod لا يُعيد قيمة و الذي يقوم بطباعة العبارة Welcome to My Method عدداً معيناً من المرات .

والمطلوب :

أ- نفذ البرنامج التالي :

```
using System;
class Program  {
public static void myMethod()
{
   Console.WriteLine(" Welcome to My  Method ");
                    }
    public static void Main(string[] args)  {

        for ( int i=1 ; i<=10 ; i++ )
              myMethod();
          }//end main
       }// end class
```

Q2 - يستخدم البرنامج التالي تابعاً يدعى maximum والذي يرجع العدد الأكبر بين ثلاثة أعداد صحيحة. ويتم تمرير الأعداد كوسائط للتابع الذي يحدد العدد الأكبر بينهم ويرجعه للتابع Main باستخدام العبارة return ويتم تعيين القيمة التي تمت إعادتها إلى المتغير Max الذي تتم طباعته.

والمطلوب :

أ- نفذ البرنامج التالي :

```
using System ;

class Maximum {
public static void Main(string[] args ) {
  int  a, b, c;
  Console.WriteLine("Enter three integers: ") ;
     a = int.Parse( Console.ReadLine() );
     b = int.Parse( Console.ReadLine() );
     c = int.Parse( Console.ReadLine() );
Console.WriteLine(" maximum  is  {0} " ,
                                  maximum (a,b,c)) ;
}//end method main

public static int maximum(int x, int y, int z)
{
    int max = x;
    if (y > max)
        max = y;
    if (z > max)
        max = z;
    //Continued
    return max;
}//end max
} // end class Name
```

==========

ب- أجري عملية التحميل الزائد للتابع maximum بحيث يقوم بحساب العدد الأكبر لثلاثة أعداد حقيقة ، ثم أجري عملية التحميل الزائد مرة أخرى للتابع maximum بحيث يقوم بحساب المحرف الأكبر لثلاثة محارف .

==========

جـ- اكتب برنامجاً بلغة #C يحتوي على تابع اسمه Fact لحساب n! (حيث n عدد صحيح يتم إدخاله من قبل المستخدم .

Q2- البرنامج التالي يستخدم التوابع الرياضية ضمن الصف Math ، يقوم بحساب :

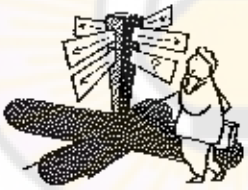
- القيمة المطلقة
- الجذر التربيعي
- الرفع إلى قوة
- التوابع المثلثية
- التابع الأسي واللوغارتمي

والمطلوب :

أ- نفذ البرنامج التالي :

```
using System ;
class math1 {
public static void Main(string[] args ) {
 Console.WriteLine(" The pi number = "+Math.PI);
 Console.WriteLine(" Absolute of -15 is: = "+
                                Math.Abs(-15));
 Console.WriteLine(" Sin(90) = "+
                                Math.Sin((Math.PI)/2));
 Console.WriteLine(" Cos(90) = "+
                                Math.Cos((Math.PI)/2));
 Console.WriteLine(" Smallest number = "+
                                Math.Ceiling(9.3));
```

```
Console.WriteLine(" Rounding value of 10.5 is = "+
                        Math.Round(10.5));
Console.WriteLine(" The Root Sguare of 36 = "+
                        Math.Sqrt(36));
Console.WriteLine("The e number ="+ Math.Exp(9)   );
Console.WriteLine(" Largest Number = "+
                        Math.Floor(9.3));
Console.WriteLine(" 2 Raised to the power of 4 = "+
                        Math.Pow(2,4));
Console.WriteLine(" log(1000) = "+Math.Log10(1000));
 } // end method main
} // end class math
```



```
    else
        return  n * fact(n-1);
}//End fact

public static int sum(int n)
{
  if (n == 1)
     return 1;
  else
     return n + sum(n - 1);
}//End sum
} // end class
```

ب-أدخل الأعداد من 1 حتى 20 ، وأحسب عاملي هذه الأعداد ؟ ماذا تلاحظ ؟ كيف يتم معالجة هذا الأمر ؟

Q2 – المثال التالي يوضح :

- ✓ توليد رقم عشوائي من 1 حتى 7 يدل على أيام الأسبوع .
- ✓ توليد رقم عشوائي من 1 حتى 12 يدل على اسم الشهر .
- ✓ بينما العام يبقى ثابت على 2017 .

وذلك باستخدام الصف Random لتوليد أعداد عشوائية .

والمطلوب :

أ- نفذ البرنامج التالي :

```
using System;
class RandIntegers {
 public static void Main(string[] args)
```

```
{
 // random number generator
  Random rnd = new Random();
  Random rnd1 = new Random();
  int rNum;
  int rNum1;
  rNum = rnd.Next(1, 8);
  rNum1 = rnd1.Next(1, 13);
  DateTime dt = Convert.ToDateTime(rNum +
                                  "/" + rNum1 + "/ 2017");
   // display generated value
  Console.WriteLine(dt);
   // display generated value
Console.WriteLine(dt.ToString("MMMM"));
switch (rNum)
{
  case 1:
        Console.WriteLine("Sunday");
        break;
  case 2:
        Console.WriteLine("Monday");
        break;
  case 3:
        Console.WriteLine("Tuseday");
        break;
  case 4:
        Console.WriteLine("Wednseday");
        break;
  case 5:
        Console.WriteLine("Thursday");
         break;
  case 6:
        Console.WriteLine("Friday");
         break;
```

الأسئلة

١- اكتب برنامجاً بلغة #C يحتوي على تابعين : التابع الأول اسمه Factorial مهمته حساب العاملي (!) لأي عدد صحيح موجب ، أما الثاني اسمه SumFactorial مهمته حساب المقدار التالي :

$$sum = n! + \frac{n!}{x!} - m!$$

التابع Main يستدعي كلاً من التابعين وطباعة قيمة المقدار Sum (التابع SumFact يستدعي التابع Factorial) .

٢- اكتب برنامجاً بلغة #C يقوم بما يلي : إدخال قيم المتحولات a,b,x وهي من النمط double ويحتوي البرنامج على تابعين :

✓ التابع الأول اسمه FunctionY مهمته حساب المقدار التالي :

$$y = x^2 + 2x + 10$$

✓ أما التابع الثاني اسمه FunctionZ مهمته حساب المقدار Z والمعرف كما يلي :

$$Z = (y-a)^2 + (y+b)^2$$

التابع Main يستدعي كلاً من التابعين لطباعة قيمة Z (التابع FunctionZ يستدعي التابع FunctionY) .

٣- أكتب برنامجاً بلغة #C يقوم بما يلي:

- إدخال أي عدد صحيح x.

- يستخدم تابعاً اسمه fun_prim لإيجاد قواسم العدد المدخل x .
- يختبر البرنامج إذا كان العدد المدخل x أولياً أو غير أولياً وذلك بناءً على خرج التابع fun_prim, وطباعة النتيجة في التابع Main

الجلسة الثامنة
(المصفوفات (1))

الهدف من الجلسة :

١. التعرف على المصفوفات أحادية البعد وكيفية إنشائها .
٢. التعرف على اسناد قيم ابتدائية للمصفوفة .
٣. التعرف على كيفية الادخال والإخراج للمصفوفات .
٤. التعرف على ترتيب المصفوفة وعكس عناصرها .

برامج الجلسة :

Q1– البرنامج التالي يقوم بإسناد قيم ابتدائية للمصفوفة a[12] وحساب وطباعة مجموع عناصر المصفوفة total .

والمطلوب :

أ- نفذ البرنامج التالي :

```
// compute the sum of the elements of the array using System;

class ArrayProg  {
   public static void Main(string[] args)
   {
     const int size = 12;
     int[] a = new int[size] { 1, 3, 5, 4, 7, 2,
                               99, 16, 45, 67, 89, 45 };
     int total = 0;
     for (int i = 0; i < a.Length; i++)
           total = total + a[i];
     Console.WriteLine(" total of array element
                                values is " + total);

   } // end method main
} // end class Name
```

ب-أعد كتابة البرنامج السابق لإدخال عناصر المصفوفة من قبل المستخدم .

ت- أعد كتابة البرنامج السابق لحساب وطباعة المتوسط الحسابي لعناصر المصفوفة بالإضافة لحساب وطباعة مجموع عناصر المصفوفة .

ث-أعد كتابة البرنامج السابق لطباعة عناصر المصفوفة بشكل مرتب باستخدام التابع

Array.Sort(src)

ج-أعد كتابة البرنامج السابق لطباعة عناصر المصفوفة بشكل معكوس باستخدام التابع

Array.Reverse(src)

Q2 - البرنامج التالي يستخدم الصف العشوائي Random لتوليد أسماء 12 شهراً بشكل عشوائي ضمن مصفوفة nameMonth والمطلوب :

١. أنشأ تابعاً اسمه PrintMyArray لطباعة عناصر المصفوفة nameMonth باستخدام الحلقة foreach أو for .
٢. أدخل عناصر المصفوفة nameMonth عن طريق توليد أسماء الأشهر باستخدام الصف Random .
٣. استخدم الطريقة Sort( ) لترتيب عناصر المصفوفة .
٤. استخدم الطريقة Reverse من أجل عكس عناصر المصفوفة .

والمطلوب :

أ- نفذ البرنامج التالي :

```
using System;
class ArrayProg  {
 static void PrintMyArray(string[] nameMonth)
  {
    foreach (string month in nameMonth)
     Console.WriteLine("The Month is : {0} ", month);

  }//end PrintMyArray
  static void Main(string[] args)
  {
     string[] nameMonth= new string[12];
     Random rnd = new Random();
     int nrt ;
     for (int i = 0; i < nameMonth.Length; i++)
     {
        nrt = rnd.Next(1, 13);
  DateTime dt=Convert.ToDateTime(nrt+"/"+nrt+"/2017");

        nameMonth[i] = dt.ToString("MMMM"); ;
     } //end for
        Console.WriteLine(" the Month is ");
        PrintMyArray(nameMonth);

     } // end method main
  } // end class
```

==========

ب-أعد تنفيذ البرنامج عدداً من المرات وسجل عناصر المصفوفة ، ماذا تلاحظ ؟ علل؟

ت-أعد كتابة البرنامج السابق من أجل طباعة المصفوفة بشكل مرتب باستخدام الطريقة Sort( ) .

ث- أعد كتابة البرنامج السابق من أجل طباعة المصفوفة بشكل معكوس باستخدام الطريقة Reverse .

==========

Array index	دليل المصفوفة
Array Name	اسم المصفوفة
Array Of Pointers	مصفوفة المؤشرات
Arrow Keys	مفاتيح الأسهم
Assemblers	المجمعات
Assembly Language	لغة التجميع
Assignment	إسناد
Assignment Statement	تعليمة الإسناد
Average	معدل
B	
Block of statements	كتلة من التعليمات
Blocks	كتل
Body of the function	جسم التابع
Boolean Expression	تعبير منطقي
Branched Flowchart	المخطط التدفقي التفرعي
Byte	بايت
C	
Cache Memory	ذاكرة الكاش
Calling	استدعاء
Calling Function	استدعاء التابع
Central Processing Unit (CPU)	وحدة المعالجة المركزية

Character	محرف (حرف أو رقم أو رمز)
Character Type	نوع المحرف
Characters and Strings	المحارف وسلاسل المحارف
Circle	دائرة
Classes	الصفوف
Combined Programming Language CPL	لغة البرمجة المجمعة
Comments	تعليقات
Complex	عقدي
Compound Assignation Operators	مؤثرات الإسناد المركبة
Compute	يحسب
Computer	حاسوب
Computer Performance	أداء الحاسوب
Condition	شرط
Conditional Operators	العوامل الشرطية
Constant	ثابت
Constant member function	تابع عضو ثابت
Control and Looping Structures	بنى التحكم والتكرار
Control Unit	وحدة التحكم
Cost	التكلفة
Counter	عداد
Counting Algorithm	خوارزمية العد
Compiler	مترجم

CPU speed	سرعة وحدة المعالجة
D	
Data Members	المعطيات الأعضاء
Data Processing	معالجة المعطيات
Data Structures	بنى المعطيات
Data type	نوع المعطيات
Decision	اتخاذ القرار
Declaration	تصريح
Declaration of Variable	التصريح عن المتغير
Declaring Function	التصريح عن التابع
Decrement	النقصان
Default Arguments	الوسطاء الافتراضية
Defining Function	تعريف التابع
Downloaded Program	برنامج التحميل
Draw	يرسم
Dynamic RAM(DRAM)	ذاكرة ديناميكية
E	
Empty	خالي
Erasable PROM (EPROM)	الذاكرة المبرمجة القابلة للمحي
Escape Characters	محارف الهروب
Example	مثال

```
using System;
class ArrayProg  {
 static void PrintMyArray(string[] nameMonth)
  {
    foreach (string month in nameMonth)
     Console.WriteLine("The Month is : {0} ", month);

  }//end PrintMyArray
  static void Main(string[] args)
  {
     string[] nameMonth= new string[12];
     Random rnd = new Random();
     int nrt ;
     for (int i = 0; i < nameMonth.Length; i++)
     {
        nrt = rnd.Next(1, 13);
 DateTime dt=Convert.ToDateTime(nrt+"/"+nrt+"/2017");

        nameMonth[i] = dt.ToString("MMMM"); ;
     } //end for
        Console.WriteLine(" the Month is ");
        PrintMyArray(nameMonth);

    } // end method main
} // end class
```

ب-أعد تنفيذ البرنامج عدداً من المرات وسجل عناصر المصفوفة ، ماذا تلاحظ ؟ علل؟

ت-أعد كتابة البرنامج السابق من أجل طباعة المصفوفة بشكل مرتب باستخدام الطريقة Sort( ) .

ث- أعد كتابة البرنامج السابق من أجل طباعة المصفوفة بشكل معكوس باستخدام الطريقة Reverse .

الأسئلة

١- اكتب برنامجاً بلغة #C لتعريف مصفوفة أحادية البعد من نوع int عدد عناصرها 20 والمطلوب :

a. إدخال قيم عناصر المصفوفة من قبل المستخدم .

b. إيجاد أكبر عناصرها و دليله .

c. حساب وطباعة متوسط الأعداد الفردية على الشاشة.

٢- اكتب برنامجاً بلغة #C يقوم بما يلي :

a. يدخل 20 عدداً حقيقياً ويخزنها في مصفوفة أحادية البعد A[20]. ويحتوي البرنامج على تابعين:

b. التابع الأول اسمه MaxArry( ) مهمته حساب العنصر الأكبر للمصفوفة . أما التابع الثاني اسمه EventArry مهمته حساب مجموع الأعداد الزوجية في المصفوفة .

c. التابع main يستدعي التابعين ويطبع العنصر الأكبر ومجموع الأعداد الزوجية على الشاشة .

الجلسة التاسعة

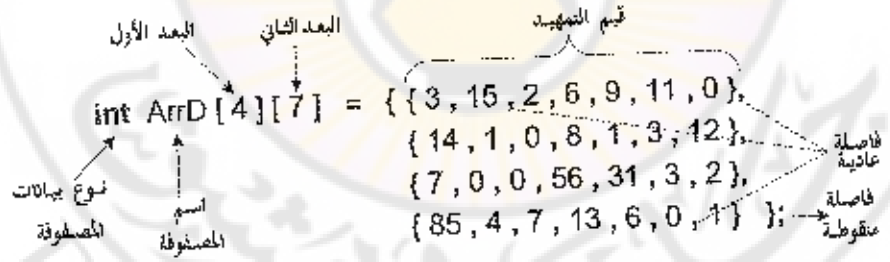
(المصفوفات (2))

الهدف من الجلسة :

١. التعرف على المصفوفات ثنائية البعد وكيفية إنشائها .
٢. التعرف على اسناد قيم ابتدائية للمصفوفة .
٣. التعرف على كيفية الادخال والإخراج للمصفوفات .
٤. التعرف على ترتيب المصفوفة وعكس عناصرها .

بنية المصفوفة ثنائية البعد :

- لنأخذ المصفوفة (ArrD) ثنائية البعد ؛ وهي مصفوفة أعداد صحيحة ثنائية البعد بُعديها (7 × 4).



التركيب النحوي للمصفوفة

Index	0	1	2	3	4	5	6
0	3	15	2	6	9	11	0
1	14	1	0	8	1	3	12
2	7	0	0	56	31	3	2
3	85	4	7	13	6	0	1

شكل توضيحي لعناصر المصفوفة الثنائية (ArrD) ، و أدلة هذه العناصر

برامج الجلسة :

Q1– البرنامج التالي يقوم بإسناد قيم ابتدائية للمصفوفتين array1[2,3] و array2[2,2] وطباعة عناصرهما على مصفوفة ثنائية البعد كما هو موضح بالبرنامج .

والمطلوب :

أ- نفذ البرنامج التالي :

```
//initializing multidimensional arrays
using System;
namespace   Example56}
class DimArry0}
public static void Main(string [ ] args ) }
int [ , ]array1;{ {6 ,5 ,4} ,{3 ,2 ,1}  } =
int[ , ] array2 ;{  {  8 , 4 } ,{ 2 ,1 } } =
Console.WriteLine("values in array1 by row are : ");
for (int i=0; i< 2 ; i++)}
 for (int j=0; j< 3; j++)
Console.Write(array1[i,j] +" ");
Console.WriteLine();  } //end for
```

```
Console.WriteLine("values in array2 by row are : ");
  for (int i=0; i< 2 ; i++) {
  for (int j=0; j< 2; j++)
 Console.Write(array2[i,j] +" ");
 Console.WriteLine();  } //end for
}//End main
} // end class Name
 } //end namespace
```

```
values in array1 by row are
1 2 3
4 5 6
values in array2 by row are :
1 2
3 4
```

أ- أعد كتابة البرنامج السابق لإدخال عناصر المصفوفتين من قبل المستخدم

ب- أعد كتابة البرنامج السابق لحساب وطباعة المتوسط الحسابي لعناصر المصفوفة array1 بالإضافة لحساب وطباعة مجموع عناصر المصفوفة array2.

Q2- البرنامج التالي يوضح كيفية ما يلي :

- اسناد قيم ابتدائية لمصفوفة منتظمة array2 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .
- اسناد قيم ابتدائية لمصفوفة غير منتظمة array1 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .
- استخدام الصنف Random لتوليد قيم عشوائية واسنادها للمصفوفة المنتظمة array1 وطباعة عناصرها واستخدام Length للوصول لحجم المصفوفة .
- الخرج يجب أن يكون على شكل مصفوفة ثنائية البعد .

والمطلوب نفذ البرنامج التالي :

```
using System ;
class DimArry0 {
public static void Main(string [ ] args ) {
int [ ][]array1 =new int[2][];
    array1[0]=new int[3] {1, 2, 3};
    array1[1]=new int[2]  {4, 5} ;
int [ , ]array2 = { { 1, 2 }, { 4 , 8  }  };

Console.WriteLine("values in array1 by row are : ");
for (int i = 0; i < array1.Length; i++) {
    for (int j = 0; j < array1[i].Length; j++)
        Console.Write(array1[i][j] + " ");
        Console.WriteLine();
} //end for
```

```
Console.WriteLine("values in array2 by row are : ");
  for (int i=0; i< 2 ; i++) {
  for (int j=0; j< 2; j++)
 Console.Write(array2[i,j] +" ");
 Console.WriteLine();
} //end for
Console.WriteLine("values in NEW array1by row are: ");
  Random rnd = new Random();
  for (int i = 0; i < array1.Length; i++)
      for (int j = 0; j < array1[i].Length; j++)
          array1[i][j] = rnd.Next(1, 13);
  for (int i = 0; i < array1.Length; i++) {
      for (int j = 0; j < array1[i].Length; j++)
          Console.Write("{0,2} ", array1[i][j]);
         Console.WriteLine();
  }
}//End main
} // end class Name
```

```
Values in array1 by row are :
1 2 3
4 5
Values in array2 by row are :
1 2
4 8
Values in NEW array1 by row are :
1 7 4
1 5
Press any key to continue . . .
```

Q3- البرنامج التالي يوضح كيفية المصفوفات كوسائط للتوابع :

- إدخال 12 عدداً صحيحاً ، ثم يتم تخزينها في مصفوفة ثنائية البعد a[3,4] .
- طباعة عناصر المصفوفة a .
- حساب المتوسط الحسابي لعناصر المصفوفة وطباعته .

- **استخدم التوابع في البرنامج كما يلي :**
  - التابع ( )**printArray** لطباعة عناصر المصفوفة .
  - التابع ( )**printArray1** لحساب مجموع عناصر المصفوفة وحساب المتوسط الحسابي .

والمطلوب نفذ البرنامج التالي :

```
// multidimensional arrays
 using System ;
 namespace   Example59 {
class DimArry2 {
public static void Main(string [ ] args ) {
int [ , ] Arry;
Arry = new int[3,4] ;
 Console.WriteLine("Enter Arry [ , ] values  :  ");
   for (int  i=0; i<3; i++ )
   for (int  j=0; j<4; j++ )
{
     int x = int.Parse( Console.ReadLine() );
   Arry[i,j]=x;
   }
```

```
   printArray(Arry);
   printArray1(Arry);
   }//End main
public static void printArray ( int [ , ] a) {
  for (int i=0; i<3; i++){
  for (int j=0; j<4; j++)
Console.Write ( a[ i , j ] +"  " );
 Console.WriteLine();}
}//End printArray
public static void printArray1(int [ , ] a) {
   int sum=0;
    for (int i=0; i<3; i++)
    for (int j=0; j<4; j++)
       sum = sum + a[ i , j ] ;
    double  avr = sum/12;
  Console.WriteLine(" avr = "+  avr );
   }//End printArray1
 } // end class Name
  } //end namespace
```

أ- عدل البرنامج السابق بحيث يقوم التابع printArray بحساب وطباعة أكبر عدد في المصفوفة .

ب- عدل البرنامج السابق بحيث يقوم التابع printArray1 بحساب وطباعة المتوسط الحسابي للأعداد الزوجية للمصفوفة .

الأسئلة

١. اكتب برنامجاً يُعرّف و يُسند قيم ابتدائية لمصفوفة أعداد صحيحة ثنائية البعد ، بُعديها [3×4] ؛ ثم يطبعها على الشاشة. و يُعرّف تابع يقوم بحساب نسبة (مجموع عناصر أحد الأسطر من هذه المصفوفة) على (مجموع كل عناصر المصفوفة) ؛ استدعي هذا التابع في البرنامج و اطبع النتائج على الشاشة.

==========

٢. أكتب برنامجاً بلغة #C يقوم بما يلي:

a. يقوم بإدخال 9 أعداد صحيحة وتخزينها في مصفوفة ثنائية البعد x[3,3] .

b. طباعة عناصر المصفوفة x[3,3] على شكل مصفوفة مربعة

c. يستخدم تابع من النوع void لإيجاد كافة الأعداد الأولية من بين عناصر المصفوفة المُدخلة وطباعتها وطباعة عددها.

d. التابع Main يستدعي التابع لطباعة النتائج .

==========

المراجع الأجنبية

References

1. Beginning Visual C#® 2015 Copyright © 2016 by John Wiley & Sons, Inc., Indianapolis, Indiana, IN 46256, Inc. 10475 Crosspoint Boulevard .
2. Beginning C# Object Oriented Programming,by Dan Clark , New Jersey 07458,400 p.
3. C# 6.0 in a Nutshell by Joseph Albahari and Ben Albahari Copyright c 2016 Joseph Albahari and Ben Albahari. All rights reserved.Printed in the United States of America.
4. FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C# by Dilyan Dimitrov &Hristo Germanov & etc. Iliyan Murdanliev Svetlin Nakov & Co., 2013.
5. Object Oriented Programming using C#,by Dilyan Dimitrov ,Hristo Germanov ,Iliyan Murdanliev,2011,Semon Kendal & ventus Publeshing Aps ISBN 978-7681-814-2.
6. Visual C#® 2012 **C HOW TO PROGRAM**". Fifth Edition DEITEL P.J ,DEITEL H.M by Pearson Education, Inc.Upper Saddle River, New Jersey, ISBN-10: 0-13-337933-7,1020 p.

المراجع العربية

١. مدخل إلى الحاسوب والبرمجة ، د.مأمون يونس ن د.جمال الياسين و د.أكرم مذكور ، جامعة دمشق ،سوريا، 2016 .

٢. البرمجة (١) البرمجة غرضية التوجه بلغة C++ ، د. مأمون يونس و د.أكرم مذكور ، جامعة دمشق ،سوريا ، 2014 .

٣. البرمجة (٢) البرمجة بلغة Java، د. مأمون يونس و د.أكرم مذكور ، جامعة دمشق ، سوريا ،2018 .

٤. البرمجة بلغة C# Visual 2008 ، م.يمان البني و أ.حسام برهان ، دار البراق ،حلب ، سوريا ، 2008 .

٥. الوحدات المحيطية للحاسوب – الدكتور مأمون يونس & الدكتور رزق غانم – جامعة دمشق – سوريا – 2011 .

==========

==========

المصطلحات العلمية

إنكليزي-عربي

English-Arabic

A	
Absolute value	القيمة المطلقة
Access	برنامج قواعد المعطيات
Algorithm	الخوارزمية
Algorithm of the Problem	خوارزمية المسألة
Allocate	يخصص
Allocation error	خطأ التخصيص
American National Standard Institute(ANSI)	المعهد الوطني الأمريكي للمعايير القياسية
Application Programming Interface	واجهة برمجة التطبيقات
Application Server	مخدم تطبيقات
Application Software	البرمجيات التطبيقية
Area	مساحة
Arg - List	قائمة الوسطاء
Argument	وسيط
Arithmetic and Logic Unit(ALU)	وحدة الحساب والمنطق
Arithmetic Expression	تعبير حسابي
Arithmetic Operators	العوامل الحسابية
Arithmetic Statement	تعليمة حسابية
Array	مصفوفة
Array Definition	تعريف المصفوفة

Array index	دليل المصفوفة
Array Name	اسم المصفوفة
Array Of Pointers	مصفوفة المؤشرات
Arrow Keys	مفاتيح الأسهم
Assemblers	المجمعات
Assembly Language	لغة التجميع
Assignment	إسناد
Assignment Statement	تعليمة الإسناد
Average	معدل
B	
Block of statements	كتلة من التعليمات
Blocks	كتل
Body of the function	جسم التابع
Boolean Expression	تعبير منطقي
Branched Flowchart	المخطط التدفقي التفرعي
Byte	بايت
C	
Cache Memory	ذاكرة الكاش
Calling	استدعاء
Calling Function	استدعاء التابع
Central Processing Unit (CPU)	وحدة المعالجة المركزية

Character	محرف (حرف أو رقم أو رمز)
Character Type	نوع المحرف
Characters and Strings	المحارف وسلاسل المحارف
Circle	دائرة
Classes	الصفوف
Combined Programming Language CPL	لغة البرمجة المجمعة
Comments	تعليقات
Complex	عقدي
Compound Assignation Operators	مؤثرات الإسناد المركبة
Compute	يحسب
Computer	حاسوب
Computer Performance	أداء الحاسوب
Condition	شرط
Conditional Operators	العوامل الشرطية
Constant	ثابت
Constant member function	تابع عضو ثابت
Control and Looping Structures	بنى التحكم والتكرار
Control Unit	وحدة التحكم
Cost	التكلفة
Counter	عداد
Counting Algorithm	خوارزمية العد
Compiler	مترجم

CPU speed	سرعة وحدة المعالجة
D	
Data Members	المعطيات الأعضاء
Data Processing	معالجة المعطيات
Data Structures	بنى المعطيات
Data type	نوع المعطيات
Decision	اتخاذ القرار
Declaration	تصريح
Declaration of Variable	التصريح عن المتغير
Declaring Function	التصريح عن التابع
Decrement	النقصان
Default Arguments	الوسطاء الافتراضية
Defining Function	تعريف التابع
Downloaded Program	برنامج التحميل
Draw	يرسم
Dynamic RAM(DRAM)	ذاكرة ديناميكية
E	
Empty	خالي
Erasable PROM (EPROM)	الذاكرة المبرمجة القابلة للمحي
Escape Characters	محارف الهروب
Example	مثال

F	
Factor	مُعامل
Factorial	العاملي (!)
False	خطأ
File	ملف
File system	نظام الملفات
Final value	القيمة النهائية
First	الأول
Flags	مؤشرات
Flash Memory	الذاكرة غير المتطايرة(الدائمة)
Floppy Disks	الأقراص المرنة
Flow line	اتجاه سير العمليات
Flowchart	المخطط التدفقي
Flowchart Method	طريقة المخططات التدفقية
FORTRAN	لغة فورتران
Fraction Part	الجزء الكسري
Function	تابع
Function	تابع (دالة)
Function name	اسم التابع
Function Keys	مفاتيح الوظائف
Function Type	نوع التابع

H	
Hard Disk	القرص الصلب
Hardware	المكونات المادية
Header File	ملف رأسي
Hexadecimal Number System	نظام العد السداسي عشر
High Level Programming Languages	اللغات البرمجية العالية المستوى
Hub	الموزع
G	
GBR	سجلات الأغراض العامة
General Purpose Computers	حواسيب الأغراض العامة
Global	شمولي(عام)
Global Variable	متغير عام
Graphic Card	بطاقة الرسوم
Graphical User Interface	المواجهة الرسومية
Graphics	رسومات
I	
I/O Functions	توابع الإدخال والإخراج
IC	الدارة المتكاملة

Identifiers	المعرفات
Increment	الزيادة
Increment and Decrement Operators	مؤثرات الزيادة والنقصان
Independent	مستقل
Information Management Programs	برمجيات إدارة المعلومات
Initial Value	القيمة الابتدائية
Initialization	تهيئة
Inline Functions	التوابع الفورية(السطرية)
Inner Loop	الحلقة الداخلية
Input Data	إدخال المعطيات
Input Statement	تعليمة الإدخال
Input Units	وحدات الإدخال
Input/Output Management Programs	برمجيات وحدات الإدخال والإخراج
Insertion	إدراج
Insertion Operator	عملية الإدراج
Institute of Electrical And Electronic Engineers(IEEE)	معهد مهندسي الكهرباء والإلكترون
Instruction	التعليمات
Instruction Unit	وحدة التعليمات
Integer	صحيح
Integer Constant	ثابت صحيح
Integer Variable	متغير صحيح

Integer Part	الجزء الصحيح
Interface	واجهة تخاطب
Interface Unit	وحدة الترابط مع الحاسوب
Internal	داخلي
International Standard Organization(ISO)	المنظمة الدولية للمواصفات والمقاييس
Internet Explorer	متصفح الانترنت
Interpreters	المفسرات
J	
Joy stick	عصا التحكم بالألعاب
K	
Kernel	النواة
Keyword	كلمة محجوزة
L	
Language Method	الطريقة اللغوية
Laptop	الحاسوب المحمول
Last	الأخير
Least significant Digit(LSD)	خانة الدلالة الدنيا
Length	طول
Linear search	البحث الخطي
Linking	ربط

Liquid Crystal Display (LCD)	شاشة السائل البلوري
List of Variable	قائمة المتغيرات
Loading	تحميل
Local Variable	متغير محلي
Logic Error	خطأ منطقي
Logical Operators	العوامل المنطقية
Logical Constant	ثابت منطقي
Logical Expression	تعبير منطقي
Logical Variable	متغير منطقي
Loop	حلقة تكرار
Loop Flowchart	المخطط التدفقي الحلقي
Low Level Programming	اللغات البرمجية المتدنية المستوى
M	
Machine Language	لغة الآلة
Main Memory	الذاكرة الرئيسي
Main Program	البرنامج الرئيسي
Manipulator	معالج
Math Library Functions	مكتبة الدوال الرياضية
Mathematics	الرياضيات
Matrixes	مصفوفات

Member	عضو
Member functions	التوابع الأعضاء
Memory Management Programs	برمجيات إدارة الذاكرة الرئيسية
Menu Interface	المواجهة بالقوائم
Microcomputer	الحاسوب الميكروي
Microelectronics	الإلكترونيات الصغيرة
Microprocessor	معالج ميكروي
Modules	وحدات(أجزاء)
Monitor	جهاز العرض(شاشة)
Month	شهر
Most Significant Digit(MSD)	خانة الدلالة العليا
Mother Board	اللوحة الأم
Mouse	الفأرة
Multiprocessing Systems	نظم المعالجة المتعددة
Multiprogramming Systems	نظم البرمجة المتعددة
Multi-Tasking	أنظمة متعددة المهام
Multi-user	أنظمة متعددة المستخدمين
N	
Name	الاسم
Names mangling	تشويه الأسماء
Naming of variables	تسمية المتغيرات

Negation	النفي
Negative	سالب
Nested Loop Flowchart	المخططات التدفقية الحلقية المتداخلة
Nested loops	الحلقات المتداخلة
Network Interface Card	بطاقة الشبكة
Number	رقم
Number Keys	مفاتيح الأرقام
Number Systems	أنظمة العد
Numerical Constant	الثابت العددي
Numerical Variable	المتغير العددي
O	
Object	غرض (هدف)
Object Oriented Programming	البرمجة غرضية التوجه
Object program	البرنامج الهدف
Octal Number System	نظام العد الثماني
One-Dimensional Array	مصفوفة أحادية البعد
Operator	عملية (مؤثر)
Outer Loop	الحلقة الخارجية
Output Statement	تعليمة الإخراج
Output Units	وحدات الإخراج

Root	جذر
S	
SBR	سجلات الأغراض الخاصة
Screen	الشاشة
Second	ثانية
Secondary Storage Units	وحدات التخزين الثانوية
Serial Port	البوابة التسلسلية
Shell	الغلاف
Show	يعرض
Silicon	السيليكون
Simple sequential Flowchart	المخطط التدفقي التتابعي البسيط
Single-Tasking	أنظمة وحيدة المهام
Single-user	أنظمة وحيدة المستخدم
Software	البرمجيات
Solution	حل
source program	البرنامج المصدر
Speakers	السماعات
Special Keys	مفاتيح الحاسوب الخاصة
Standard	معياري (قياسي)
Start	البداية
Starting address	عنوان البداية

Statement	تعليمة
Static	ساكن
Static RAM (SRAM)	ذاكرة ساكنة
Stop	النهاية
String Constant	الثابت من نوع سلاسل المحارف
String Variables	المتغيرات من نوع سلاسل المحارف
Structure	بنية
Structured Language	اللغة البنيوية
Structured Programming	البرمجة البنيوية(الهيكلية)
Subdirectories	الأدلة الفرعية
Summers Algorithm	خوارزمية التجميع
Switches	مفاتيح الكترونية
Symbols	رموز
Syntax analysis	التحليل النحوي واللغوي
System Board	لوحة النظام
System File Protection	حماية ملفات النظام
T	
Tables	جداول
Task Manager	إدارة المهام
Touch Screen	شاشة اللمس
Tow-Dimensional Array	مصفوفة ثنائية البعد

Typing Keys	مفاتيح الأحرف والرموز
U	
Ultraviolet Light	الأشعة فوق البنفسجية
Universal Serial Bus	المسرى التسلسلي العام
Using Pointer To Objects	استخدام المؤشر إلى الأغراض
Utilities Programs	البرامج الخدمية
V	
Vacuum Tubes	الصمامات المفرغة
Variable	متغير (متحول)
Video Display unit	وحدة العرض المرئي
Virtual Memory	الذاكرة الافتراضية
W	
Wire Communications	وسائط الاتصال السلكية
Word	برمجيات معالجة النصوص
Word size	عرض الكلمة
World Wide Web(WWW)	الشبكة العنكبوتية العالمية
Write Time	زمن الكتابة

اللجنة العلمية :

الدكتور جمال الياسين

الدكتور سمير كرمان

الدكتور رزق غانم

المدقق اللغوي :

الدكتور دياب راشد