جامعة المنوفية
كلية التربية النوعية
قسم تكنولوجيا التعليم والحاسب الآلي

محاضرات فى مقرر

الحاسب الآلى

(بيسيك متقدم)

لطلاب الفرقة الثانية بقسم تكنولوجيا التعليم

إعداد

قسم تكنولوجيا التعليم والحاسب الآلى

لعام ٢٠١٩/٢٠٢٠

نموذج رقم (١٢)

جامعة / أكاديمية: المنوفية

كلية / معهد: التربية النوعية

قسم: تكنولوجيا التعليم والحاسب الآلي

توصيف مقرر دراسي

٢٠٢٠/٢٠١٩

١- بيانات المقرر		
الرمز الكودي :	**اسم المقرر : الحاسب الآلي بيسك (متقدم)**	**الفرقة / المستوى : الثانية**
التخصص : تكنولوجيا التعليم والحاسب الآلي	**عدد الوحدات الدراسية : ٦**	**نظري** ٢ **عملي** ٤

٢- هدف المقرر	١/٢ المعرفة بالمفاهيم النظرية للمشروع المصمم بلغة البرمجة. ٢/٢ اكتساب المعارف والمهارات المرتبطة بتحليل المشروع المصمم بلغة البرمجة. ٣/٢ التخطيط لكيفية تنفيذ مشروع مصمم بلغة البرمجة. ٤/٢ تنفيذ المشروع المصمم بلغة البرمجة بالاشتراك مع فريق العمل. ٥/٢ تقييم المشروع النهائي المصمم بلغة البرمجة بطريقة فعَّالة. ٦/٢ ربط المشروع النهائي المصمم بلغة البرمجة باوامر التفرع والتكرار. ٧/٢ مشاركة المشروع النهائي المصمم بلغة البرمجة مع الأقران.
٣- المستهدف من تدريس المقرر :	**طلاب الفرقة الثانية بقسم تكنووجيا التعليم والحاسب الآلي**

٣-أ- المعلومات والمفاهيم	٣-أ-١يعرف المفاهيم الأساسية فى لغة البرمجة فيجوال بيسك. نت داخل شاشة Windows Application. ٣-أ-٢ يشرح وظيفة كل أداة من أدوات لغة البرمجة فيجوال بيسك. نت. ٣-أ-٣ يحدد مكونات الشاشة الرئيسية لبرنامج الفيجوال بيسك. نت. ٣-أ-٤ يفرق بين الخصائص المشتركة والخصائص الخاصة بالأدوات . ٣-أ-٥ يوضح نظام الرسوم GDI+ وأوامرها. ٣-أ- ٦ يعرف لغة c++ ومكوناتها. ٣-أ-٧ يعرف خطوات ربط لغة c++ باوامر التفرع والتكرار.
٣-ب- المهارات الذهنية	٣-ب-١ يحلل مشكلات متعددة من أجل تصميم برامج لحلها بلغة البرمجة. ٣-ب-٢ يقارن بين المربعات الحوارية في لغة البرمجة. ٣-ب-٣ يستنتج طرق مختلفة لحل المشكلة من أجل تصميم برامج اوامر التفرع والتكرار بلغة البرمجة. ٣-ب-٥ يوظف استخدام أوامر البرمجة في حل مشكلة في موقف تعليمي في مجال تخصصه.
٣-جـ- المهارات المهنية الخاصة بالمقرر	٣-ج-١ يستخدم النوافذ داخل الشاشات الرئيسية لبرنامج الفيجوال بيسك. نت. ٣-ج-٢ يستخدم الشاشة Windows Application. ٣-ج-٣ يستخدم المربعات الحوارية والقوائم وشاشة البرنامج Form وخصائصها داخل لغة البرمجة فى تصميم برامج. ٣-ج-٤ يستخدم أوامر الرسم في برامج الرسوم مختلفة الأشكال بإستخدام لغة البرمجة فيجوال بيسك. نت. ٣-ج-٥ يصمم برامج متعددة الأهداف والأغراض باستخدام أوامر لغة البرمجة فيجوال بيسك. نت. ٣-ج-٦ يمارس تطبيقات الويب في برنامج الفيجوال بيزيك. نت. ٣-ج-٧ يختبر أدوات التحكم في الفيجوال بيزيك. نت . ٣-ج-٨ يستخدم لغة c++ وأهم أوامرها. ٣-ج-٩ يصمم برامج متعددة الأهداف والأغراض باستخدام أوامر لغة البرمجة c++.

٣-د - المهارات العامة	٣-د-٢ يقدر على التعلم الذاتي من خلال تقديم أوراق العمل الخاصة به. ٣-د-٣ يشارك أقرانه في تقديم الواجبات التعليمية باستخدم الفيجوال بيسك. نت في تخصصه. ٣-د-٤ يقدر على البحث والتطبيق من خلال تقديم أنشطة تطبيق مصاحبه لوحدات المقرر النظرية. ٣-د-٥ يعمل ضمن فريق لفهم نطاق. نت. ٣-د-٦ يستخدم شبكة المعلومات في ممارسة تطبيقات البرمجة بلغة c++ ٣-د-٧ يتواصل مع الآخرين لاستخدام أوامر التحكم في لغة c++ ٣-د-٨ يعمل ضمن فريق لنشر لغة c++ وأهم أوامرها. ٣-د-٩ يشارك أقرانه في تصميم البرمجيات التعليمية باستخدم لغة c++. ٣-د-١٠ يعمل ضمن فريق لربط برنامج بلغة c++ باوامر التفرع والتكرار.

٤- **محتوى المقرر :**

الأسبوع	عـــــدد الساعات	محتوى المقرر
١	٤	١/٤المفاهيم الأساسية فى لغة البرمجة فيجوال بيسك. نت، نطاق دوت نت.
٢	٤	٢/٤مكونات الشاشة الرئيسية لبرنامج الفيجوال بيسك. نت.
٣	٤	٣/٤أدوات التحكم في لغة البرمجة فيجوال بيسك. نت.
٤	٤	٤/٤تطبيقات الحاسب والويب في برنامج الفيجوال بيزيك. نت
٥	٤	٥/٤اوامر التفرع والتكرار في الفيجوال بيزيك. نت
٦	٤	٦/٤الخصائص الخاصة بالأدوات التحكم في لغة فيجوال بيزيك. نت.
٧	٤	٧/٤لغة C++ وأهم أوامرها.
٨	٤	٨/٤نظام الرسوم GDI+ وأوامرها، المربعات الحوارية والقوائم وشاشة البرنامج Form وخصائصها.
٩	٤	٩/٤تصميم البرمجيات التعليمية باستخدم الفيجوال بيزيك في المعمل
١٠	٤	١٠/٤أوامر الرسم في برامج الرسوم مختلفة الأشكال بإستخدام لغة البرمجة فيجوال بيسك. نت.
١١	٤	١١/٤لغة C++ ومكوناتها
١٢	٤	١٢/٤تصميم البرمجيات التعليمية باستخدام لغة الC++
١٣	٤	١٣/٤اوامر التفرع والتكرار في لغة C++
١٤	٤	١٤/٤امتحان شفهي.
	٥٦	اجمالي عدد الساعات

٥- **أساليب التعليم والتعلم**

١/٥ محاضرات √

٢/٥ جلسات مناقشة √

٣/٥ أنشطة فى الفصل (السكشن) √

٦- أساليب التعليم والتعلم للطلاب ذوى القدارت المحدودة :	لا يوجد طلاب ذوى قدرات محدودة

٧- تقويم الطلاب :

٧-أ- الأساليب المستخدمة:	٧-أ-١ الحضور والمشاركة ٧-أ-٢ الاختبارات الدورية ٧-أ-٣ امتحان منتصف الفصل الدراسى ٧-أ-٤ الإمتحان العملى ٧-أ-٥ الامتحان الشفوى ٧-أ-٦ الامتحان النظرى
٧-ب- التوقيت:	٧-ب-١ أعمال الفصل : الأسبوعين الخامس والعاشر ٧-ب-٢ الشفهي: الأسبوع الرابع عشر ٧-ب- ٣ التطبيقي : الأسبوع الثالث عشر ٧-ب- ٤ امتحان أخر الفصل : الأسبوع الخامس عشر
٧-جـ- توزيع الدرجات	**التقييم ١ أعمال الفصل : ٢٠ درجة ١٣%** **التقييم ٢ امتحان التطبيقي : ٣٠ درجة ٢٠%** **التقييم ٤ امتحان الشفهي: ١٠ درجة ٧%** **التقييم ٥ امتحان نهاية الفصل ٩٠ درجة %**

٨- قائمة الكتب الدراسية والمراجع :

٨-أ- مذكرات : مذكرة الحاسب الالي بيسيك متقدم المعدة بواسطة استاذ المادة وقسم تكنولوجيا التعليم والحاسب الالي

٨-ب- كتب ملزمة : كتب أساتذة القسم والمجال في فنيات الحاسب الالي بيسيك متقدم

٨-جـ- كتب مقترحة

كتاب أ.د / محمد عطية خميس ثقافة الكمبيوتر
Maakter, A. (2010). *C++*. Retrieved May, 26, 2014 from http://www.mkasoft.com
Davis, H. (2002). *Visual basic.net programming*. In Marilyn Smith and Scott Swigrat (Eds.). USA: Richard Mills.
Grundgeiger, D. (2002). *Programming c.c++* USA: O'reilly.

٨–د – دوريات علمية أو نشرات ... إلخ	٨–د–١ مجلة المكتبات والمعلومات العربية. ٨–د–٢ مجلة تكنولوجيا الاتصالات والمعلومات. ٨–د–٣ مجلة جامعة الملك سعود. ٨–د–٤ مجلة عالم الكتب.

منسق المادة: د. مينا وديع جرجس

رئيس مجلس القسم العلمي أ.د / أحمد مصطفى عصر

الفصل الاول

الدوال في لغة VISUAL BASIC .NET

البنية التكرارية Loops

أحيانا نحتاج في تطبيقاتنا إلى تكرار أمر برمجي معين أكثر من مرة، وهذا لا يعني أن نعيد تنفيذ نفس الأمر فقط، بل يمكننا كذلك إجراء بعض العمليات على كل عنصر يصله التكرار، فلو افترضنا أنه لدينا قاعدة بيانات تحتوي على بيانات العملاء، وأردنا إضافة أرقام الهواتف لهم جميعا، ستلاحظ أن عملية إسناد هذا الحقل لكل عميل ستستنزف منك وقتا طويلا لا سيما وإن كان عدد العملاء كبيرا جدا،

لذلك تعتبر الآليات التكرارية أو الحلقات هي أفضل خيار لتكرار تعليمات معينة بقدر محدد مع إمكانية إضفاء لمسة خاصة على كل عنصر تبلغه البنية التكرارية.

أو لنفترض مثلا أننا بصدد برمجة تطبيق صغير يخزن قيمة رقمية في متغير معين ويطلب من المستخدم تخمين هذه القيمة، حتما نحن لا نعرف عدد الاحتمالات التي سيقوم بها المستخدم لكي يصل إلى نفس القيمة المخزنة عندنا، لذلك يتوجب طرح نفس السؤال في كل مرة يدخل المستخدم فيها قيمة مخالفة للقيمة المخزنة.

أو لنفترض أننا بصدد إنتاج برنامج بسيط يسأل المستخدم عن اسم أول خلفاء الدولة العباسية، فإن كان الجواب صائبا يتوقف البرنامج وإن كان الجواب خاطئا يعيد طرح السؤال على المستخدم إلى حين يحصل على الجواب الصحيح، في حالتنا هذه نحن لا نعلم عدد المرات اللازمة لكي يجيب المستخدم بالجواب الصائب، لذلك فنحن بحاجة إلى إجراء تكرار للأمر الذي يطبع السؤال في كل مرة يخطىء المستخدم في الإجابة.

توفر لنا لغة الفيجوال بيسك عدة أنواع من البنى التكرارية سنستعرضها بالترتيب في الفقرات القادمة.

الصيغة التكرارية الشرطية For Next:

تمكننا البنية التكرارية For..Next من تكرار أمر برمجي معين لعدد مرات محدد، حيث تسمح لنا بتحديد نقطتي بداية ونهاية التكرار مع إمكانية تحديد معامل التدرج، وصيغتها كما يلي:

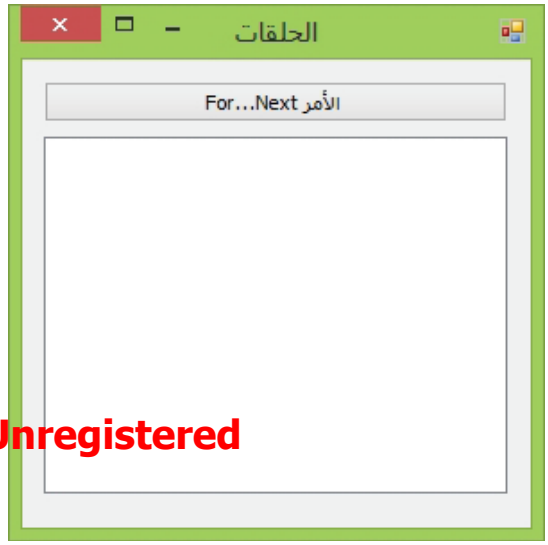
```
For Count = Start To End [Step Step]
        '[statements]
         [Exit For]
Next Count
```


حيث المتغير Count هو العداد الذي سيحدد عدد مرات تكرار الأمر برمجي، ويبدأ من القيمة الرقمية Start وينتهي بالقيمة الرقمية End مع إمكانية تحديد معامل التدرج (في كل تكرار بكم تزداد أو تنقص نسبة العداد افتراضيا تكون الزيادة بواحد) في الأمر Step، بعد ذلك نقوم بكتابة الأمر البرمجي الذي نود تكراره، ويمكننا الخروج من التكرار من خلال الأمر Exit For، في ختام البنية التكرارية نضع الكلمة Next متبوعة باسم العداد (اختيارية ويمكننا عدم كتابة اسم العداد)، وفيما يلي مثال بسيط يوضح كيفية استخدام هذا النوع من الحلقات:

قم بإنشاء مشروع من نوع Windows Forms Application، ثم اسحب على الفورم الأدوات التالية وغير خصائصها كما يلي:

الأداة	اسمها	دورها
ListBox	lbNumbers	من أجل عرض القيم الناتجة عن التكرار
Button	btnRepeat	من أجل تنفيذ أمر التكرار باستخدام الحلقات For..Next

قم بتصميم الفورم كما تعرض الصورة الآتية:



انقر على الزر btnRepeat مرتين من أجل الانتقال إلى الحدث Click الخاص به، واكتب الأمر التالي:

```
    Private Sub btnRepeat_Click(sender As Object, e As
EventArgs) Handles btnRepeat.Click
        'متغير رقمي بمثابة عداد للحلقات
        Dim nCount As Integer

        For nCount = 0 To 10 Step 1
            Me.lbNumbers.Items.Add("العنصر: " & nCount)
        Next

    End Sub
```

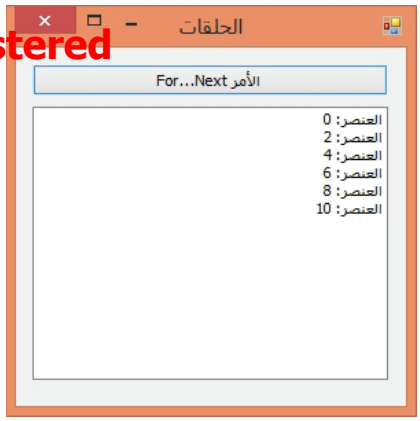

قمنا بالإعلان عن متغير رقمي اسمه Count ثم استعملناه في عملية التكرار ليبدأ من العدد 0 وينتهي بالعدد 10 مع الزيادة بواحد عند كل تكرار، وفي كل مرة يقوم التكرار بإضافة قيمة العداد إلى أداة lbNumbers مدموجة مع النص "العنصر:"، عند تنفيذ هذا الكود ينبغي أن نحصل على نتيجة مماثلة للصورة الآتية:

ملحوظة:

الأمر Step في المثال أعلاه لا دور له لأن نسبة الزيادة الافتراضية هي 1، ويمكننا استخدامه في حال أردنا تغيير قيمة التدرج، كما يوضح الكود الآتي الذي غيرنا فيه قيمة Step:


```
Dim nCount As Integer

For nCount = 0 To 10 Step 2
    Me.lbNumbers.Items.Add("العنصر: " & nCount)
Next
```

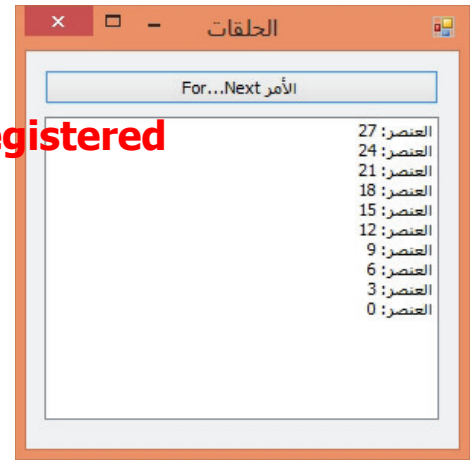
هذه المرة جعلنا نسبة التدرج هي 2 وبالتالي سنحصل على النتيجة التالية عند تنفيذ الكود أعلاه:



يمكننا كذلك جعل قيمة التدرج سلبية / تراجعية كما يلي:


```
Dim nCount As Integer

For nCount = 27 To 0 Step -3
    Me.lbNumbers.Items.Add("العنصر: " & nCount)
Next
```

بعد تنفيذ الكود أعلاه سنحصل على النتيجة الآتية:



الصيغة التكرارية الشرطية Do..Loop :

ترتكز البنية التكرارية من نوع Do..Loop على شرط التكرار وحسب نتيجته يتم إعادة تنفيذ الأوامر، ويوجد بها عدة أنواع سوف نتعرف عليها بالترتيب:

الصيغة التكرارية الشرطية Do..While :

هذه البنية التكرارية تعتمد على نتيجة الشرط الذي يأتي بعد الكلمة While، ويتم تكرار تنفيذ الأوامر مادامت نتيجة الشرط متحققة True، وهذه صيغتها:


```
Do
    'الأوامر
Loop While 'الشرط
```

سنتعرف الآن بحول الله على مثال لاستخدام هذه البنية التكرارية، قم بإنشاء مشروع جديد من نوع Windows Forms Application واسحب على الفورم الأدوات التالية:

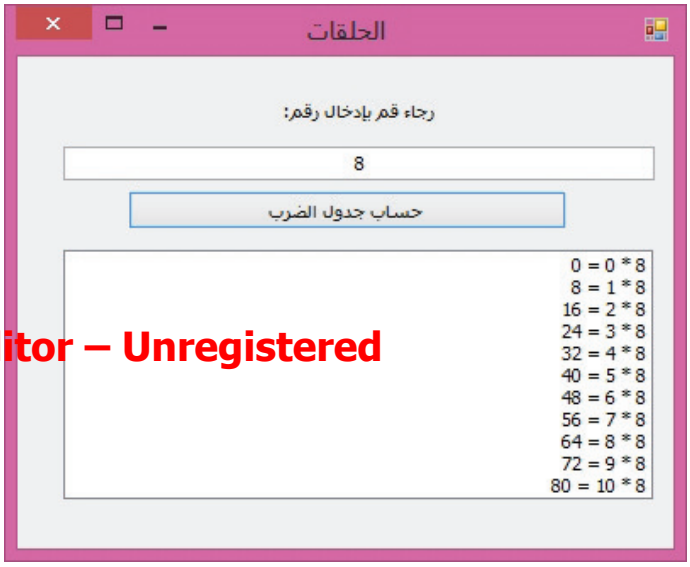
الأداة	اسمها	دورها
Label	lblNumber	من أجل عرض السؤال
TextBox	txtNumber	من أجل إدخال الرقم
Button	btnCalculate	من أجل حساب جدول ضرب الرقم
ListBox	lbDetails	من أجل عرض نتائج جدول الضرب

الآن أدخل إلى الحدث Click الخاص بالزر btnCalculate وقم بكتابة الكود الآتي:

```
    Private Sub btnCheck_Click(sender As Object, e As
EventArgs) Handles btnCalculate.Click

        Dim Number As Integer = Val(txtNumber.Text)
        Dim Counter As Integer = 0
        Do
            lbDetails.Items.Add(Number & " * " & Counter & "
= " & Number * Counter)
            Counter += 1
        Loop While Counter <= 10

    End Sub
```


أعلنا في الأول عن متغير اسمه Number يستقبل القيمة المدخلة في مربع النص txtNumber بعد أن يقوم بتحويلها إلى قيمة رقمية من خلال الدالة Val ثم بعد ذلك أهلنا عن متغير عداد اسمه Counter أعطيناه القيمة 0 كقيمة بدئية، ثم بدأنا عملية التكرر التي تقوم بعرض نتيجة ضرب قيمة العدد المدخل Number مع قيمة العداد Counter التي تتغير تزايديا عند كل تكرار، ويتم تكرار هذه العملية التكرارية إلى أن تصبح قيمة العداد 10 أي أن جدول الضرب الذي سيعرض سيظهر لنا

نتيجة جداء العدد المدخل مع الأعداد الابتدائية بدءا من 0 وانتهاء ب 10، أي بعد تنفيذ البرنامج لو أدخلنا القيمة 8 سنحصل على النتيجة الآتية:



يمكننا تقديم الكلمة While بعد الكلمة Do وسنحصل على نفس النتيجة كما يلي:

```
        Dim Number As Integer = Val(txtNumber.Text)
        Dim Counter As Integer = 0
        Do While Counter <= 10
            lbDetails.Items.Add(Number & " * " & Counter & "
= " & Number * Counter)
            Counter += 1
        Loop
```


الفرق بين تقديم While وبين تأخيرها يتمثل في أن الأولى تمنع تنفيذ الأمر إلا بعد أن يتحقق الشرط، بينما While التي تأتي في الأخير بعد Loop تسمح بتنفيذ الأمر للمرة الأولى حتى مع عدم تحقق الشرط، والكود التالي يوضح الفرق بين الأمرين.

تقديم While:


```
Do While 1 = 3
    MsgBox("هذا الكود لن ينفذ أبدا")
Loop
```

في الكود أعلاه لن يتم عرض الرسالة لأن شرط While غير متحقق لأن 1 لا يساوي ثلاثة ولأن While جاءت أولا في بداية التكرار.

تأخير While:

```
Do
    MsgBox("هذا الكود سينفذ مرة واحدة")
Loop While 1 = 3
```


في الكود أعلاه سيتم عرض الرسالة في كل مرة يتم تنفيذه لأن While أتت في نهاية التكرار على خلاف الكود الأول.

الصيغة التكرارية الشرطية Do..Until :

مثلها مثل البنية السابقة Do..While إلا أن هذه البنية تسمح بتكرار الأوامر إلى أن يتحقق الشرط وليس مادام الشرط متحققا، أي أن التعليمات التي ستأتي بعد الكلمة Do سيتم تكرارها إلى غاية تحقق الشرط الذي يأتي بعد الكلمة Until.

الفرق بين Until و While:

Until	While
تسمح بتكرار الأوامر إلى أن يتحقق الشرط أي تصبح نتيجته True	تسمح بالتكرار مادام الشرط متحققا ويخرج من التكرار حينما لا يتحقق الشرط أي تصبح نتيجته False

سنشتغل على نفس المثال السابق الذي يقوم بحساب جدول الضرب، لكن هذه المرة مع استخدام الأمر Until بدل While، أي أن الشفرة ستكون كما يلي:

```
    Private Sub btnCheck_Click(sender As Object, e As
EventArgs) Handles btnCalculate.Click

        Dim Number As Integer = Val(txtNumber.Text)
        Dim Counter As Integer = 0
        Do Until Counter = 10
            lbDetails.Items.Add(Number & " * " & Counter & "
= " & Number * Counter)
            Counter += 1
        Loop

    End Sub
```


لاحظ أننا غيرنا الشرط، بدل أن يكون العداد أصغر من أو يساوي 10 أصبح معنى الشرط، قم بتكرار الأوامر إلى أن يصبح العداد Counter مساويا ل 10 وفي كل مرة تزداد قيمته بواحد إلى أن يتحقق الشرط فينتهي التكرار، لو نفذنا الكود أعلاه وأدخلنا العدد 70 مثلا سوف نحصل على النتيجة التالية:

رأينا فيما تقدم أنه يمكننا تسبق الكلمة While وتأخيرها نفس الكلام ينطبق على الكلمة Until يمكننا وضعها في البداية بعد الكلمة Do ويمكننا كذلك وضعها في نهاية التكرار بعد الكلمة Loop، في الحالتين معا سيتم تنفيذ الأوامر مادام الشرط غير متحققا، بمعنى أن الكود التالي:


```
Do Until 1 = 3
    MsgBox("هذا الكود سينفذ من دون توقف لأن الشرط غير متحقق ولن يتحقق")
Loop
```


هو نفسه الكود التالي:

```
Do
    MsgBox("هذا الكود سينفذ من دون توقف لأن الشرط غير متحقق ولن يتحقق")
Loop Until 1 = 3
```

ففي الحالتين معا سيتم عرض الرسالة من دون توقف لأن الشرط غير متحقق دائما، ومادام الأمر كذلك فستستمر عملية تكرار الأوامر.

الصيغة التكرارية For Each...Next :

تسمح لنا هذه البنية التكرارية بتكرار أمر معين على جميع عناصر مجموعة معينة وصيغتها كما يلي:

```
For Each Item As DATA_TYPE In GROUP
    'الأوامر المراد تكرارها على العناصر
Next Item
```


سنقوم بإنشاء برنامج بسيط يستقبل اسم المستخدم ويقوم بتقسيمه وتوزيعه إلى حروف متفرقة وعرض كل حرف في سطر.

قم بإنشاء مشروع جديد من نوع Windows Forms Application وضع عليه الأدوات التالية:

الأداة	اسمها	دورها
Label	lblName	من أجل عرض السؤال
TextBox	txtName	من أجل إدخال الاسم
Button	btnSplit	من أجل تقسيم الاسم وتوزيعه
ListBox	lbDetails	من أجل عرض الحروف المكونة للاسم

انقر على الزر btnSplit مرتين من أجل الولوج إلى الحدث Click وقم بكتابة الكود التالي:

```
    Private Sub btnSplit_Click(sender As Object, e As
EventArgs) Handles btnSplit.Click

        Dim FullName As String = txtFullName.Text
```

```
        For Each chr As Char In FullName
            Me.lbDetails.Items.Add(chr)
        Next

    End Sub
```


في الشفرة أعلاه قمنا بالإعلان عن متغير اسمه FullName يستقبل القيمة النصية المدخلى في مربع النص txtFullName، ثم بعد ذلك بدأنا البنية التكرارية For Each التي تأخذ كل عنصر من عناصر المجموعة (في حالتنا هذه كل حرف من السلسلة النصية) وتقوم بعرضه في أداة ListBox.

الأمر With:

يسمح لنا الأمر With بتكرار مجموعة من الأوامر التي تشير إلى نفس الكائن Object على سبيل المثال، نريد تغيير خصائص زر معين بواسطة الكود، بدل أن نكتب الأوامر التالية:


```
Button1.Text = "النص الظاهر"
Button1.BackColor = Color.Blue
Button1.Size = New Point(100, 40)
Button1.ForeColor = Color. Yellow
Button1.TextAlign = ContentAlignment.MiddleCenter
Button1.Focus()
```

نقوم بتعيين اسم الكائن وفي حالتنا هذه هو الزر المسمى Button1 في الجزء With ثم نقوم باستدعاء الوظائف والخصائص مباشرة بعد كتابة رمز النقطة كما يلي:


```
With Button1
    .Text = "الظاهر النص"
    .BackColor = Color.Blue
    .Size = New Point(100, 40)
    .ForeColor = Color.Yellow
    .Focus()
    .TextAlign = ContentAlignment.MiddleCenter
End With
```

معالجة النصوص Handling Strings

توفر لنا لغة الفيجوال بيسك مجموعة من الدوال التي تمكننا من التعامل مع النصوص وإجراء مختلف العمليات التي ترغب فيها من تقطيع ودمج وبحث وتصغير وتكبير حالات الأحرف وغير ذلك من الدوال المهمة التي سنستعرضها بالترتيب مستعرضين لكل دالة مثالا تطبيقيا.

الحصول على طول النص:

إذا أردنا أن نعرف عدد الحروف المكونة لنص معين، يمكننا استخدام الخاصية Length وكذلك الدالة Len وفيما يلي كيفية استخدامهما:

```
Dim Name As String = "Khalid"
MsgBox(Name.Length)
'أو
MsgBox(Len(Name))
```

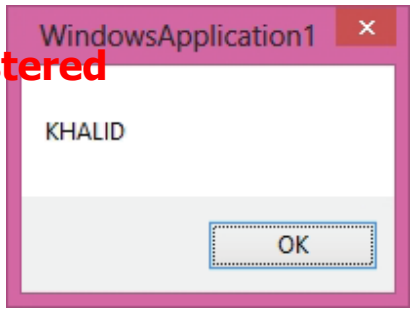

في الحالتين معا، سيتم عرض رسالة تحتوي على طول السلسلة النصية وهو 6 أحرف.

تكبير حالة الأحرف:

إذا أردنا تكبير حالة الأحرف، فيمكننا عمل ذلك بواسطة الدالة UCase، وكذلك الخاصية ToUpper كما يوضح المثال التالي:

```
Dim Name As String = "khalid"
MsgBox(Name.ToUpper())
'أو
MsgBox(UCase(Name))
```


النتيجة الحاصلة من الدالة UCase أو الخاصية ToUpper هي نفس القيمة النصية khalid لكن بحالة أحرف كبيرة هكذا KHALID كما توضح هذه الرسالة:

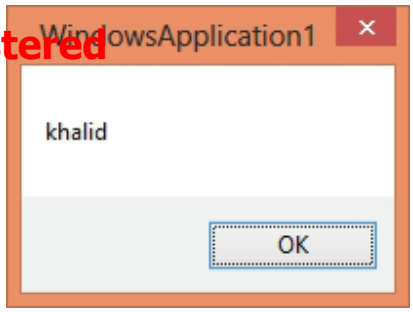


تصغير حالة الأحرف

بمقابل الدالة UCase التي تقوم بتكبير حالة الأحرف، يوجد لدينا الدالة LCase التي تقوم بتصغير حالة الأحرف، وكذلك بمقابل الخاصية ToUpper تتيح لنا لغة الفيجوال بيسك الخاصية ToLower التي تسمح بتصغير حالة الحروف، كما يوضح لنا المثال التالي:


```
Dim Name As String = "KHALID"
MsgBox(Name.ToLower())
'أو
MsgBox(LCase(Name))
```


سنحصل على النتيجة التالية في الحالتين معا:



اجتزاء النصوص SubString

توفر لنا لغة الفيجوال بيسك طريقتين للحصول على جزء معين من سلسلة نصية، الدالة الأولى هي Mid والدالة Substring التابعة للفئة String، الدالة الأولى Mid تنتظر ثلاثة برامترات كما يلي:

```
Mid(string_Name, start_Index, Length)
```

البرامتر الأول string_Name نضع فيه النص الأصلي الذي نريد اجتزاءه.

البرامتر الثاني start_Index نحدد فيه رتبة أول حرف نريد أن يبدأ منه النص الجديد الناتج عن الاجتزاء علما أن رتبة أول حرف في النص الأصلي هي 1 وليست 0.

البرامتر الثالث Length نضع فيه عدد النص الذي نريد الحصول عليه.

وهذه أمثلة على كيفية استخدام الدالة Mid من أجل اجتزاء النصوص:

```
Dim Name As String = "KHALID"

Dim strResult1 As String = Mid(Name, 1, 3)

Dim strResult2 As String = Mid(Name, 2, 4)
```

في المتغير الأول strResult1 قمنا بتخزين القيمة المجتزأة من النص الأصلي KHALID بدء من أول حرف وطول نتيجة النص الجديد هي 3، أي أن النتيجة هي KHA.

بينما في المتغير الثاني strResult2 قمنا بتخزين القيمة المجتزأة من النص الأصلي KHALID بدء من ثاني حرف وطول نتيجة النص الجديد هو 4، أي أن النتيجة هي HALI.

الدالة الثانية التي تمكننا من تجزئة النص هي SubString التابعة للفئة String، وصيغتها كما يلي:


```
String_Name.Substring(start_Index, Length)
```

نستبدل الكلمة String_Name باسم النص الذي نريد تقطيعه، ثم نحدد في البرامتر Start_Index بداية التقطيع التي تبدأ من 0 على خلاف الدالة MID التي تعتبر أن أول حرف في السلسلة النصية رتبته 1. بعد ذلك نحدد طول النص المراد الحصول عليه بعد عملية الاجتزاء، كما في الأمثلة التالية:


```
Dim Name As String = "KHALID"

Dim strResult1 As String = Name.Substring(0, 3)

Dim strResult2 As String = Name.Substring(2, 4)
```

المتغير الأول سيحتوي على نتيجة النص بعد التقطيع الذي يبدأ من أول حرف و يحسب ثلاثة حروف معه أي أن strResult1 سيحتوي على KHA.

أما المتغير الثاني فإن قيمته ستكون هي نتيجة التقطيع بدءا من الحرف الثالث (لأن العد في الدالة SubString يبدأ من 0) وطول النص لدينا هو 4 وبالتالي سيحتوي المتغير strResult2 على ALID.

تقسيم النصوص Splitting

تمكننا الدالة Split التابعة للفئة String من تقطيع النص إلى مجموعة من العناصر وتخزينها في مصفوفة إذ سيتم اعتبار كل جزء من أجزاء النص عنصرا من عناصر المصفوفة الناتجة.

في المثال التالي، سنقوم بتخزين مجموعة من الأسماء المفصولة بعلامة شرطة في متغير نصي، ثم نقوم بتقطيع هذا النص حسب هذه الشرطة بواسطة الدالة Split ثم نخزن القيم الناتجة في مصفوفة نصية:


```
Dim Names As String = "Khalid-Hamid-Nihad-Hussain"

Dim ArrayNames() As String = Names.Split("-")

MsgBox(ArrayNames(0)) 'display Khalid
MsgBox(ArrayNames(1)) 'display Hamid
MsgBox(ArrayNames(2)) 'display Nihad
MsgBox(ArrayNames(3)) 'display Hussain
```

كما تلاحظ معي فالدالة Split تستلزم منا تحديد الرمز المراد اعتماده في التقطيع، وفي حالتنا هذه هو الشرطة-.

دمج النصوص Concatening

تسمح لنا لغة الفيجوال بيسك باستخدام مجموعة من الطرق لدمج النصوص، رأينا فيما تقدم معنا من فصول أولى أن الرمز & والرمز + يستخدمان لدمج القيم النصية، كما يبين لنا المثال التالي:

```
Dim First_Name As String = "Khalid"
Dim Last_Name As String = " ESSAADANI"

Dim FullName As String

FullName = First_Name + Last_Name
'Or
FullName = First_Name & Last_Name
```


المتغير FullName في الحالتين معا سيحتوي على نتيجة دمج قيمتي المتغيرين First_Name و Last_Name أي أن النتيجة ستكون: Khalid ESSAADANI.

توجد لدينا كذلك الدالة Concat التابعة للفئة String التي تقوم بدمج النصين الممررين لها كما يوضح المثال التالي:

```
Dim First_Name As String = "Khalid"
Dim Last_Name As String = " ESSAADANI"

Dim FullName As String

FullName = String.Concat(First_Name, Last_Name)
```

المتغير FullName سيضم قيمتي المتغيرين First_Name و Last_Name بعد أن تقوم الدالة Concat بدمجهما.

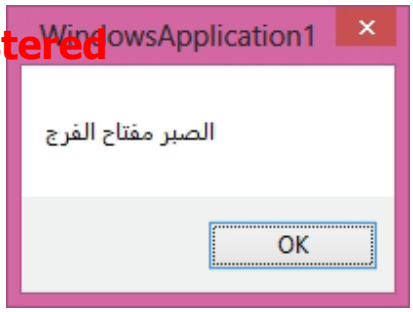
يوجد لدينا في لغة الفيجوال بيسك، الدالة Join التي تقوم بنفس الدور، لكن تتطلب منا تحديد رمز الفصل بين النصوص المراد دمجها، وهذا مثال على كيفية استخدام هذه الدالة:

```
Dim First_Name As String = "Khalid"
Dim Last_Name As String = "Mansour"

Dim FullName As String

FullName = String.Join(" ", First_Name, Last_Name)
```


لاحظ أننا وضعنا في رمز الربط بين القيمتين رمز المساحة الفارغة Space لكي يتم ترك فراغ بين الاسم والنسب، ويمكننا الاستغناء عن تحديد رمز الربط وترك قيمته فارغة كما يمكننا وضع أي نص نريد وضعه كرابط للقيم المراد دمجها وهذا مثال أوضح على استخدام الدالة Join التابعة للفئة String:


```
Dim Expression1 As String = "الصبر"
Dim Expression2 As String = "الفرج"

Dim Expression3 As String = String.Join(" مفتاح ",
Expression1, Expression2)

MsgBox(Expression3)
```


بعد تنفيذ الكود أعلاه، سوف تحصل على النتيجة التالية:



البحث داخل النصوص Contains

نستطيع البحث عن قيمة معينة داخل قيمة نصية من خلال الدالة Contains التي تعيد لنا القيمة True في حال العثور على القيمة المبحوث عنها، وتعيد لنا القيمة False في حال عدم العثور عليها، وهذا مثال على كيفية استخدام الدالة Contains التابعة للفئة String:

```
Dim strText As String = "إذا فشلت في التخطيط فقد خططت للفشل"

If strText.Contains("التخطيط") = True Then
    MsgBox("القيمة المبحوث عنها موجودة")
Else
    MsgBox("آسف، القيمة المبحوث عنها غير موجودة")
End If
```


ستبحث الدالة Contains عن الكلمة "التخطيط" داخل المتغير النصي strText وتظهر لنا الرسالة حسب نتيجة البحث، في حالتنا هذه القيمة المعادة للدالة Contains هي True إذن ستظهر الرسالة الأولى: القيمة المبحوث عنها موجودة.

يمكننا القيام بنفس عملية البحث من نصية من داخل سلسلة نصية، من خلال الدالة IndexOf التي تعيد لنا رتبة أول حرف من القيمة المبحوث عنها في حال العثور عليها، أما في حال عدم العثور عليها سوف تعيد لنا الدالة IndexOf القيمة السلبية -1 .

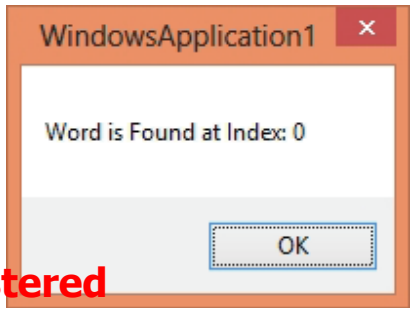
فيما يلي مثال يعرض للحالتين معا، الحالة الأولى تم العثور على النص المبحوث عنه، والحالة الثانية لم يتم العثور عليه.

الحالة الأولى:


```
Dim strText As String = "Think First, Code Later"

If strText.IndexOf("Think") > -1 Then
    MsgBox("Word is Found at Index: " &
           strText.IndexOf("Think"))
Else
    MsgBox("Not Found :(")
End If
```

الكلمة Think موجودة داخل قيمة المتغير strText إذن الدالة IndexOf سوف تعيد

قيمة مخالفة ل -1، وبالتالي سيتم عرض الرسالة الأولى التي تفيد أن الكلمة المبحوث عنها تم العثور عليها مع تحديد رتبة أول حرف منها، كما تعرض الرسالة التالية:

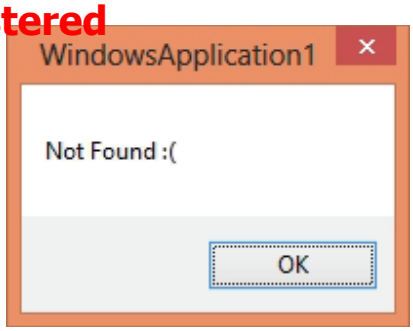


لأن الكلمة Think هي أول كلمة في السلسلة النصية التي ستشملها عملية البحث فتم إرجاع الرتبة صفر لأن أول حرف من الكلمة يوجد في هذه الرتبة.

بينما في الحالة الثانية التي يعرضها الكود التالي فإن الدالة IndexOf لن تجد الكلمة المبحوث عنها وسوف تعيد لنا القيمة -1 وبالتالي ستظهر الرسالة التي تفيد أن الكلمة المبحوث عنها غير موجودة:

```
        Dim strText As String = "Think First, Code Later"
        If strText.IndexOf("Manage") = -1 Then
            MsgBox("Not Found :(")
        Else
            MsgBox("Word is Found at Index: " &
strText.IndexOf("Think"))
        End If
```


طبعا لأن الكلمة Manage غير موجودة فإن الشرط متحقق لأن الدالة IndexOf سوف تعيد بالفعل القيمة -1 وبالتالي ظهور الرسالة الآتية:



استبدال النصوص Replace

أحيانا قد نحتاج إلى استبدال نص معين بنص آخر، إذا أردنا عمل ذلك فإن لغة الفيجوال بيسك توفر لنا الدالة Replace التي تقوم بهذه العملية كما يبين المثال الآتي:

```
Dim FalseText As String = "الرياض عاصمة العراق"

Dim TrueText As String = FalseText.Replace("الرياض",
"بغداد")

MsgBox(TrueText)
```


المتغير الأول يحتوي على عبارة خاطئة مفادها أن الرياض عاصمة بغداد، وقمنا بتصحيح العبارة من خلال الدالة Replace التي مكنتنا من استبدال مدينة الرياض بالمدينة الصحيحة التي تعد عاصمة للعراق.

المتغير TrueText سيحتوي على العبارة التالية: بغداد عاصمة العراق.

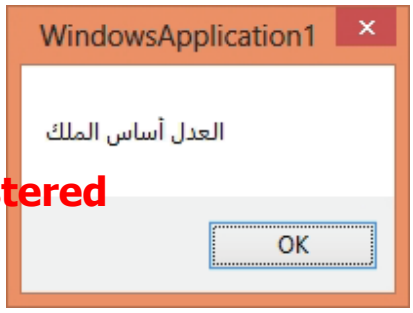
إدراج النصوص Insert

إذا أردنا أن نضيف نصا ما إلى نص آخر مع إمكانية تحديد الرتبة التي نريد إدخال النص الجديد فيها، فإن الدالة Insert تفي بالغرض وتسمح لنا بهذه العملية كما يوضح المثال الآتي:

```
Dim Expression As String = "أساس الملك"

Dim FinalExpression As String = Expression.Insert(0,
"العدل ")
```

```
MsgBox(FinalExpression)
```


المتغير الأول يحتوي على عبارة ناقصة هي: أساس الملك، فقمنا بإضافة الكلمة الناقصة في الرتبة 0 وهي الكلمة: العدل، وبالتالي ستصبح قيمة المتغير الثاني FinalExpression هي:



مقارنة النصوص Compare

قد نحتاج في بعض الأحيان إلى إجراء مقارنات على النصوص من أجا التحقق من أن نصين متوافقان أو غير متوافقان، لعمل ذلك توجد عندنا الدالة Compare والدالة CompareTo اللتان تقومان بمقارنة نصين فإن تم إرجاع القيمة 0 فهذا يعني أن النصين متماثلان، وإن تم إرجاع القيمة 1 فهذا يعني أن القيمتين غير متوافقتين، كما يعرض لنا المثال التالي الذي استخدمنا فيه الدالتين معا:


```
Dim First_String As String = "Khalid"

Dim Second_String As String = "Hussain"

Dim Result As Integer

Result = String.Compare(First_String, Second_String)

If Result = 0 Then
    MsgBox("النص الأول مماثل للنص الثاني")
```

```
Else
    MsgBox("النص الأول غير مماثل للنص الثاني ")
End If
'أو
Result = First_String.CompareTo(Second_String)

If Result = 0 Then
    MsgBox("النص الأول مماثل للنص الثاني ")
Else
    MsgBox("النص الأول غير مماثل للنص الثاني ")
End If
```


الدالة Format

قبل أن نتطرق لشرح كيفية التحكم في عرض العملات المالية والتواريخ وغيرهم، ينبغي أن نستعرض أولا الدالة Format التي تنتمي إلى الفئة String والتي تسمح لنا بالتعامل مع مختلف القيم، بحيث نحدد رتبة كل قيمة وشكل العرض بين مزدوجتين ثم بعد ذلك نحدد القيمة على شكل بارامتر للدالة Format كما يبين المثال التالي:

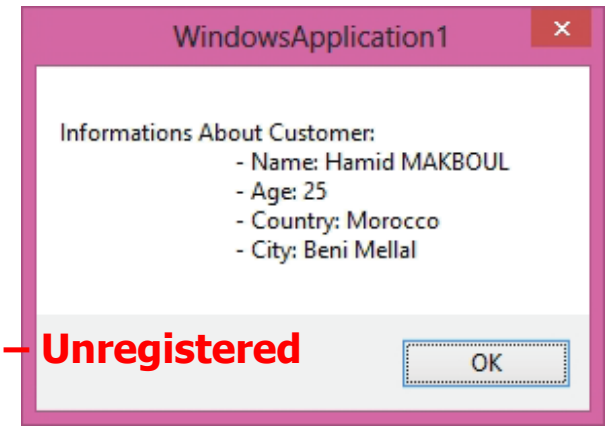

```
        Dim First_Name As String = "Khalid"
        Dim Last_Name As String = "ESSAADANI"
        Dim FullName As String
        FullName = String.Format("My name is: {0} {1}",
First_Name, Last_Name)
        MsgBox(FullName)
```

لاحظ أننا حددنا داخل الدالة Format ترتيب المتغيرات وشكل الظهور، حيث تركنا فراغا بين الاسم وبين النسب، عند تنفيذ الكود أعلاه سوف نحصل على هذه النتيجة:

وهذا مثال آخر للاستئناس بالدالة Format:

```
Dim Name As String = "Hamid MAKBOUL"
Dim Age As Byte = 25
Dim Country As String = "Morocco"
Dim City As String = "Beni Mellal"

MsgBox(String.Format("Informations About Customer:
                     - Name: {0}
                     - Age: {1}
                     - Country: {2}
                     - City: {3}", Name, Age,
Country, City))
```


سيتم عرض معلومات الشخص بشكل منسق حسب الترتيب الذي حددناه في الدالة Format، إذا قمنا بتنفيذ الكود أعلاه سوف نحصل على الرسالة التالية:

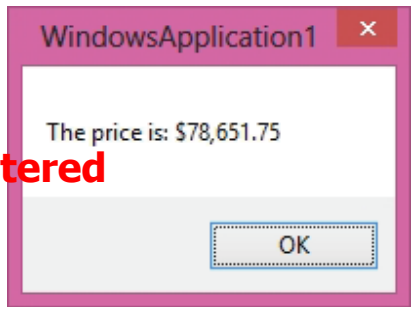


تنسيق العملات

لتنسيق العملات وعرضها بالشكل الصحيح، نستخدم الدالة Format ونضع في تنسيق القيمة المالية الرمز c دلالة على أن التنسيق سيكون على شكل عملة Currency كما يوضح الكود التالي:

```
Dim Price As Double = 78651.75
MsgBox(String.Format("The price is: {0:c}", Price))
```


سيتم تنسيق القيمة العشرية بالشكل المناسب لعرض المبالغ المالية في نظام التشغيل، عند تنفيذ الكود أعلاه سوف تحصل على نتيجة مماثلة لما يلي مع اختلاف في التنسيق والعملة حسب إعدادات نظام التشغيل عند:

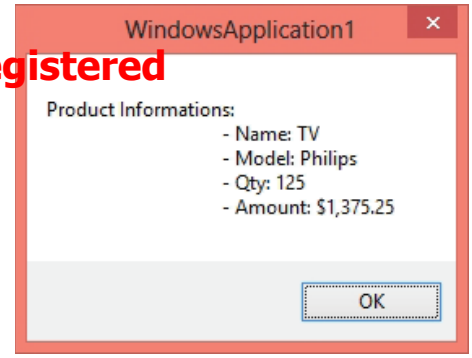


في المثال التالي سنقوم بنفس العمل، لكن مع استخدام متغيرات أخرى داخل الدالة Format:

```
Dim Name As String = "TV"
Dim Model As String = "Philips"
Dim Quantity As Short = 125
Dim Amount As Double = 1375.25
MsgBox(String.Format("Product Informations:
                                - Name: {0}
```

```
- Model: {1}
- Qty: {2}
- Amount: {3:c}", Name,
Model, Quantity, Amount))
```

كل ما قمنا به في المثال أعلاه هو أننا أعلنا عن مجموعة من المتغيرات بما فيها المتغير العشري Amount الذي قمنا بتنسيق طريقة عرضه ليظهر بشكل مالي، عند تنفيذ هذه الشفرة سنحصل على نتيجة شبيهة بالصورة الآتية:



ويمكننا أيضا التحكم في شكل التنسيق، بحيث يمكننا عرض المبالغ بالشكل الذي نريد، كأن نعرض الوحدات والمئات والألوف وغيرهم مفصولين بفاصل آخر بدل الفاصل أو أن نقوم بوضع الأصفار في الأرقام الناقصة في كل جزء لتسهل قراءة المبالغ أو أن نحدد عدد الأرقام بعد الفاصلة العشرية، إذا أردنا عمل ذلك فما علينا سوى وضع التنسيق الذي نريد على شكل رموز # في الدالة Format كما يوضح المثال التالي:


```
Dim Amount As Double = 45612.658
MsgBox(String.Format("{0:### ###.##}", Amount))
```

في هذا المثال سيتم عرض الألوف مفصولة عن المئات بفراغ وسيتم عرض رقمين فقط وراء الفاصلة العشرية كما يلي:

يمكننا تغيير شكل التنسيق كنا نحب، مثلا نعرض بدل الفراغ فاصلة، ونظهر بعد الفاصلة العشرية رقما واحدا فقط:

```
Dim Amount As Double = 45612.658
MsgBox(String.Format("{0:###,###.#}", Amount))
```

هذه المرة ستكون النتيجة هكذا:

إذا أحببنا أن نستعيض بأصفار عن الأماكن الناقصة في الأعداد والتي ليس لها تأثير حسابي لكنها تسهل عملية قراءة الأعداد يجب أن نضع بدل الرمز # القيمة 0 كما يوضح المثال التالي:

```
Dim Amount As Double = 45612.5
MsgBox(String.Format("{0:000 000.00}", Amount))
```


ستكون النتيجة كما يلي:

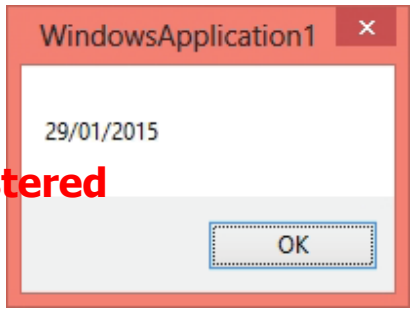
تنسيق التاريخ والوقت

بنفس الصيغة التي رأيناها في تنسيق العملات بواسطة الدالة Format يمكننا تنسيق القيم التاريخية لكن عبر تحديد شكل التنسيق بما يتلاءم مع طبيعة التاريخ والوقت الذي نود عرضه، افتراضيا نقوم بوضع الحرف d دلالة على Date من أجب تنسيق القيمة على شكل تاريخ.

في الفيجوال بيسك يوجد لدينا الخاصية Now() التي تقوم بإرجاع التاريخ والوقت الحاليين في نظام التشغيل، وهي القيمة التي سنقوم بالعمل على تنسيقها في المثال التالي:


```
Dim CurrentDate As Date = Now
MsgBox(String.Format("{0:d}", CurrentDate))
```

في المثال فوقه، قمنا بالإعلان عن متغير اسمه CurrentDate وبستقبل قيمة الخاصية Date، وبعد ذلك قمنا بعرض قيمته في رسالة بواسطة الدالة Format وحددنا في شكل التنسيق الرمز d للإشارة أن القيمة تاريخية، عند تنفيذ الطود سنحصل على النتيجة التالية:



يمكننا التحكم في تنسيق التاريخ بالشكل الذي نريد، بحيث يمكننا تغيير ترتيب الأيام والأشهر والسنوات والساعات والدقائق والثواني، سنورد في المثال التالي مجموعة من التنسيقات وبعدها سنفرد صورة لنتيجة كل تنسيق لكي نفهم معناه:


```
Dim CurrentDate As Date = Now

'عرض التاريخ كاملا
MsgBox(String.Format("{0:d}", CurrentDate))

'عرض اليوم والشهر فقط
MsgBox(String.Format("{0:dd/MM}", CurrentDate))

'عرض اليوم فقط
MsgBox(String.Format("{0:dd}", CurrentDate))
```



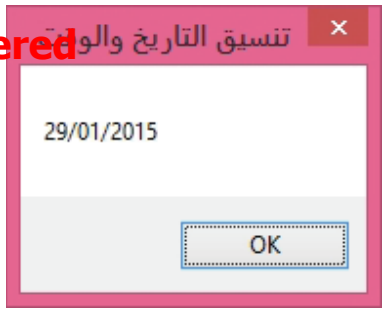
```
'عرض الشهر والسنة فقط
MsgBox(String.Format("{0:MM/yyyy}", CurrentDate))
```

```
        'عرض السنة فقط
        MsgBox(String.Format("{0:yyyy}", CurrentDate))

        'عرض اليوم أولا، الشهر ثانيا، ثم السنة
        MsgBox(String.Format("{0:yyyy/MM/dd}", CurrentDate))

        'عرض التاريخ والوقت كاملين
        MsgBox(String.Format("{0:ss:mm:hh - yyyy/MM/dd}",
CurrentDate))

        'عرض الثواني فقط
        MsgBox(String.Format("{0:ss}", CurrentDate))

        'عرض الدقائق فقط
        MsgBox(String.Format("{0:mm}", CurrentDate))

        'عرض الساعات فقط
        MsgBox(String.Format("{0:hh}", CurrentDate))

        'عرض الوقت كاملا
        MsgBox(String.Format("{0:ss:mm:hh}", CurrentDate))

        'عرض الدقائق والساعات
        MsgBox(String.Format("{0:mm:hh}", CurrentDate))
```


الأمر التالي:

```
        'عرض التاريخ كاملا
        MsgBox(String.Format("{0:d}", CurrentDate))
```

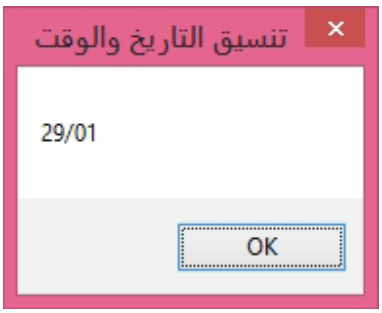
نتيجته:



الأمر التالي:

```
'عرض اليوم والشهر فقط
MsgBox(String.Format("{0:dd/MM}", CurrentDate))
```

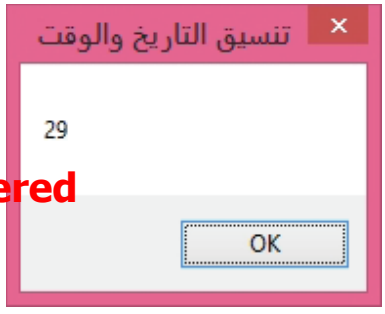

نتيجته:



الأمر التالي:

```
'عرض اليوم فقط
MsgBox(String.Format("{0:dd}", CurrentDate))
```

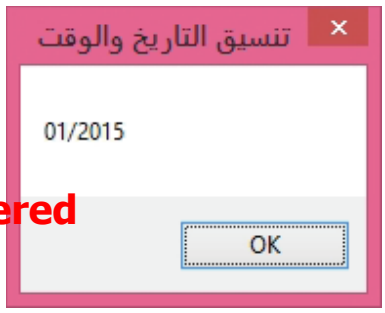
نتيجته:



الأمر التالي:


```
'عرض الشهر والسنة فقط
MsgBox(String.Format("{0:MM/yyyy}", CurrentDate))
```

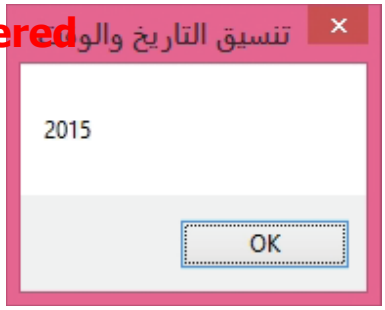
نتيجته:



الأمر التالي:

```
'عرض السنة فقط
MsgBox(String.Format("{0:yyyy}", CurrentDate))
```

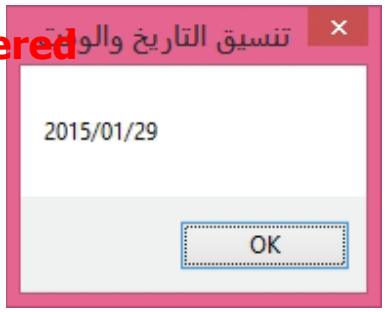
نتيجته:



الأمر التالي:

```
'عرض اليوم أولا، الشهر ثانيا، ثم السنة
MsgBox(String.Format("{0:yyyy/MM/dd}", CurrentDate))
```

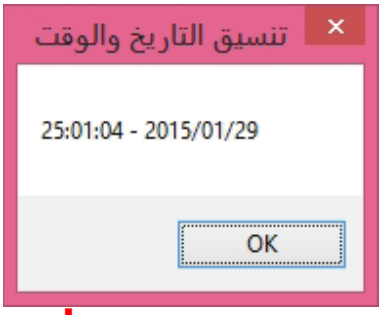

نتيجته:



الأمر التالي:

```
        'عرض التاريخ والوقت كاملين
        MsgBox(String.Format("{0:ss:mm:hh - yyyy/MM/dd}",
CurrentDate))
```

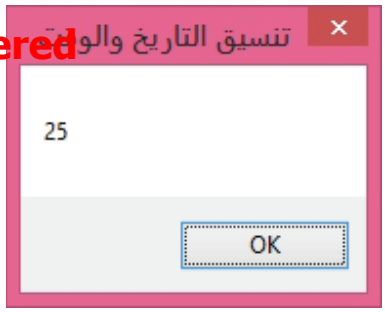

نتيجته:



الأمر التالي:

```
        'عرض الثواني فقط
        MsgBox(String.Format("{0:ss}", CurrentDate))
```

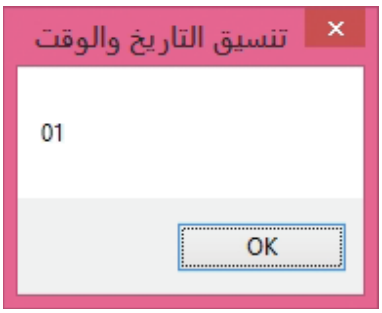
نتيجته:



الأمر التالي:

```
'عرض الدقائق فقط
MsgBox(String.Format("{0:mm}", CurrentDate))
```

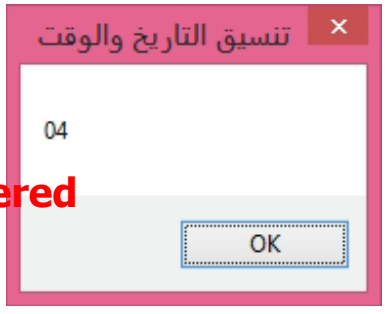

نتيجته:



الأمر التالي:

```
'عرض الساعات فقط
MsgBox(String.Format("{0:hh}", CurrentDate))
```

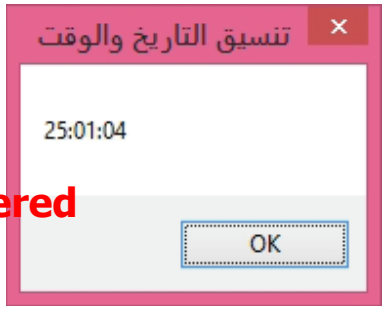
نتيجته:



الأمر التالي:


```
'عرض الوقت كاملا
MsgBox(String.Format("{0:ss:mm:hh}", CurrentDate))
```

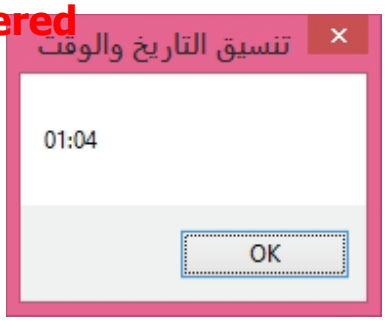
نتيجته:



الأمر التالي:

```
'عرض الدقائق والساعات
MsgBox(String.Format("{0:mm:hh}", CurrentDate))
```

نتيجته:



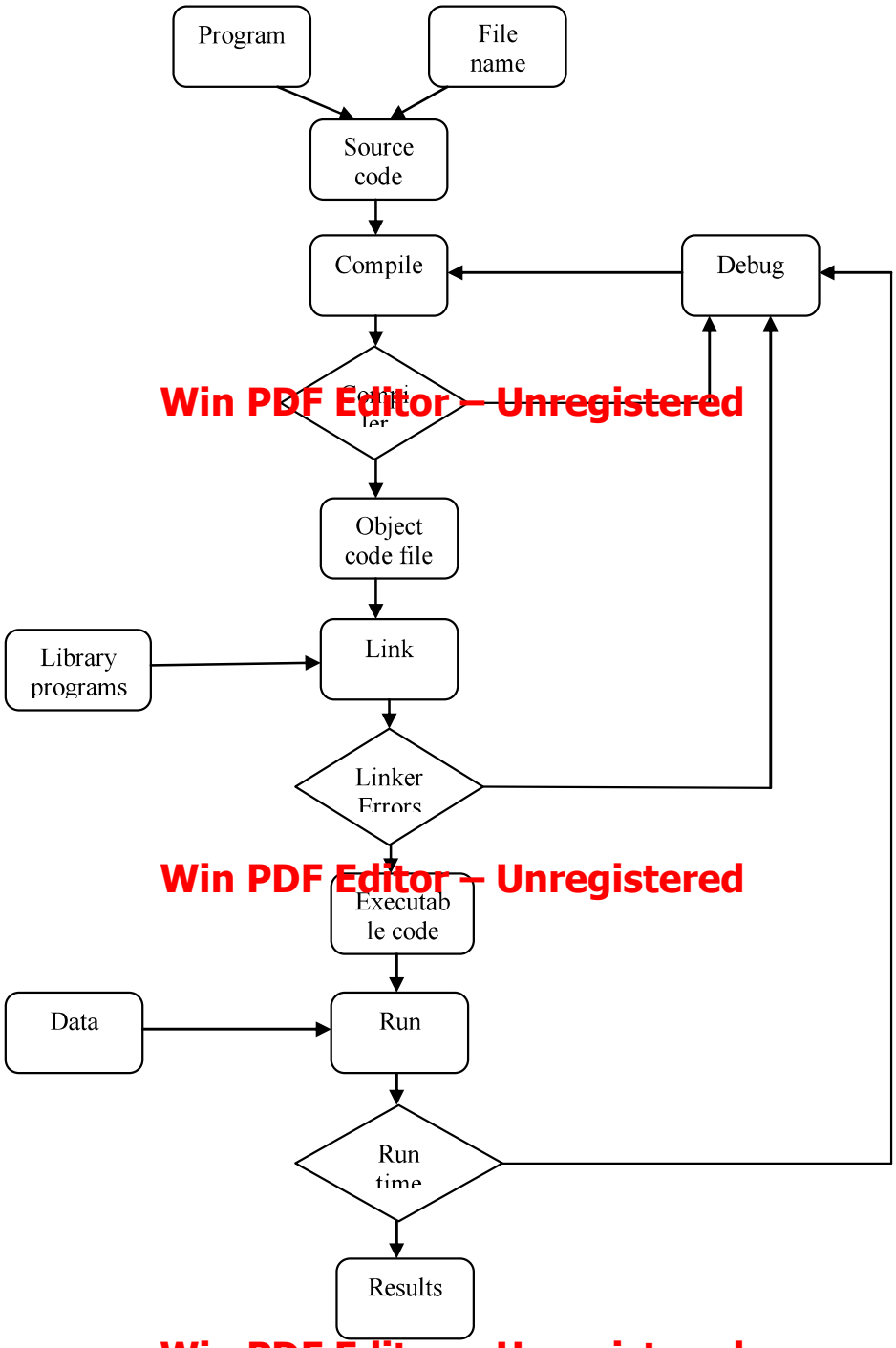
chapter 2

C++

صممت C++ كجسر بين المبرمج والحاسوب. الفكرة بجعل المبرمج ينظم البرنامج بطريقة (هو/ هي) يفهمة بسهولة. بعدها يقوم المترجم (هو برنامج واجبه تحويل اللغة العليا الى اللغة التي يتعامل بها الحاسوب) بنقل اللغة (البرنامج) الى صيغة تستطيع الماكنة استخدامها (التعامل معها). برنامج الحاسوب يتكون من جزئين: هيكل البيانات والايعازات. يفرض الحاسوب او لايفرض القليل من التنظيم على هذين الجزئين. بعد هذا كله فان الحواسيب مصممة لان تكون عامة قدر الأمكان. البيانات في الحاسوب تخزن كسلسلة من البايتات و C++ تنظم هذه البايتات ببيانات مفيدة. الأعلان عن البيانات تستخدم من قبل المبرمج لوصف المعلومات التي (هو/هي) يتعامل معها.

برامج C++ تكتب بلغة عليا باستخدام الأحرف، الأرقام، والرموز الأخرى التي نجدها على لوحة المفاتيح. واقعا فان الحواسيب تنفذ البرامج المكتوبة بلغة دنيا تدعى لغة الماكنة (machine code) (والتي هي سلسلة من الأرقام ممثلة بطريقة الصفر، واحد). لذلك، وقبل ان يتم استخدام البرنامج يجب أن يكون هناك عدد من التحويلات. البرامج تبدأ كفكرة في رأس المبرمج. يقوم المبرمج بكتابة افكارة في ملف، يدعى ملف المصدر (source file or source code) مستخدما محرر اللغة. هذا الملف يحول بواسطة المترجم الى (object file). بعدها يستدعي البرنامج الرابط (linker) حيث ياخذ الملف الهدف ليربطة أو يشركة مع روتينات معرفة مسبقا من المكتبة القياسية (standard library) لينتج برنامج قابل للتنفيذ (والذي هو عبارة عن مجموعة من ايعازات لغة الماكنة). الشكل (1.1) يبين خطوات تحويل البرنامج المكتوب بلغة عليا إلى برنامج قابل للتنفيذ.

في لغة البرمجة C++ فأن البرنامج هو **تجميع للدوال**. والبرامج البسيطة تحتوي على دالة واحدة فقط هي ((main)) وعادة فأن التنفيذ يبدأ عند (main) حيث أن جميع البرامج بلغة C++ يجب أن تحتوي على الدالة ((main)).



شكل (1.1): خطوات تنفيذ البرنامج

ملاحظة://

كل عبارة في لغة C++ يجب أن تنتهي بفارزة منقوطة عدا بعض الحالات الأستثنائية التي سيشار اليها في حينها.

ملاحظة://

- الايعازات (الأوامر او العبارات statements): تبدو مختلفة في لغات البرمجة المختلفة، ولكن هناك وظائف أو دوال اساسية قليلة تظهر في كل البرامج تقريبا منها:

الأدخال input وهي عملية الحصول على البيانات من لوحة المفاتيح او الملفات او الأجهزه الاخرى.

الأخراج output عرض البيانات على الشاشة او ارسالها الى ملف او الأجهزة الأخرى.

الرياضيات math أنجاز العمليات الرياضية الاساسية مثل الجمع والضرب.

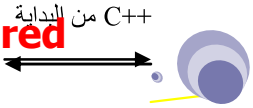
الاختبار testing: تقيم بعض الشروط وتقوم بعض العبارات وفقا لذلك.

التكرار repetition أنجاز بعض الاعمال بشكل متكرر، عادة مع بعض التغيرات.

1.6.1 لماذا لغة C++

C++ هي اللغة الاكثر استخداما في العالم. هذه اللغة لها صفات وخصائص تميزها عن لغات البرمجة الاخرى، واكثر هذه الصفات هي:

- البرمجة الكيانية



امكانية تنظيم البرنامج على شكل كيانات تسمح للمبرمج تصميم تطبيقاته، لتكون اكثر اتصال بين الكيانات بدلا من هيكل الشفرة المتتالية. فضلا عن انها تسمح بامكانيه كبيرة الى اعادة استخدام الشفرة بطرق اكثر منطقية وانتاجيه.

- النقل

بامكانك عمليا ان تترجم نفس شفرة C++ على الاغلب في اي نوع من الحواسيب وانظمة التشغيل دون اجراء تغييرات صعبة.

- الأيجاز

الشفرة التي تكتب بلغة C++ هي قصيرة جدا بالمقارنة مع اللغات الاخرى، حيث يفضل استخدام الرموز الخاصة للكلمات المفتاحية، وهذه تختزل بعض الجهد المبذول من المبرمج.

- برمجة الاجزاء

من الممكن ان تكون تطبيقات C++ من عدد من الملفات لشفرة المصدر والتي تترجم بشكل منفصل، ثم يتم ربطها مع بعض، هذا يساعد على تقليل الوقت وليس من الضروري اعادة ترجمة كامل التطبيق عندما يتم عمل تغيير مفرد ولكن فقط الملف الذي تم تغييره. بالاضافة لذلك، فان هذه الخاصية تسمح بربط شفرة C++ مع الشفرة الناتجة بلغات اخرى مثل المجمع (assembler) او C.

- التوافق مع لغة C

C++ هي البوابة الخلفية للتوافق مع لغة C، اي شفرة تكتب بلغة C سيكون من السهولة تضمينها في برنامج C++ دون الحاجة لاي تغييرات صعبة.

- السرعة

الشفرة الناتجة من تجميع C++ هي كفوءة جدا، وذلك بسبب كونها لغة ثنائية، فهي تعد من اللغات ذات المستوى العالي ومن اللغات ذات المستوى الواطيء فضلا عن صغر حجم اللغة نفسها.

1.7 أوامر المعالج الأولي The C++ Preprocessor Commands

1.7.1 الموجة #include

تعد هذه التعليمة الأشهر والأوسع أستعمالا بعد التعليمة (#define) في لغة C++، عمل هذا الموجة هو أنه يطلب من المعالج الاولي (preprocessor) اضافة محتويات الملف المطلوب مع هذه التعليمة (يذكر أسم هذا الملف بعد #include مباشرة ويكون محدد بين العلامتين (< >)) وحشرة في الملف المصدر، حيث يتم ضم وأحتواء هذا الملف مع ملف البرنامج عند التنفيذ، هذا الملف يدعى ملف التعليمات، ويعود السبب في ذلك الى ان بعض الايعازات داخل البرنامج تحتاج الى تعاريف ودوال يتضمنها هذا الملف.

1.8 المعرفات Identifires

كل البرامج تحتوي على نوعين من الرموز:

النوع الاول.. وهي **الرموز التي تعود الى اللغة**.. تستخدم هذه الرموز بطريقتين أما أن تكون على شكل رمز واحد أو أثنين مثل (:، ;، ()، +، -) أو على شكل كلمات تسمى الكلمات المحجوزه او الكلمات المفتاحية (KeyWords) مثل: **(inline ،do ،while ،else ،if)**

النوع الثاني.. هي **المعرفات** وهي عبارة عن رموز تستخدم في البرامج فأما أن تكون معرفات قياسية مثل **(char, int, float... etc)**, أو أن تكون معرفات يتم أختيارها من قبل المبرمج، وهذه المعرفات الأخيرة نسميها أيضا المتغيرات (Variables)، و**المتغير** هو رمز أو أكثر يستخدم في البرنامج ليشير الى محتوى موقع في الذاكرة.

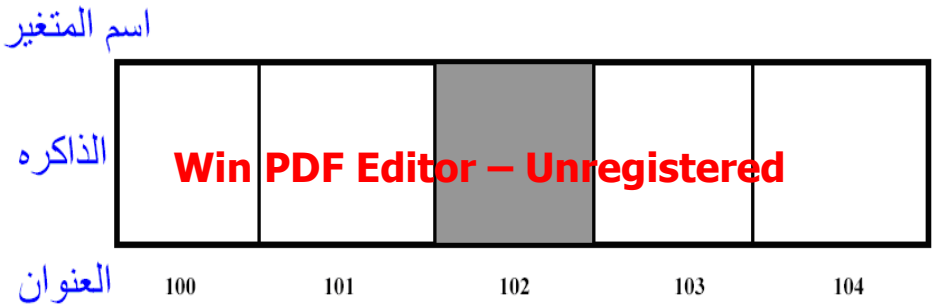
ملاحظة://

المتغير.. في أغلب لغات البرمجة فأن المتغير هو مكان لتخزين المعلومات، المتغير يكون أسم موقع في ذاكرة الحاسوب حيث يمكن تخزين قيمة

بداخلة ثم أمكانية أستعادة هذه القيمة فيما بعد.

والمتغير هو أسم يمثل برقم أو سلسلة حرفية (وممكن حرف واحد أو تعبير منطقي).

من الممكن تصور ذاكرة الجهاز على أنها مجموعة من المواقع التي تخزن فيها المعلومات، هذه المواقع مرقمة بشكل متسلسل تبدأ من الصفر وتنتهي بحجم الذاكرة، تدعى هذه الأرقام عناوين الذاكرة (Addresses)، يمثل أسم المتغير (بطاقة عنونة) ملصقة على أحد المواقع بحيث تستطيع الوصول اليه سريعا دون الحاجة الى معرفة العناوين الحقيقية في الذاكرة (لذا فان المتغير سيشير الى أحد هذه العناوين، وعند حاجتك وضع قيمة في الموقع الذي يشير له هذا المتغير فان المعالج (processor) سيذهب الى العنوان الذي يشير له المتغير ويضع فيه القيمة وكذلك عندما تريد أن تعرف قيمة المتغير فأن المعالج يذهب الى العنوان الذي يشير له المتغير ويقرأ القيمة التي فيه). يعرض الشكل التالي هذه الفكرة والتي تبين بعض المواقع في الذاكرة والتي من الممكن ان يشير اليها المتغير.



شكل رقم (1.2).: بعض مواقع الذاكرة المنطقية

ملاحظة://

لغة C++ تعد حساسة لحالة الأحرف (أي أنها تميز بين الأحرف الكبيرة والصغيرة)، لذلك فأن الحرف الصغير يعد معرفا غير مساوي لشكلة الكبير (أي أن (a) لا يساوي (A)). على ان بعض لغات البرمجة لا تميز بين حالات الاحرف.

تتكون أسماء المتغيرات من " حرف واحد، مجموعة حروف، أو حروف وأرقام ومن الممكن استخدام الشارحة ".. على أن يكون دائما **أول رمز باسم المتغير حرف او شارحة** حتما مثل:

(x, ad, _count, endofpoint, end_of_point, Saad6, x345)

هذه جميعا متغيرات مقبولة.

أما المتغيرات التالية فهي متغيرات غير مقبولة:

(first name, next.word, 15may, Ten%)

والسبب هو أن المتغير الأول يحتوي على فراغ، الثاني يحتوي على نقطة، الثالث يبدأ برقم، أما الأخير فيحتوي على رمز لايمكن استخدامة مع المتغيرات.. وهذه جميعها غير مقبولة في البرنامج. أن أختيار المتغير من قبل المبرمج تعد مسألة مهمة ويفضل أن يعكس المتغير المعنى الذي يستخدم لأجلة المتغير فمثلا يفضل استخدام المتغير (sum) مع الجمع وأذا ما استخدم متغير أخر فان ذلك سوف لا يؤدي الى أي خطأ، وكذلك يفضل أن لا يكون المتغير طويل فمثلا يفضل استخدام متغير متكون من حرف واحد عندما نستخدمة في برنامج قصير ولا يتكرر كثيرا، أما استخدام متغير من حرف واحد ويستخدم بشكل متكرر وبأجزاء متكررة في برنامج طويل فأنه يعد أختيارا سيئا بالرغم من أنه لا يعيق عمل البرنامج.

1.9 البيانات Data

كل عنصر من البيانات في البرنامج أما أن تكون قيمتة ثابتة أو متغيرة (قيمة المتغير ربما تتغير خلال تنفيذ البرنامج). كل متغير (والذي هو بيانات) في البرنامج يجب أن يكون له نوع وبموجب هذا النوع سيتم تحديد المساحة الخزنية اللازمة لقيمة هذا المتغير، وكذلك تحدد العمليات التي ممكن أجراؤها على هذا المتغير (تحدد لكل نوع عدد البايتات في الذاكرة التي تحجز لخزن قيم ذلك النوع وعند الكتابة في هذا الموقع فان الكتابة ستحدد بعدد بايتات هذا النوع أي لا يتم تجاوز هذا العدد من البايتات حتى وان كانت القيمة تتجاوز الحدود العليا والدنيا لهذا

النوع، وعند القراءة فانه سيتم قراءة القيم الموجودة في هذه البايتات فقط وبذلك تتجنب الخطأ في القراءة والكتابة). والأنواع القياسية التي تستخدم في لغة C++ هي:

1.9.1 الاعداد الصحيحة Integers

الأعداد الصحيحة هي كل الأعداد الموجبة والسالبة التي لا تحتوي على كسر. فالصفر عدد صحيح و 123 هو عدد صحيح و 45- أيضا عدد صحيح. أما (123.345 و 1.45-) فهي ليست أعداد صحيحة.

أن أي محاولة لاستخدام قيم خارج نطاق الحدود العليا والدنيا للأعداد الصحيحة سيؤدي الى حدوث خطأ. وبشكل عام فأن المتغيرات من نوع الأعداد الصحيحة تستخدم اضافة الى العمليات الرياضية في العدادات والفهارس.

العلاقات الرياضية التي تستخدم مع الأعداد الصحيحة هي(+ , - , * , / , %) وهي على التوالي (الجمع، الطرح، الضرب، القسمة، وحساب باقي القسمة).

أمثلة://

21 / 3 = 7

9 / 2 = 4

2+3*4 = 14

هنا ينفذ داخل القوس أولا

(2+3) * 4 = 20

5 % 2 = 1

7 % 4 = 3

ويصرح عن الأعداد الصحيحة بلغة C++ في أي مكان داخل جسم البرنامج بالمعرف (int) والتي تعني (integer) وهي تكتب قبل المتغيرات، مثال

int x ;

ملاحظة://

نتيجة قسمة عدد صحيح على عدد صحيح أخر هو عدد صحيح.

أما أذا كان أحد العددين هو حقيقي فأن النتيجة ستكون عددا حقيقيا، مثال

2.0 / 3 = 0.66666667

50 / 2.0 = 25.0

ملاحظة://

فضلا عن الارقام العشرية (وهي التي أساسها عشرة والتي تستخدم بالأعمال الأعتيادية (0 ..9))، فان C++ تسمح لك باستخدام ثوابت من الأرقام وفق النظام الثماني (octal numbers) (أساسها 8) وكذلك أرقام وفق النظام السادس عشر (hexadecimal) (أساسها 16). ولتنفيذ ذلك فاذا أردت تمثيل رقم بالنظام الثماني فضع (0) (صفر) أمام الرقم للدلالة على أنه بالنظام الثماني، أما أذا وضعت (0x) (صفر ثم x) أمام الرقم فذلك يعني أن الرقم ممثل بالنظام السادس عشر. المثال اللاحق يمثل ثوابت بالانظمة الثلاثة وكل منها مكافيء للأخر (جميعها تمثل الرقم 75 وهو خمسة وسبعون).

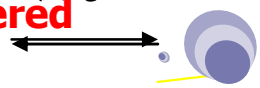
75 // نظام عشري

0113 // نظام ثماني

0x4b // نظام سادس عشر

جدول (1.1): أنواع الأعداد الصحيحة وحجمها بالبتات

انواع البيانات	المـــــــــــدى	الحجم بالبتات
short	-32767...32767	16



int	-32767...32767	16
long	-2147483647... 2147483647	32
unsigned short	0...65535	16
unsigned	0...65535	16
unsigned long	0...4294967295	32

1.9.2 الأعداد الحقيقية Real Numbers

وهي الأعداد التي تحتوي على كسور مثل

(10.0 , -356.67890 , 12.5 , 0.03). الأعداد الحقيقية ممكن ان تمثل بعدد صحيح وفارزة (تستخدم نقطة لتفصل العدد الصحيح عن الجزء الكسري)، وممكن أن تستخدم الرمز (e) والذي يمثل الرقم عشرة مرفوعا الى أس معين (الأس هو الرقم الذي يلي الحرف (e)) (الرقم الذي يلي الحرف (e) يجب ان يكون عددا صحيحا)، مثال

3.14159	//	=	3 .14159
6.02e23	//	=	6.02 x 10^23
1.6e-19	//	=	1.6 x 10^ -19
3.0	//	=	3.0

المثال أعلاه يحتوي على أربعة نماذج من الأرقام الحقيقية المقبولة في C++. العدد الاول يمثل (PI) (النسبه الثابتة) اما الثاني فهو يمثل عدد افوكادرو، الثالث يمثل الشحنة الكهربائية للألكترون (وهو عدد صغير جدا) وكل هذه الاعداد هي تقريبية، اما العدد الأخير فهو يمثل الرقم (3) ولكن كعدد حقيقي.

أما العمليات الرياضية التي ممكن أجراؤها على الأعداد الحقيقية فهي (/ , * , - , +) وهي على التوالي (الجمع، الطرح، الضرب، القسمة). ويصرح عن الأعداد الحقيقية في لغة البرمجة C++ في أي مكان داخل جسم البرنامج بالمعرف (float) التي تسبق المتغيرات، **مثال**


```
float x;
```

ملاحظة://

تمثل الأرقام بطريقتين فأما أرقام صحيحة بدون كسر أو أرقام كسرية.

القواعد التالية تطبق عند كتابة أرقام في الحاسوب:

1. الفارزة (,) لا يمكن أن تظهر في أي مكان في الرقم.
2. ممكن أن تسبق الأرقام أحد العلامتين (- ، +) للدلالة على كون الرقم موجب أو سالب (يعد الرقم موجبا أذا لم تظهر أي من العلامتين على يسارة).
3. يمكن تمثيل الأرقام بطريقة العلامة العلمية (وذلك بأستبدال الرقم (10) بالحرف (e)). مثلا الرقم (2.7×10^{-6}) يكتب حسب العلامة العلمية كما يلي (2.7 e -6). كذلك فأن العدد (6×10^{12}) يمكن ان يمثل حسب العلامة العلمية كما يلي (6 e 12), وكما وضح اعلاه.

ملاحظة://

يفضل عند استخدام التعريف (long) وضع حرف (L) بعد القيمة فمثلا :

```
long  SunDistance = 93000000 ;
```

هنا ستنتج قيمة خاطئة (1864704) ويعطي المترجم رسالة تحذير ولتجنب ذلك تكتب كما يلي :

```
long SunDistance = 93000000L ;
```


ملاحظة://

أدناه بعض القواعد المهمة التي يجب أن تراعى عند كتابة العلاقات الرياضية :

أن وضع أشارة السالب قبل المتغيرات تكافئ ضرب المتغير بالقيمة

(-1). مثلا المتغيرات (x+y) - من الممكن أن تكتب ((x+y) * (-1).

يجب أن تكتب العلاقات الرياضية وفقا للطريقة التي تحددها لغة البرمجة C++ بحيث تذكر كل العلامات الرياضية دون أختصار. مثال : العلاقه الرياضية الأتية غير مقبولة ((3x2 + x1) 2) هذه العلاقه لكي تكون مقبولة في لغة البرمجة C++ يجب أن تكتب بالشكل التالي: ((2 * x1 + 3 * x2)) العلاقة الأولى هي التي تعودنا على استخدامها في الرياضيات.

العدد المرفوع الى قيمة معينة سيضرب بنفسة عدد من المرات بقدر الأس اذا كان الاس عددا صحيحا ولا يهم فيما اذا كان الأساس سالبا أو موجبا.

لايجوز رفع القيمة السالبة الى أس كسري (وذلك لأن حساب ناتج الرقم المرفوع الى أس كسري يتم بحساب اللوغاريثم للأساس، ويضرب هذا اللوغاريثم بالأس، وعندها يحسب معكوس اللوغاريثم، وأن اللوغاريثم للرقم السالب غير معرف لذا لايمكن أيجاد النتيجة).

العمليات الرياضية لايمكن أجراؤها على السلاسل الرمزية. مثال (34 + "xyz") هذا غير مقبول وذلك لأن (xyz) هو سلسلة حرفية وليس عددا أو متغيرا رقمي (لاحظ أنه محصور بين علامتي أقتباس (quotation mark) للدلالة على أنه سلسلة حرفية).

جدول (1.2): أنواع الأعداد الحقيقية وحجومها بالبتات

نوع البيانات	المـــــــــــــــدى	الحجم بالبتات
float	$3.4x10^{-38}$. $3.4x10^{+38}$	32
double	$1.7x10^{-308}$. $1.7x10^{+308}$	64
long double	$3.4x10^{-4932}$.. $1.1x10^{+4932}$	80

1.9.3 الرموز Characters

وهي كافة الرموز التي تستخدم في الحاسوب والتي غالبا ما نجدها على لوحة المفاتيح والتي تشمل الحروف الأبجدية سواء كانت حروف كبيرة (**A..Z**) أو حروفا صغيرة (**a..z**)، الأرقام (**0..9**)، الرموز الاخرى التي نراها على لوحة المفاتيح مثل (etc ... ? , & , % , ! , + , / , .) وتستخدم بشكل مفرد. ويصرح عن الرموز بلغة البرمجة C++ في أي مكان داخل جسم البرنامج بالمعرف (char) التي تسبق المتغيرات.

أن أكثر مجاميع الحروف استخداما هما أثنان:

ASCII

(American Standard Code for Information International)

EBCDIC

(Extended Binary Coded Decimal Information Code)

وكل منهم له صفاتة الخاصة به (لمزيد من المعلومات راجع الملاحق في نهاية الكتاب).

ملاحظة://

تكتب الحروف بين علامتي أقتباس مفردة (' ').

- **عمليات الأحرف**

الأحرف تمثل داخل الحاسوب بواسطة أرقام صحيحة وفقا لنظام (ASCII) تسمى الأعداد الترتيبية (ordinal numbers)، لذا فأن المبرمج بأمكانة أن يمزج بين الرموز والأعداد الصحيحة [illegible]

أذا فرضنا أن المتغير الرمزي (ch) هو متغير من نوع حروف وتم أسناد قيمة له كما يأتي

(ch = 'S')

عليه فأن التعبير التالي ; ch = ch + 1

سيؤدي الى أن تكون قيمة المتغير الرمزي (ch) تساوي الرمز (' T ')، وكذلك فأن التعبير التالي ch = ch – 3

سيؤدي الى أن تكون قيمة المتغير الرمزي (ch) تساوي الرمز (' P ') وهذا يعتمد على القيم الرقمية التي تمثل الاحرف بنظام (ASCII).

ملاحظة://

الفرق العددي بين تمثيل الأرقام الكبيرة والأرقام الصغيرة هو (32) (اي ان الحرف الصغير اكبر من الحرف الكبير بالقيمة 32).

فمثلا أن قيمة الرمز (A = 65) حسب نظام (ASCII) بينما قيمة الرمز (a = 97) وفقا لنفس النظام. عليه فأذا كانت

ch = ' E ' ;

أذن ch = ch + 32;

ستؤدي الى أن تكون قيمة المتغير الرمزي (ch = ' e ')

وكذلك أذا كانت قيمة المتغير الرمزي ch = ' d ' ;

فأن ch = ch – 32;

ستؤدي الى أن تكون قيمة المتغير الرمزي (ch = ' D ')

العدد الترتيبي للصفر هو (48) لذا فأن الاعداد (0..9) تأخذ الأعداد الترتيبية (48 – 57)

ملاحظة://

الرموز تحدد بعلامة اقتباس مفردة مثل (‘ 5 ‘) او (‘ { ‘) اما السلاسل الرمزية فهي تحدد بعلامة اقتباس مزدوجة مثل (“ 51 “) او (“ good “) بينما الارقام لاتحدد باي علامة مثل (5) او (456).

1.9.3.1 رموز الدلالة Directing Characters

وهي حروف خاصة عادة تستخدم مع الشرطة العكسية (/) لاعطاء تأثير معين يلاحظ ضمن مخرجات البرنامج. الجدول (1.3) يبين هذه الرموز مع التاثير الذي تحدثة.

وهذه تسمى ايضا سلاسل الهروب Escape Sequences. فالشارطة المعكوسة (\) التي تسبق بعض الاحرف تخبر المترجم بان هذا الحرف الذي يلي الشارطة المعكوسة ليس له نفس المعنى كما لو ظهر الحرف بنفسة دون هذه الشارطة المعكوسة (\). هذه السلاسل يتم كتابتها كرمزين دون وجود فراغ بينهما. بعض هذه السلاسل معرفة في C++.

اذا وضعت (\) او (“) في سلسلة حرفية ثابتة، فانك يجب ان تهرب من قدرة (“) على انهاء سلسلة حرفية ثابتة وذلك باستخدام (\”)، او قدرة (\) للهرب باستخدام (\\). لاحظ انها (\\) تعني المترجم ان تعرض شارطة معكوسة حقيقية (\)، وليست شارطة معكوسة لسلسلة هروب، وان (\”) تعني حاصرة حقيقية وليس نهاية سلسلة ثابتة.

لاحظ دائما تستخدم سلاسل الهروب مع حاصرتين مزدوجتين مثل (“\n”).

جدول (1.3): رموز الدلالة في لغة C++

الرمز	الناتج (التأثير على المخرجات)
\a	صوت أو صفير (Beep)

\b	التراجع خطوة واحدة للخلف (Backspace)
\f	التغذية (form feed)
\n	سطر جديد (new line)
\r	الاعادة او الرجوع (carriage return)
\t	الازاحة الأفقية (horizontal tabulator)
\v	الازاحة العمودية (vertical tabulator)
\\	الشرطة المعكوسة (Backslash)
\'	حاصرة مفردة (single quota)
\"	حاصرة مزدوجة(double quota)

1.9.4 النوع المنطقي (Boolean)

النوع الاخر هو النوع المنطقي والذي يرمز له (bool). هذا النوع اضيف حديثا الى لغة C++ بواسطة هيئة (ISO\ANSI) (منظمة المقاييس العالمية/ منظمة المقاييس الامريكية الوطنية).

التعابير المنطقية تشير الى واحدة من القيم وهي (صح، او خطأ). التعابير المنطقية تستخدم في التفرع او حلقات التكرار والتي سندرسها لاحقا.

1.10 التعابير المنطقية The Boolean Expressions

وهي التعابير التي تمثل نتيجتها بحالة واحدة من أثنتين وهما (صح أو خطأ **(true OR false)**، وهناك ثلاث عوامل منطقية وهي **(Not، Or، And)**.

التعبير المنطقي يعيد القيمة (1) عندما يكون التعبير (TRUE) والقيمة (0) عندما يكون التعبير (FALSE). وهي تستخدم لوصف أي تعبير فيما أذا كان صح أو خطأ. أن أنواع المتغيرات التي تستخدم لهذا الغرض يصرح عنها في حقل المتغيرات بالدالة (bool) (هذه عادة لاتجدها في جميع نسخ C++ وانما النسخ الحديثة فقط).

فمثلا عندما نعرف العبارة التالية على انها من نوع القيم المنطقية كمايأتي

```
bool  c = (a= =b) ;
```

نلاحظ هنا اننا استخدمنا علامة المساواة للدلالة على ان نتيجة الطرف الايمن ستؤول الى المتغير في الطرف الأيسر بينما استخدمنا العلامة (==) وهي تستخدم لعمليات **فحص المساواة**, فاذا كان (a ، b) متساويان فان (c) ستكون قيمتها تساوي (true) وبخلاف ذلك تكون قيمتها تساوي (false).

1.11.1 العمليات المنطقية Logical Operators

هناك ثلاثة أنواع من العمليات المنطقية وهي (**NOT ، OR ، AND**) كل منها يتعامل مع التعابير الشرطية (أي التي تحتوي شرط). كل واحد من هذه التعابير له تأثير مختلف على التعابير الشرطية. أدناه أمثلة تبين كيفية استخدام هذه التعابير والتي من الممكن أن تستخدم بين تعبيرين أو أكثر من التعابير الشرطية.

AND

العامل (&&) يستخدم للدلالة على العامل المنطقي (and) في لغة C++ وهو يستخدم لمقارنة تعبيرين لتحصل على نتيجة منطقية مفردة، والنتيجة التي تحصل عليها تحدد بجدول الصدق (1.4) أدناه.

جدول (1.4): جدول الصدق للعامل (و) (&&) (And)

A	*B*	*A && B*

A	B	A && B
true	true	True
true	false	false
false	true	false
false	false	false

OR

العامل (||) يستخدم للدلالة على العامل المنطقي (or) في لغة C++ وهو يستخدم لمقارنة تعبيرين لتحصل على نتيجة منطقية مفردة، والنتيجة التي تحصل عليها تحدد بجدول الصدق (1.5) ادناه:

جدول (1.5): جدول الصدق للعامل (أو) (||) (Or)

A	B	A || B
true	true	True
true	false	True
false	true	True
false	false	False

((5 = = 5) && (3 > 6)) // النتيجة خطأ (صح && خطأ)

((5 = = 5) || (3 > 6)) // النتيجة صح (صح || خطأ)

NOT

لاحظ في لغة C++ فان العامل (!) يمثل العامل (لا) (not) وهو يأخذ معامل واحد يتواجد في يمينة والعمل الوحيد الذي يقوم به هو عكس قيمتة (قيمة المعامل الذي على يمينة) فاذا كانت قيمتة (صح) تصبح خطأ واذا كانت خطأ تصبح صح. نتيجة استخدام العامل (لا) موضحه بالجدول (1.6)

جدول (1.6): جدول الصدق للعامل (لا) (!) (Not)

A	! A
true	False
false	True

مثال://

!(5==5) // النتيجة تصبح خطأ لان التعبير (5==5) هو صح

!(6<=4) // النتيجة تصبح صح لان (6<=4) هي خطأ

!true // النتيجة تصبح خطأ

!false // النتيجة تصبح صح

ملاحظة://

من الممكن ان تستخدم عوامل العلاقات المنطقية للمقارنة بين قيمتين ومن الممكن ان تكون هذه القيم من أي نوع من أنواع البيانات مثل (float, int, char…etc)، او ممكن أن تكون (كما سنرى لاحقا) اصنافا معرفة من المستخدم

أن نتيجة المقارنة أما أن تكون (صح او خطا) (true ،false). فمثلا العبارة التالية

```
cout <<  5  <  23 ;
```

ستطبع القيمة (1) لان العبارة صحيحة.. اما العبارة التالية

```
cout <<  45  > 60 ;
```

ستطبع القيمة (0) لان النتيجة خاطئة

ملاحظة://

العامل (NOT) يختلف عن العاملين السابقين اذ أنه يتقبل مدخلا واحدا ودائما يعكس حالة العبارة التي يدخل عليها فأذا كانت صحيحة يجعلها خاطئه وأن كانت خاطئة يجعلها صحيحة.

ملاحظة://

أن أسناد قيمة لمتغير من نوع معين خارج المدى المحدد له سيؤدي الى حدوث خطأ، وهذا الخطأ أما أن يوقف التنفيذ أو يؤدي الى ظهور نتائج غير متوقعة.

1.11 الأعلان عن المتغيرات Declarations

يتم الاعلان عن المتغير وذلك بان يتم كتابة النوع أولا ثم يتبع ذلك اسم المتغير والذي يجب ان يخضع للقواعد المذكورm انفا فمثلا:

```
int  a;
float  mynum ;
```

وبالأمكان الأعلان عن أكثر من متغير من ذات النوع بنفس الطريقة أعلاه على أن تفصل فارزه بين اسم متغير وأخر، مثال:


```
int  x ،y ،z ;
```

وهذه تكافيء الأعلان التالي

```
int  x ;
int  y ;
int  z ;
```

الطريقتان صحيحتان والفرق هو ان الأولى أكثر اختصارا.

ملاحظة://

بالامكان استخدام (signed ،unsigned) لوحدهم، وتعني انها من نوع الاعداد الصحيحة مثال

```
unsigned   nextpage ;
unsigned int   nextpage ;
```

العبارتان متكافأتان

1.12 الثوابت Constants

في بعض البرامج تحتاج الى استخدام قيم ربما تكون معروفة مسبقا قبل تنفيذ البرنامج ولا يمكن أن تتغير داخل البرنامج مثل النسبة الثابتة (Л) والتي قيمتها (3.1415926585) هذه القيم الثابتة سواء كانت ذات قيمة معروفة مسبقا أو أي قيمة ممكن أن تسند الى متغير، جميعها ممكن أن يعلن عنها في أي مكان من جسم البرنامج وباحدى الطرق التالية، الأعلان عنها (باستخدام الكلمة المفتاحية (const)، استخدام الكلمة المفتاحية (enum)، أو باستخدام الموجة (#define)) والتي تسبق أنواع البيانات للمعرف المراد تعريف قيمتة على انها ثابتة.

ملاحظة://

المعرفات التي تعرف على أنها ثوابت لا يمكن ان تتغير قيمها أثناء تنفيذ البرنامج بأي شكل من الأشكال.

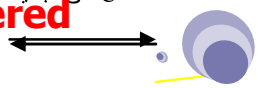
- const

وهي تسبق انواع البيانات لتعرف واحد او أكثر من المتغيرات على أنها ثابتة وفقا للصيغة القواعدية التالية:

```
const   TYPE   variable_name = value ;
```

مثال:

```
const   float   Pi = 3.1413926535 ;
```



```
const    string  Error = ' Run_Time Error ' ;
```

- **Enum**

وهي تستخدم لتعريف قائمة من المتغيرات على أنها ثابتة وفقا للصيغة القواعدية التالية:

enum TYPE *{CONSTANT1=value ،CONSTANT2 = value,...}*;

وسناتي عليها لاحقا لتوضيح عملها بشكل اكثر تفصيلا

- **الموجة (التعليمة) #define**

وهي تقوم بتعريف رموز كثوابت، وبالرغم من عدم شيوع استخدام هذا الهيكل في لغة (C++)، ولكن بالامكان استخدامة لتعريف المتغيرات الحسابية أو الرمزية في بداية البرنامج وتعوض قيمتها الحسابية أو الرمزية في أي مكان تذكر فيه هذه الأسماء في البرنامج وتستخدم الحروف الأبجدية الكبيرة عادة لتعريف أسماء هذه المتغيرات. مثال:

```
#define    TRUE  1
#define    PI    3.1415927
#define    EOF   -1
```


ملاحظة://

هذا الهيكل شائع في لغة (C)، وان كل ما موجود في لغة (C) ممكن استخدامة في لغة C++ .. العكس ليس صحيح

ملاحظة://

من الممكن الاستعاضة عن (#define) بالكلمة المفتاحية (const) مثال

```
const    TRUE = 1
const    PI = 3.1415927
```

مع ملاحظة استخدام علامة المساواة

1.12.1 أسباب استخدام الثوابت:

- أذا كان هناك عدد يستخدم بشكل متكرر داخل البرنامج فأن المبرمج يفضل أن يصفة بأسم يشير اليه على أن يحمل قيمة ثابتة.

- من الممكن استخدام الثوابت لتسمية متغيرات من نوع السلاسل الرمزية والتي تستخدم بشكل متكرر في مخرجات البرنامج وهي في جميع الأحوال تستخدم لتسهيل العمل البرمجي.

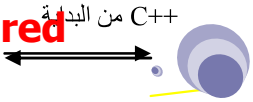
مثال:

نفرض أننا نحتاج الى طباعة أسم جامعة مثلا بشكل متكرر في البرنامج، ممكن أن نقوم بمايأتي:

```
const  string  University  = "Al _ Kufa University " ;
const  string  Underline   = "-------------------------" ;
```


الأن من الممكن استخدام الأسماء المعرفة كثوابت في البرنامج وكما يأتي:

```
cout << University << endl ;
cout <<  Underline  ;
```



ملاحظة://

يستخدم تعريف الثابت في أي مكان داخل جسم البرنامج، وان أي محاولة لتغيير قيمة أثناء تنفيذ البرنامج سيؤدي الى صدور رسالة خطأ.

1.13 العوامل Operotors

عند وجود المتغيرات والثوابت، فبامكانك القيام بالعديد من العمليات عليها مستخدما العوامل المناسبة لكل عملية.. منها:

1.13.1 عامل التخصيص (= Assignment)

عامل التخصيص واجبة اسناد قيمة الى متغير مثل

```
A = 7 ;
```

هنا تم أسناد القيمة (7) الى المتغير (A) ودائما تسند القيمة في الجانب الأيمن من عامل التخصيص الى المتغير في الجانب الأيسر من التخصيص.

تختلف C++ عن اللغات الأخرى بأمكانية استخدام علامة التخصيص في الجانب الأيمن أو ان تكون جزء من الجانب الأيمن لعملية تخصيص أخرى مثال

```
A = 8 + (b = 4) ;
```


وهي تكافيء العبارات التالية

```
b = 4 ;
A = 8 + b ;
```

كذلك فان التعبير التالي مقبول أيضا

```
A = b = c = d = 6 ;
```

1.13.2 العمليات الرياضية ARITHMETIC OPERATORS

(+ , - , * , / , %)

وهي العمليات المعروفة لنا في الرياضيات، والتي هي (الجمع، الطرح، الضرب، والقسمة)، يضاف لها عامل أخر وهو أستخراج باقي القسمة باستخدام العلامة (%) الجدول (1.7) يبين هذه العمليات:

جدول (1.7): يبين العمليات الرياضية التي تدعمها لغة C++

العملية الرياضية	العامل
الجمع Addition	+
الطرح Subtraction	-
الضرب Multiplication	*
القسمة Division	/
أستخراج باقي القسمة Modulo	%

1.13.3 المساواة المركبة Compound Assignment

وهي استخدام المساواة مع عوامل اخرى

(+ = ، - = ، * = ، / = ، % = ، > = ، < = ، & = ، | =)

عندما نرغب بتحوير قيمة متغير بأنجاز عمليات رياضية على القيمة المخزونة حاليا بالموقع الذي يشير له المتغير فاننا يمكن ان نستخدم عوامل المساواة المركبة، هذه العمليات تستخدم بطريقة مختلفة عن العمليات المتعارف عليها حيث ان العوامل الموجودة مع المساواة هي جميعا عوامل ثنائية أي تستخدم مع أثنين من المتغيرات أو القيم، وجميعها تستخدم وفقا للقاعده التالية:

حيث يستخدم العامل على الجانب الأيسر من المساواة لأجراء العملية الرياضية أو المنطقية بين المتغير في الجانب الأيسر من المساواة مع المتغير أو القيمة على الجانب الأيمن من المساواة، وتسند النتيجة الى المتغير الذي في الجانب الأيسر من المساواة. مثال يوضح ذلك في الجدول (1.8):

جدول (1.8): أمثله توضح استخدام المساواة المركبة

التعبير	المكافيء له
value += increase;	value = value + increase;
a - = 5;	a = a - 5;
a /= b;	a = a / b;
price *= units + 1;	price = price * (units + 1);

ملاحظة://

لايجوز ان يكون في الطرف الايسر من (المساواة) تعبير وأنما يكون متغير ومتغير واحد فقط.

1.12.4 الفاصلة (,) كأداة The comma (,) operator

وهي أداة ثنائية (binary) وتحتل الاسبقية الأخيرة في سلم أسبقيات الأدوات المختلفة، وتأخذ الصيغة العامة:

Expression1 ، Expression2

وتستخدم لفصل تعبيرين على يمين المساواة، فعند استخدام فاصلة لتفصل بين تعبيرين، فأن تسلسل العمليات يأخذ الترتيب التالي:

1. تستخرج قيمة التعبير الأول الذي على يسار الفاصلة (الفارزة) ثم تسند للتعبير الثاني على يمين الفاصلة (الفارزة).
2. تستخرج قيمة التعبير الثاني الذي على يمين الفاصلة (الفارزة) كقيمة نهائية لكامل التعبير.

مثال:

A = (b = 2 ، b+1) ;

في هذا المثال سيعمل المترجم على يمين المساواة كما هو متعارف، اذ سيسند القيمة (2) الى المتغير (b) (يبدأ أولا بالتعبير الذي على يسار الفاصلة)،

المرحلة الثانية, العمل على التعبير الذي موجود على يمين الفاصلة في هذه الحالة فأن قيمة (b) هي (2) ومنها يستخرج القيمة النهائيه للتعبير (b+1) لتكون النتيجة هي (3) وهي تمثل نتيجة التعبيرين على يمين المساواة والتي ستسند الى المتغير (A) على يسار المساواة.

1.13.5 عوامل المساواة والعلائق Relation And Equality Opetotors

وتستخدم هذه العوامل لأغراض المقارنة، وهي (==، =!، <، >، =>، =<) والجدول (1.9) يوضح استخدام هذه العوامل.

جدول (1.9): عوامل المساواة والمقارنة المستخدمة في لغة C++

العامل	استخدامة
= =	تساوي
!=	لا تساوي
>	أكبر من
<	أصغر من
>=	أكبر من أو تساوي
<=	أصغر من أو تساوي

لغرض المقارنة بين متغيرين فانه يتم استخدام عوامل العلاقات والمساواة. نتيجة عملية المقارنة هي قيمة منطقية (Boolean) اي (صح او خطأ) وفقا للنتيجة. **مثال**

```
(7 = = 5)      //  النتيجة خطأ
(5 > 4)        //  النتيجة صح
(3 != 2)       //  النتيجة صح
(6 > = 6)      //  النتيجة صح
(5 =< 5)       //  النتيجة خطأ
```

بالطبع بدلا من استعمال قيمة رقمية ثابتة واحدة فانك بامكانك استعمال اي تعبير مقبول يتضمن متغيرات، كمثال نفرض ان

(a = 2 ،b = 3 ،and c = 6)

ولنلاحظ العلاقات التالية

(a = = 5) // النتيجة خطأ لان (a) لاتساوي (5)

(a*b >= c) // النتيجة صح حيث ان (2*3 >= 6)

(b+4 > a*c) // النتيجة خطأ حيث ان (3+4 > 2*6) خطأ

((b=2) = = a) // النتيجة صحيحة

عند كتابة تعبير معقد يحتوي على عدد من العمليات ربما يحدث لنا بعض الغموض عن كيفية أجراء العمليات الرياضية بمعنى أي من المعاملات يحسب أولا وأيهما لاحقا مثال:

a = 5 + 7 % 2

ربما يكون هناك غموض فهل هذا التعبير يعني التعبير اللاحق الاول ام التعبير اللاحق الثاني

a = 5 + (7 % 2) // مع نتيجة قدرها (6) او

a = (5 + 7) % 2 // مع نتيجة قدرها صفر

النتيجة الصحيحة هي التعبير الاول مع نتيجة قدرها (6)، وذلك لأعتمادنا على ترتيب لأسبقيات حساب العوامل (جدول 1.10 يبين الاسبقيات) وهي ليست للعوامل الحسابية فقط وأنما لكل العوامل التي تظهر في C++.

1.14 التعبير Expression

أي ترتيب من المتغيرات والعوامل الرياضية والذي في النهاية يمثل عملية حسابية يسمى **تعبير**، والتعبير عبارة عن أشتراك عناصر البيانات مع العوامل الحسابية وهذه العناصر يمكن ان تكون ثوابت او متغيرات، وعند أجراء

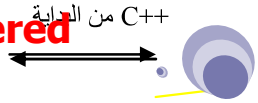
العملية الحسابية فان النتيجة تكون قيمة واحدة.. ومن الممكن ان يكون جزء من التعبير تعبير أيضا.. مثل

(a+20) * b/3

هذا كله يسمى تعبير واجزاءه مثل (a+ 20) و (b/3) كل منها يسمى تعبير أيضا.

وتستخدم مع التعبير عادة عبارة الأسناد (assignment statement) وهي علامة او عبارة تستخدم لأسناد قيمة الى متغير وتستخدم علامة المساواة (=) لتحقيق هذا الغرض.. وبالتأكيد فان العملية ستتم باسناد القيمة المستحصلة من الطرف الأيمن من المساواة الى المتغير الموجود في الطرف الأيسر من المساواة.

بالأمكان كتابة تعبير معين يحتوي على متغيرات من أنواع بيانات مختلفة، مثلا تعبير يحتوي على متغيرات من نوع بيانات صحيحة وبيانات من نوع بيانات حقيقية.. في هذه الحالة فان عملية تحويل أليه داخل الحاسوب ستتم دون تدخل المستخدم حيث سيتم تحويل المتغيرات ذات النوع الاقل اسبقية الى النوع الاكثر اسبقية، الجدول (1.10) يبين أسبقيات العوامل:



جدول (1.10): يبين اسبقيات العوامل

قواعـــــــد الأسبـــقيات		
The Unary Operators العوامل الاحادية	!، --، ++، -، +	الاسبقية العليا (تنفذ اولا)
The Binary Arithmetic Operations العوامل الرياضية الثنائية	*، /، %	
The Binary Arithmetic operations العوامل الرياضية الثنائية	+، -	
The Boolean operations العوامل المنطقية	<، >، <=، >=	
The Boolean operations العوامل المنطقية	==، !=	
The Boolean Operations العوامل المنطقية	&&	
The Boolean Operations العوامل المنطقية	||	الاسبقية الدنيا (تنفذ اخيرا)

1.15 توليد الأرقام العشوائي Random Numbers Generation

تحتاج بعض التطبيقات الى استخدام أرقام عشوائية، وهذا ممكن في لغة البرمجة C++ وذلك من خلال استخدام الأمر (Random) الذي يعمل على توليد رقم بشكل عشوائي، وهو يعمل وفقا لما يأتي:

* يستخدم الأمر (random) لتوليد أرقام عشوائية من نوع الأعداد الصحيحة تتراوح قيمتها بين الصفر والواحد. والامر (random) هو ماكرو معرف في (stdlib).

randomize : وهي تستخدم لتوليد أساس للأرقام العشوائية التي ستعتمد على الوقت

```
randomize ;
x  = random ;
```

هنا المتغير (x) تكون قيمتة (0 <= x < 1) وفي كل مرة يتم تنفيذ هذا الأمر سنحصل على قيمة جديدة ضمن نفس المدى.

* **الطريقة الثانية:** هي باستعمال الأمر (Randomize), ثم الأمر (Random) على أن يحتوي الأمر (Random) على المدى المطلوب لأيجاد الرقم العشوائي ضمنه (أي أنه سيولد أعداد صحيحة موجبة عشوائيا تتراوح قيمتها بين الصفر والعدد المحدد بين القوسين بعد (Random) ناقص واحد والذي يمثل الحد الأعلى)،

مثال:

```
Randomize ;
x = random (100) ;
```

هنا تكون قيمة المتغير (x) (0<= x < 100) وفي كل مرة يعاد تنفيذ هذا الأمر ستحصل على قيمة جديدة. أن المدى المحدد يمكن تغيره حسب طبيعة التطبيق المراد تنفيذة.

* الطريقة الثالثة: لاستخدام الأمر (Random) هي بدون استخدام الأمر (Randomize) وبدلا منه استخدم المتغير (Randseed) قبل الأمر (Random) على أن يتم أسناد قيمة للمتغير (Randseed). من المفروض ان يتم تغيير قيمة (Randseed) عند كل تنفيذ لكي نحصل على عشوائية.

مثال

```
randseed = 1200 ;
x = random ;
OR
randseed  = 3425 ;
x = random (1000) ;
```

في الحالة الأولى فأن المتغير (randseed) أسند له قيمة وهي (1200) ووفقا لها سيولد أرقام عشوائية حقيقية قيمتها أقل من واحد ولو أعدنا التنفيذ مع أسناد قيمة

مختلفة للمتغير (randseed) فأن أرقام عشوائية مختلفة ستولد (حاول تنفيذ الطريقتين ولاحظ الفرق).

أما المثال الثاني فأنه سيولد أرقام عشوائية صحيحة أكبر من الصفر وأصغر من (1000).

1.16 التعليقات Comments

تعد التعليقات من الأمور المهمة في البرنامج، واغلب المبرمجين لايستعملونها ب[illegible]ـة الذكاء التي يتمتع بها المبرمج.. فضلا عن أنك تحتاج أحيانا الى شرح وتوضيح أكثر لبيان الفكرة او الغاية من كتابة عبارة او أيعاز معين او واجب هذه العبارة ضمن البرنامج. كذلك، فان المبرمج ربما لايتذكر بعد مضي شهر او أكثر التفاصيل الكافية وراء كتابة عبارة او أيعاز معين ضمن البرنامج.. لذلك تستخدم التعليقات التي تكتب على البرنامج وفقا لقاعدة كتابتها التي سناتي عليها لتشرح لمن يقرأ البرنامج ماذا نحن عاملون. ولما كانت التعليقات تكتب أمام عبارات البرنامج لذلك يفضل ان تعطي الصورة العامة وليس التفاصيل الدقيقة جدا والتي تكفي لتوضيح الفكرة. وبشكل عام فان التعليقات لاتعد جزء من البرنامج وسيهملها المترجم عند ترجمة البرنامج

التعليقات نوعان.. الأول يبدأ بخطين متوازين (//) وهنا المترجم سيعتبر مابعد الخطين تعليق ليس له علاقة بالبرنامج ويبدأ التعليق من الخطين المتوازيين وينتهي بنهاية السطر.مثال

```
int  x ;  // تعليق قصير
```

أما النوع الثاني فهي التعليقات التي من الممكن أن تكون على عدة أسطر فيتم تحديد نص التعليق بواسطة (/* و */) وهي مفيدة مع التعليقات الطويلة، اذ يستعمل الرمز (*/) لبداية التعليق والرمز (/*) للدلالة على نهاية التعليق.مثال

```
int  x ;      /*       هذا هو تعليق على عبارات البرنامج
                وهو تعليق طويل يراد منه توضيح أسباب أستعمال نوع البيانات
```

لذلك أضطررنا الى استعمال عدة سطور من التعليق... الخ */

يجب أن تلاحظ مايلي عند كتابة تعليق:

1. عدم ترك فراغ بين الشرطة (/) والنجمه (*) من كل جهات جملة التعليق.
2. يقوم مترجم C++ بأهمال النصوص المستعملة في جملة التعليق (أي لا ينفذها).
3. من الممكن وضع جملة التعليق في أي مكان من البرنامج، ما عدا وسط الأسم التعريفي (identifier) أو الكلمة المحجوزة (keyword). فمثلا الأمثلة أدناه غير مقبولة:


```
* whi /* name = ' saad ' */ le     c = ' Ahmed '

* Sum =  /* xxx  */  0 ;
```

4. لا ينصح بوضع تعليق داخل تعليق أخر، لأن ذلك قد يتسبب بحدوث أخطاء، مثال

```
/* Program /* written by Saad */ card game */
```

هنا المترجم سيعتبر الجمله التعليقية تنتهي عند (Saad)، والباقي سيعتبره كود.

5. يهمل المترجم السطر أو بقية السطر الذي يبدأ بخطين مائلين (//).

1.17 عامل الزيادة Increment Operator

تستعمل في بعض التطبيقات عدادات لأغراض محددة وهي عادة تبدأ بالرقم (0) أو أي رقم أخر وتزداد بمقدار واحد (او اكثر) في كل مرة وتكتب عادة كما ياتي:

```
count = count + 1 ;
```

ونظرا لأن هذا العامل واسع الاستعمال لذا فأن لغة C++ وفرت عامل مفرد (للأختصار) لهذا الغرض وهو (++) لاغراض الزيادة بمقدار واحد أو (- -)

لأغراض النقصان بمقدار واحد حيث يستخدم هذا العامل بطريقتين أما أن يسبق المتغير مثل (++m) او أن يلي المتغير مثل (m++) وهما ليسا متشابهين فكل منهما له معنى خاص فعندما يسبق المتغير عامل الزيادة فان المتغير تزاد قيمتة بمقدار واحد ثم يستخدم أما أذا جاء عامل الزيادة بعد المتغير فان المتغير يستخدم حسب قيمتة الحالية وبعدها يزاد بمقدار واحد. أما العامل (- -) فتعمل بالطريقة نفسها التي يستخدم فيها عامل الزيادة أي قبل وبعد المتغير مع الأختلاف ان استخدامها يقلل قيمة المتغير بمقدار واحد، مثال

أذا فرضنا ان المتغير (b = 7) والمتغير (a = 2) فان قيمة (C) في التعبير التالي:

```
C = a * ++b ;
```

تكون قيمتها (16)، حيث ان المترجم سيقوم بزيادة قيمة (b) لتكون (8) ثم يعوض عنها في التعبير ويحسب نتيجة التعبير، أما قيمتها في التعبير التالي:

```
C = a* b++ ;
```

فتكون (14)، حيث ان المترجم سيستخدم القيمة الحقيقية للمتغير (b) ثم يقوم بحساب نتيجة التعبير وبعد ذلك تتم زيادة قيمة المتغير (b) لتكون (8)


```
C = a* --b ;
```

هنا قيمة (C) تكون (12) حيث سيقوم المترجم بأنقاص قيمة (b) بواحد لتكون قيمتة (6) ثم تعوض قيمتة في التعبير لايجاد قيمة (C)

أما قيمتها بالتعبير التالي:

```
C = a* b-- ;
```

تكون (14) حيث ستستخدم قيمة (b) الحقيقية (7) لأيجاد قيمة (C) بعدها تقلل قيمة (b) لتكون قيتها (6).

1.18 بعض المحددات الخاصة

هذه المحددات ستكون اكثر وضوحا في الفصول الاخرى وسيتم شرحها بالتفصيل, هنا فقط يتم الاشارة لها.

1.18.1 المحدد (متطايرة) volatile

بعكس المحدد (const) الذي يؤدي الى جعل قيمة المتغير ثابتة فأن المحدد (volatile) يؤدي الى جعل قيمة المتغير تتغير كلما تطلب الأمر ذلك بدون سيطرة المترجم أو توجيه تحذير الى المبرمج، وهذا المحدد مفيد في العمليات المتعددة التي تأخذ معلوماتها من الذاكرة. وبعبارة أخرى يحتاج المبرمج الى استخدام (volatile) عندما يتعامل البرنامج مع البرامج الفرعية ذات العلاقة المباشرة بالمكونات المادية للحاسوب .. مثال

```
volatile print_register ;
volatile port ;
volatile A[10] ;
```


1.18.2 المحدد (المسجل) register

يستعمل هذا المحدد لأعلام المترجم أن يحفظ قيم المتغيرات في مسجلات (registers) وحدة المعالجة المركزية (CPU) مباشرة، وليس في الذاكرة حيث تخزن عادة قيم المتغيرات. وهذا يعني أن العمليات التي تجري على هذا النوع من المتغيرات تكون أسرع من العمليات التي تجري على المتغيرات المخزنة في الذاكرة. ومما تجدر الاشارة له أن المحدد (register) يتعامل مع نوعين من المتغيرات هما الأعداد الصحيحة والرموز (characters) كما أنه يستعمل في حالات المتغير الموضعي أو متغير الدالة اللذان يعتبران من نوع المتغيرات الذاتية (Auto)، ولذا فأن (register) لا تستعمل للمتغير العام، وتستخدم هذه المتغيرات في برامج التكرار (Loops)، مثال

```
register  int  i ;
for (i = 0 ; i < last ; ++ i)
```

أن عدد المتغيرات من هذا النوع يعتمد على نوع المعالج المستعمل وعلى تطبيقات C++ فمثلا في الأنظمه ذات (bit8) يستخدم متغير واحد وفي نظام (bits16) يستخدم متغيران.

وكمبرمج بلغة C++ يمكنك استخدام أي عدد من هذه المتغيرات لأن المترجم سيسجل الفائض من هذه المتغيرات كمتغيرات عادية وليس متغيرات (register) بشكل تلقائي.

وينصح باستخدام متغيرات (register) في التطبيقات التي تستخدم حلقات التكرار (Loops) عادة.

1.19 الأدوات الدقيقة Bitwise Operators

تتميز لغة C++ عن سائر لغات البرمجة الراقية باستخدامها أدوات دقيقة تعمل على مستوى وحدة التخزين الاولية (bit)، وسميت هذه الأدوات بالدقيقة لأنها تتعامل مع البت بشكل مباشر، فحصا، ضبطا، وازاحة. وتستعمل هذه الأدوات مع البيانات الصحيحة (int) والرمزية (char) فقط ولا تستعمل مع غيرها من البيانات والجدول (1.11) يوضح الأدوات الدقيقة وعملها:

جدول (1.11): الأدوات الدقيقة واستخداماتها

العوامل الدقيقة	العمليات الرياضية المكافئة	استخدامها (عملها)
&	AND	تقوم بعملية (و) بين البتات Bitwise AND
|	OR	تقوم بعملية (أو) بين البتات Bitwise Inclusive OR
^	XOR	تقوم بعملية (xor) بين البتات

		Bitwise Exclusive Or
~	NOT	عكس قيمة البت bit inversion
<<	SHL	أزاحة البتات لليسار Shift Left
>>	SHR	أزاحة البتات لليمين Shift Right

1. النفي يحول كل صفر الى واحد وكل واحد الى صفر.
2. أدوات الازاحة أستعمالها يؤدي الى أزاحة قيمة المتغير الصحيح (الممثل بالنظام الثنائي) يمينا أو شمالا عدد من الخانات (البتات) وحسب الطلب، وتملأ الخانات المفرغة أصفارا أو واحدات حسب أشارة العدد (فالعدد الموجب عند أزاحتة تملأ فراغاتة أصفار، بينما العدد السالب تملأ فراغاتة واحدات عند أزاحتة)، مثال..

أذا أردنا أزاحة المتغير (X) الى اليمين خانتين فيكتب كمايأتي:

```
X >> 2;
```

جدول (1.12): جدول يبين أسبقيات العمليات الدقيقة

	أسبقيات الأدوات الدقيقة
1	~
2	<< >>
3	&
4	^
5	|

ملاحظة://

للتأكد من سلامة نتائج عمليات الأزاحة فمن الممكن استخدام القاعدة التالية:

كل ازاحة الى اليمين بمقدار بت واحد ينتج عنها قسمة القيمة المزاحة على (2) (أي لكل بت ازاحة نقسم العدد على 2)

كل ازاحة الى اليسار بمقدار بت واحد ينتج عنها ضرب القيمة المزاحة بالرقم (2) (أي لكل بت أزاحة نضرب العدد في 2)

1.20 تحويل نوع البيانات Type Conversions

عند استخدام أكثر من نوع من البيانات في تعبير معين، فأنه من الممكن أن نحول نوع متغير معين ضمن التعبير الى نوع أخر، وذلك بأجراء التحويل على المتغير الموجود الى يمين المساواة، ليصبح نوعة حسب نوع المتغير في جانبها الأيسر. مثال


```
int   a,b ;
char  name;
float   x ;
name = a ;    b = x ;   x = name ;     x = a ;
```

نلاحظ أن هذه الأشكال من التحولات بين أنواع البيانات غير موجودة في العديد من اللغات الأخرى، وذلك لأن C++ صممت أصلا لتكون لغة وسيطة بين اللغات العليا ولغة التجميع (Assembly).

*** تغيير نوع المتغير**

ان تغيير نوع المتغير هو اسم معقد لمفهوم بسيط. فعند تغيير نوع المتغير من نوع الى اخر، فان كل الذي تعملة هو اخبار الحاسوب باستعمال نوع مختلف لخزن المتغير. اذن لماذا نحتاج الى عمل ذلك؟ دعنا نقول بانك اعلنت عن متغير من نوع short، في اغلب الاحيان ان هذا يعني ان اكبر قيمة موجبة من الممكن ان تخزنها ستكون 32,767، ولكن في مكان ما في البرنامج، ادركت انك ستقوم بعملية حساب ستؤدي الى زيادة القيمة فوق هذه القيمة العظمى. فمثلا لحساب طول c (وتر المثلث القائم الزاوية)، فانك تحتاج الى حساب الجذر التربيعي لمربع الظلعين الاخرين $a^2 + b^2$، ولكن ماذا يحدث لو كانت قيم كل من a، b كبيرة جدا، عليه سيكون التربيع

كبيرا جدا، فاذا اصبحت القيمة اكبر من 32,767 فان قيمتك ستكون ليس كما تتوقع (اذا استخدمت النوع short لخزن الناتج) ستكون قيمة الناتج غير صحيحة.

عليه فان الحل هو تغيير النوع، فبامكانك ان تغير النوع للارقام الى نوع بيانات اكبر، مثل (long ,int) لاغراض الحساب.. وبعدها من الممكن اعادتها ثانية الى short عند الانتهاء، اذ ان القيمة النهائية للمتغير c ربما ستكون صغيرة بما يكفي ان تخزنها بالنوع short. في الحقيقة هذا مثال بسيط ويمكن حل المشكلة بان تخزن المتغير من البداية بالنوع int ، مثال اكثر فائدة هو اذا كان لديك رقم والذي يمثل معدلا مثلاً, فانك ربما ترغب ان تمثل الرقم بالنوع float لتكون القيمة اكثر دقة عند حسابها. ويمكن تغيير النوع ليكون int.

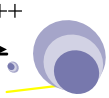
كيف يتم تغيير النوع:

عملية تغيير النوع في C++ عملية سهلة. لنقل لديك المتغير (average) من النوع float لخزن رقم مثل الرقم (0.314188526) وترغب ان يكون لديك خزن من نوع int لخزن جزء العدد الصحيح من الرقم اعلاه. ادناه كيف تعمل ذلك:

```
int  CastAverage = (int) average ;
```

لاحظ هنا اننا اعلنا عن متغير (CastAverage) من النوع int لنضع فيه القيمة بعد تغيير النوع وهنا اننا غيرنا النوع وذلك بوضع النوع الذي نرغب ان نغير نوع المتغير اليه نضعة بين قوسين قبل اسم المتغير.

> ***ملاحظة ://***
> عند التحويل من البيانات الطويلة الى أخرى أقصر فأن عدد من الخانات (البتات) ستفقد.



ملاحظة://

أن التحويل بين نوع وأخر من أنواع البيانات، يتم بصورة تلقائية (أوتوماتيكية) داخل التعبير الواحد، اذ يقوم مترجم C++ بتحويل جميع المتغيرات الى النوع ذي الطول الأكبر، فيتحول الصحيح الى حقيقي ويتحول الحقيقي الى مضاعف وهكذا.

1.20.1 عامل تحويل النوع الخارجي Explicit Type Casting Operator

عامل تحويل النوع يسمح لك بتحويل نوع معين الى نوع أخر. هناك عدة طرق لعمل ذلك في C++، ابسط طريقة والتي ورثت من لغة C هو بأن تسبق التعبير المراد تحويلها بالنوع الجديد محاط بقوسين (()):


```
int i;
float  f = 3.14;
i = (int) f;
```

المثال السابق يحول العدد الحقيقي (3.14) الى عدد صحيح (3)، طبعا الباقي (الكسر) سيفقد. هنا معامل التحويل هو (int). طريقة أخرى لعمل نفس الشيء في C++ وذلك باستخدام النوع الذي سبق التعبير المراد تحويلة بالنوع الجديد وتحديد التعبير بأقواس


```
i = int (f);
```

كلا الطريقتين مقبول في C++

1.21 حجم البيانات (sizeof)

هذا العامل يقبل وسيط واحد والذي ممكن ان يكون نوعا او المتغير نفسة ويعيد قيمة تمثل حجم النوع او الكيان **بالبايت**:

```
a = sizeof (char);
```

في المثال اعلاه فان قيمة (a) ستكون (1) وذلك لان النوع (char) هو نوع بطول بايت واحد. القيمة المعادة بواسطة (sizeof) ثابتة، لذلك دائما تحسب قبل

تنفيذ البرنامج.

1.22 الأخطاء التي ترافق البرامج Errors

هناك أربع أنواع من الأخطاء التي تحدث في الحاسوب عند تنفيذ برنامج وهي:

1. أخطاء المترجم Compiler errors

تحدث هذه الأخطاء أثناء محاولة المترجم ترجمة البرنامج وهي ناتجة عن خطأ قواعدي في كتابة البرنامج، مثل عدم وضع فارزة منقوطة في نهاية عبارة كاملة.

2. أخطاء الربط Linker errors

ان أغلب الأخطاء من هذا النوع تحدث عندما لايتمكن الرابط (Linker) من أيجاد الدوال أو عناصر البرنامج الأخرى والتي يشار اليها في البرنامج.

3. أخطاء وقت التنفيذ Run-time errors

في بعض الأحيان لايتم الكشف عن الخطأ الا أثناء تنفيذ البرنامج، مثال القسمة على صفر.

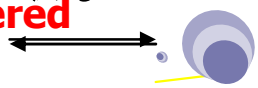
4. أخطاء مرئية Conceptual errors

هذه أخطاء يقع بها المبرمج نتيجة لخطأ في الطباعة أو السهو وهي صحيحة للمترجم ولكنها تعطي نتائج خاطئة.

1.23 موجهات التضمين وفضاء الاسماء Include Directives and Namespaces

جميع برامجك تبدأ بالسطرين التاليين

```
#include<iostream>
using namespace std;
```



ولمناقشة وظيفة هذين السطرين سنبدأ بالسطر الاول والذي هو يتضمن جزئين، الجزء الاول هو (#include) وهذا يعني ان المطلوب هو تضمين برنامجك بالملف الموضح اسمة لاحقا، وهذه من الممكن ان تكون اكثر من ملف واحد (اي اكثر من #include) كل واحد منها له وظيفة اضافة ملف معين تحتاج له في تنفيذ برنامجك وهذه الملفات موجودة ضمن المكتبة القياسية للغة، اما الجزء الثاني من السطر الاول والذي سنطلق عليه تسمية الموجة او الملفات الرأسية (السطر الاول) فانه يحتوي على اسم الملف المطلوب اضافتة الى البرنامج ويكون محددا بين العلامتين (< >) عادة يكون اسم الملف الموضح في هذا السطر هو باسم (iostream) وهذا الملف هو المسؤول عن توفير وتفعيل اوامر الادخال والاخراج ونظرا الى انك في كل برنامج تكتبة لابد من الاحتياج الى عملية ادخال او اخراج او كليهما لذلك فلا بد من ان تكون مكتبة iostream متوفرة، هذه المكتبة تتضمن تعريف cin، cout / (وهي اوامر الادخال والاخراج وسيتم شرحها في الفصل القادم)، فضلا عن امور اخرى. هناك ملفات اخرى كثيرة ربما تحتاج لها في تنفيذ برنامجك ولكل منها واجب محدد (لمزيد من المعلومات يمكنك الاطلاع على هذه الملفات في الملاحق).

السطر الثاني يتضمن التعبير: **using namespace std;**

C++ تقسم الاسماء الى فضاءات اسماء، وفضاء الاسماء هو تجمع للاسماء، مثل الاسماء (cout ،cin). العبارة التي تحدد فضاء الاسماء بالطريقة الموضحة ادناة تدعى الموجة using.

```
using namespace std;
```

هذا الموجة الخاص (using) يفيد ان برنامجك يستخدم او يفرض استخدام فضاء الاسماء القياسية (std)، هذا يعني بان الاسماء التي تستخدمها سيكون لها المعاني المحددة لها في فضاء الاسماء القياسية. في هذه الحالة، الشيء المهم هو عندما تكون الاسماء مثل cout ،cin معرفة في iostream، تعريفها يفيد انتماءهم

الى فضاء الاسماء القياسية. لذا ولاجل استخدام الاسماء مثل cin، cout فانك تحتاج الى اخبار المترجم بانك تستخدم فضاء الاسماء القياسية.

هذا كل ماتحتاج الى معرفة الان حول فضاء الاسماء، ولكن توضيح مختصر سوف يحل اللغز الذي يحيط استخدام فضاء الاسماء. السبب ان C++ له فضاء اسماء بشكل مطلق وذلك بسبب وجود اشياء كثيرة يجب تسميتها. كنتيجة، احيانا يستلم عنصران او اكثر نفس الاسم، بمعنى اسم مفرد وممكن ان يحصل على تعريفين مختلفين. ولازالة هذا الغموض، C++ يقسم العناصر الى مجاميع، لذا لايوجد عنصران في نفس التجمع (نفس فضاء الاسماء) لهما نفس الاسم.

لاحظ ان فضاء الاسماء هو ليس تجميع بسيط للاسماء. هو جسم لشفرة C++ والتي تحدد المعنى لبعض الاسماء، مثل بعض التعريفات و/او الاعلانات. وظيفة فضاء الاسماء هو تقسيم جميع مواصفات اسماء C++ الى تجمعات (تدعى فضاء الاسماء) اذ ان كل اسم في فضاء الاسماء يملك فقط مواصفة واحدة (تعريف واحد) في فضاء الاسماء.

فضاء الاسماء يقسم الاسماء ولكن ياخذ الكثير من شفرة C++ مع الاسماء.

ماذا لو اردت ان تستخدم عنصرين في فضائي اسماء مختلفين، اذ ان كلا العنصرين له نفس الاسم؟ من الممكن ان تقوم بذلك وهي ليست معقدة، وهذا سنشير اليه لاحقا في هذا الكتاب.

ملاحظة://

بعض نسخ C++ تستخدم التالي، والذي هو نسخة قديمة او شكل قديم للموجة include (دون استخدام فضاء الاسماء):

```
#include<iostream.h>
```

فاذا كانت برامجك لاتترجم او لاتنفذ مع العبارات التالية

```
#include<iostream>
using namespace std;
```

عليك ان تحاول استخدام السطر التالي بدلا من السطرين السابقين

#include<iostream.h>

فاذا طلب برنامجك iostream.h بدلا من iostream، عليه فان ذلك يعني انك تملك نسخة قديمة من مترجم C++ وعليك ان تحصل على نسخة حديثة.

جدول (1.13) بعض الدوال المهمة

التسلسل	الدالة	وظيفتها
1	abort( )	دالة أيقاف البرنامج (تنهي تنفيذ البرنامج فورا)
2	abs()	دالة القيمة المطلقة الصحيحة
3	ceil()	دالة أيجاد أكبر عدد صحيح للقيمة (x) مثال Ceil (8.7) هـــو 9
4	clrscr()	دالة تنظيف الشاشه
5	exit()	دالة الخروج من البرنامج
6	floor()	دالة أيجاد أصغر عدد صحيح للقيمة (x) (تستعمل لأيجاد أصغر عدد صحيح للقيمة الحسابية حسب التعريف الرياضي المعروف [x] (أذا كانت (x) سالبه يحذف كسرها وتنقص واحد)
7	log()	دالة اللوغاريتم الطبيعي (تحسب اللوغاريتم الطبيعي (lin (x)) ويجب أن تكون قيمة (x) أكبر من الصفر.
8	log10()	دالة اللوغاريتم العشري (تحسب اللوغاريتم للأساس 10 ($\log_{10}(x)$) ويجب أن تكون (x) أكبر من الصفر
9	pow()	تحسب هذه الداله قيمة المقدار (x^y) وكمايلي Z = pow (x ،y) ;
10	sqrt()	دالة أيجاد الجذر التربيعي لعدد موجب (x)، مثال Y = sqrt (x);

اسئلة للحل:

1. اي من التعابير التالية هو متغير مقبول:

int n = - 10 ;

int x = 2.9 ;

int 2k ;

float y = y * 2 ;

char c = 123 ;

char h = “c” + 23 ;

int !b ;

float c ;

2. اي من العبارات ادناه يمثل معرف مقبول:

Seven_11

_unique

Gross-income

Gross$income

2by2

Averag_ weight_of_a_large_pizaa

Object.oriented

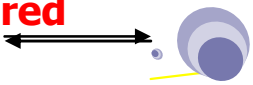
Default

@yahoo

chapter 3

INPUT / OUTPUT INSTRUCTIONS IN

C++



أوامر الأدخال والأخراج
INPUT / OUTPUT INSTRUCTIONS

2.1 المقدمة

جميع اللغات الطبيعية التي يتعامل بها الأنسان كوسيلة للتخاطب والتواصل لها قواعد وضوابط تحدد ألية استخدامها، ولما كانت لغات البرمجة تصنف على أنها من اللغات العليا (أي اللغات القريبة من لغات البشر) فكان لابد وأن تكون لها قواعد تحدد ألية استخدامها لتكون واضحة للمتعامل معها وكذلك للمترجم داخل الحاسوب. علية فأن هذا الفصل والفصول اللاحقة ستوضح هذه القواعد وسنبدأ خلال هذا الفصل بمعرفة كيفية تلقيم الحاسوب بالمعلومات وطرق الحصول على النتائج بعد أنجاز عمليات الحساب.

2.2 هيكلية البرنامج Program Construction

يتكون برنامج لغة C++ من (الرأس والجسم) (head and block) والرأس هو السطر الأول في البرنامج ويبدأ بكلمة (#include) ويتبع باسم الملف الرأسي (header file) والذي يكون محدد بين علامتي الاكبر والاصغر (< >) وكما يأتي:


```
#include<iostream>
```

اما جسم البرنامج فيبدأ بالدالة (main()) ثم يتبع بالايعازات والأوامر التي تمثل الخطوات الواجب أتباعها أو تنفيذها من قبل الحاسوب للحصول على النتائج المطلوبة من البرنامج، وتكون هذه الايعازات محددة بأشارة البداية والنهاية حيث تستخدم الأقواس المتوسطة لهذا الغرض ({ }).

```
#include<iostream>

main ( )

{
```

Set of instructions;

}

2.3 المخرجات والمدخلات Input / Output

كل برنامج يجب أن تكون له مخرجات تبين النتائج التي تم الحصول عليها من البرنامج، هذه النتائج سيتم عرضها على شاشة الحاسوب باستخدام عبارة الأخراج (<< cout) أن الأمر (<< cout)من الممكن ان يترجم على انه أكتب

ماموجود بعد العلامة (<<) على السطر الذي يؤشر عليه المسيطر (controller) في شاشة التنفيذ.

عبارة الأخراج لها أثنان من صفات C++ الجديدة وهي (cout) و (<<)، حيث أن المعرف (cout) يلفظ (C out) وهو كيان معرف مسبقا يمثل تدفق المخرجات القياسية في++C، هنا تدفق المخرجات القياسية يمثل طباعتها على الشاشة، ومن الممكن أعادة توجية المخرجات الى أجهزة أخرى.

أما العامل (<<) ويدعى (insertion OR put to operator) (عامل الحشر أو الوضع) وواجبة حشر أو أرسال محتويات المتغير الذي على جانبها الأيمن الى الكيان الذي موجود على جانبها الأيسر.

ملاحظة://

(bit_wise) يستخدم أيضا العامل (<<) كعامل تزحيف الى اليسار(يعمل) على مستوى البتات, كما سبق وان اشرنا في الفصل الاول.

أن ما يوضع بعد العلامة (<<) سيأخذ حالة من أثنتين:

2.3.1 الحالة الأولى

ان يكون ما بعدها محدد بعلامات اقتباس مزدوجة (double quotation mark) (" ") وبهذه الحالة فان ما موجود بين علامتي الأقتباس سيتم طباعتة على الشاشة كما هو دون أي تغيير.

برنامج لطباعة عبارة معينة على الشاشة

```
// Example 2.1
#include <iostream>
using namespace std;
```

```
main()
{
  cout <<" Hello World. Prepare to learn C++ !!" ;
}
```


لاحظ مايلي://

اولا/ان مخرجات هذا البرنامج هي العبارة التي تلي العامل (<<)، وستظهر على الشاشة كما يلي:

مخرجات البرنامج 2.1:

```
Hello World. Prepare to learn C++ !!
```

ثانيا/ عند تنفيذ هذا البرنامج سوف لا يمكن ملاحظة المخرجات والسبب هو أن الحاسوب سريع جدا بحيث يعرض ويخفي شاشة التنفيذ دون ان تلاحظ ذلك، ولغرض رؤية المخرجات فيمكن بعد ان يتم التنفيذ ضغط الزرين (Alt+ F5) معا وعندها ستظهر شاشة التنفيذ (السوداء).. ويمكن الخروج من شاشة التنفيذ بضغط الزر (Enter)

ملاحظة://

لغرض ايقاف شاشة التنفيذ بعد انتهاء التنفيذ لرؤية النتائج، استخدم الامر التالي في نهاية البرنامج:

```
system ( "pause") ;
```

مع ملاحظة ان هذا الامر يعمل مع الموجهة #include<stdlib>

وعند استخدامة سوف لاتختفي شاشة التنفيذ بعد انتهاء التنفيذ مع وجود ملاحظة تخبر المستخدم بالضغط على اي زر لغرض الأستمرار.

2.3.2 الحالة الثانية

أما أذا كان ما موجود بعد العلامة (<<) ليس محدد بين علامتي أقتباس فعند ذلك سيعامل ما موجود بعدها على أنه معرف والمعرفات هنا تكون على واحدة من الحالات ادناه:

* أما أن تكون مقادير ثابتة (قيم حسابية) مثل القيم (4567، 123-، 78.456...الخ) فهي تطبع مباشرة على الشاشة دون تغيير، مثلا

```
cout << 3456 ;
```

هنا سيتم طباعة (3456) على الشاشة.

* أو تكون على شكل تعبير حسابي (expression) (اي مقادير تفصل بينها العوامل الرياضية او المنطقية مثل (+، - ، * .. الخ) وبهذه الحالة فسيتم استخراج قيمة العملية الحسابية او المنطقية وطباعتها على الشاشة، مثال

```
cout << 34 + 56 ;
```


في هذه الحالة سيتم طباعة (90) على الشاشة.

* أو أن تكون على شكل رموز، وتعد انذاك متغيرات (والمتغيرات لها اسماء) تؤشر الى قيم في الذاكرة (**يجب أن تكون لها قيمة**) (كما سبق ان وضحنا بالفصل الاول فان المتغيرات تشير الى مواقع في الذاكرة وهذه المواقع تحتوي على قيم)، لذا فان الحاسوب سيطبع قيمة المعرف (المتغير) على شاشة التنفيذ (أي تطبع القيمة الموجودة او المخزونة في موقع الذاكرة الذي يشير له المتغير).

هنا عليك أن تلاحظ أن استخدام أي معرف (متغير) داخل البرنامج يحتاج الى شرطين:

الأول/ أن يتم الأعلان عن المعرف قبل أن يتم استخدامة لأول مرة في البرنامج ويحدد نوعة وفقا للأنواع التي سبق أن نوهنا عنها في الفصل الاول، فأذا كانت قيمة المتغير غير ثابتة ويمكن ان تتغير قيمتة (تتغير قيمتة أثناء تنفيذ البرنامج) فيعلن عنه ويحدد نوعة (ويتم ذلك بكتابة اسم المتغير مسبوقا بنوعة)، فمثلا أذا كان المطلوب استخدام المتغير (x) وهو من نوع الأعداد الصحيحة، فيكون بكتابة النوع أولا ثم يتبع ذلك كتابة أسم المتغير(على أن يكون هناك فراغ بين النوع واسم المتغير) وتنتهي العبارة دائما بفارزة منقوطة، وكما يأتي:


```
int  x ;
```

هذا المتغير هو من نوع الأعداد الصحيحة (integer) أي أن القيمة التي يحملها دائما ستكون عدد صحيح. ويجب ان تلاحظ ان الاعلان عن المعرف يكون لمرة واحدة في البرنامج.

ثانيا/ يجب أن تكون لهذا المتغير أو الثابت قيمة عند أول استخدام له داخل البرنامج فمثلا أنك عرفت المتغير (x) من نوع الاعداد الصحيحة لكن كم هي قيمة هذا المتغير؟ هو عدد صحيح لكن كم !! فعندما تعطي الأمر (cout << x ;) فكم يجب على المترجم أن يطبع على شاشة التنفيذ ! لذا يجب أن تحدد قيمة المتغير أو الثابت قبل او اثناء أول استخدام.

هذه القيمة التي تحدد وتسند للمتغير تأتي من احدى عمليتين فأما أن تسند القيمة للمتغير اثناء كتابة البرنامج أو تسند القيمة للمتغيراثناء تنفيذ البرنامج... لنناقش الحالتين:

ملاحظة://

سبق وان ذكرنا ان بالامكان أسناد الاعداد الصحيحة للمتغيرات من نوع الاعداد الصحيحة، والقيم الحقيقية للمتغيرات من نوع الاعداد الحقيقية، والحروف للمتغيرات من نوع الحروف وهكذا.. ولكن الحقيقة ان هذا القول ليس دقيقا وذلك لان لغة C++ تحول بين الانواع أليا في بعض الحالات، مثال:


```
int number;
```

number = ‘a’;

cout << number <<endl ;

النـــاتج هنـــا ســـيكون (97) وهـــو الـــرقم الـــذي يســـتخدم داخليـــا فـــي لغـــة C++ (ASCII) لتمثيل الحرف (a)، ولكن من المناسب استخدام الاعداد الصحيحة لمتغير الاعداد الصحيحة والحروف لمتغير الحروف ولا تحول بينهما الا اذا كان هناك سبب معقول.

برنامج لـ[illegible] لطباعة عبارة معينة وعدد يمثل العمر، مع ملاحظة زيادة هذه الارقام وانقاصها.

```
// Example 2.2
#include <iostream>
using namespace std;
 int main()
 {
int myAge = 39; // initialize two integers
int yourAge = 39;
cout << "I am: " << myAge << " years old.\n";
cout << "You are: " << yourAge << " years old\n";
myAge++;   // postfix increment
++yourAge; // prefix increment
cout << "One year passes...\n";
cout << "I am: " << myAge << " years old.\n";
cout << "You are: " << yourAge << " years old\n";
cout << "Another year passes\n";
cout << "I am: " << myAge++ << " years old.\n";
cout << "You are: " << ++yourAge << " years old\n";
```

```
cout << "Let's print it again.\n";
cout << "I am: " << myAge << " years old.\n";
cout << "You are: " << yourAge << " years old\n";
return 0;
}
```

مخرجات البرنامج 2.2:


```
I am 39 years old
You are 39 years old
One year passes
I am 40 years old
You are 40 years old
Another year passes
I am 40 years old
You are 41 years old
Let's print it again
I am 41 years old
You are 41 years old
```


ملاحظة://

لغرض اخراج رسالة خطأ فبالامكان استخدام الايعاز (cerr<<) بدلا من ايعاز الاخراج الاعتيادي (cout<<)، وطبعا عليك ان تكتب ماهي الرسالة التي ترغب ان تظهر عند وجود خطأ، مع ملاحظة ان مايكتب بعد (cerr<<) سيكون محدد بحاصرة مزدوجة. مثال

```
cerr<< " Error, can't divide by zero " ;
```

* اسناد القيم أثناء كتابة البرنامج:

ويتم ذلك من خلال استخدام التعابير (expression)، ويستخدم التعبير مع معادلة (والمعادلة عبارة عن طرفين يفصل بينهما علامة التخصيص (assignment) الطرف الأيمن هو عبارة عن تعبير او قيمة ثابتة بينما الطرف الأيسر يكون متغيرا ومتغير واحد فقط، لذا فان المساواة تستخدم لاسناد قيمة للمتغير)، فمثلا تقول:

```
x  = 5 ;
```


هنا استخدمنا المساواة (=) وبذلك فان قيمة المتغير (x) ستكون مساوية الى العدد الصحيح (5)، أو ممكن أن تكون المعادلة على شكل:

```
x  = 3 * 2 +5 ;
```

هنا قيمة (x) تساوي (11)، وكذلك ممكن أن تحدد قيمة للمتغير بالمساواة ولكن في حقل الأعلان عن الثوابت.

ملاحظة://

دائما عند وجود علامة المساواة (=) فان الضوابط التالية ستطبق:

يجب أن يكون هناك طرفين تفصل بينهما علامة المساواة، وبذلك ممكن أن نطلق عليها تسمية المعادلة.

الطرف الأيسر من المعادلة أي الذي يقع على الجانب الأيسر من المساواة يكون متغيرا ومتغير واحد فقط دائما، ولا يجوز أن يكون قيمة ثابتة (مثلا 6، 456، 34.2..الخ)، ولا يجوز أن يكون رمز معرف ومعلن عنه على أنه ثابت، كذلك لا يجوز أن يحتوي على علاقات رياضية مثل (x + 6).

أما الطرف الأيمن فيمكن أن يكون قيمة رقمية أو عددية واحدة أو علاقة رياضية (تعبير) تحتوي على (قيم عددية تفصل بينها العلامات الرياضية، أو علاقة رياضية تحتوي متغير واحد، متغيرات، أو متغيرات وقيم عددية). مثلا العلاقات التالية مقبولة

```
X  = 89 ;
X  = 34 – 45 +3;
X  = y ;
X  = 3 * y +90 ;
```

من الممكن أن يكون في التعبير الواحد أكثر من مساواة واحدة (سنأتي عليها في موضعها).

عند تنفيذ البرنامج فأن المترجم سيبدأ بالطرف الأيمن من المعادلة دائما ويتم فحص هذا الطرف لما يحتوي فيه متغيرات فسيبحث المترجم في الخطوات السابقة للخطوة التي هو فيها ضمن البرنامج للتأكد من أن المتغير معلن عنه (له نوع) أولا، ثم يجب أن تكون له قيمة قبل هذه الخطوة، وتجلب هذه القيمة لتعوض عن المتغير في المعادلة (ممكن أن تتخيل الطرف الأيمن عندها سيصبح عبارة عن مجموعة من القيم الثابتة بعد ان يتم تعويض قيم المتغيرات داخليا في الحاسوب)، بعدها تجرى العمليات الحسابية وتكون من اليسار ألى اليمين وحسب أسبقيات العمليات الرياضية، فالأسبقية الأعلى تنفذ أولا وأذا تساوت عمليتان بالاسبقية فتنفذ العملية التى في اليسار اولا، من ذلك سينتج لنا قيمة واحدة ثابتة، هذه القيمة ستؤول الى المتغير الذي في الطرف الأيسر (دائما القيمة تنتقل من الطرف الأيمن للمعادلة (التعبير) الى المتغير الذي في الطرف الأيسر اي تخزن في الذاكرة في الموقع الذي يشير له المتغير الذي بالطرف الأيسر).

يجب أن يكون المتغير الذي على يسار المساواة والمتغير أو المتغيرات على يمين المساواة من نفس النوع وأذا ما اختلفت الأنواع فهناك عمليات من الممكن أن تجرى أليا لتحويل الأنواع سنأتي عليها لاحقا.

* أسناد القيم أثناء تنفيذ البرنامج:

وتتم عملية اسناد (ادخال) قيمة للمتغير أثناء تنفيذ البرنامج وذلك باستخدام أمر القراءة (cin >>) وهي تعني (أقرأ القيمة المطبوعة على شاشة التنفيذ واخزنها في موقع الذاكرة الذي يشار اليه بواسطة المتغير الموجود بعد العلامة (>>).

* برنامج لادخال قيمتين لمتغيرين اثناء تنفيذ البرنامج وايجاد مجموعهما.


```
// Example 2.3
#include <iostream>
using namespace std;

main()   // no semicolon
{
  int num1 ،num2 ،sum ;
  cout<< "input number 1 :";
  cin>> num1;
  cout<< "input number 2 :";
  cin>> num2;
  sum = num1 + num2; //addition
  cout<<sum ;
  return 0;
  }
```


مخرجات البرنامج 2.3 :

```
input number1:   20    // Press enter
```

```
input number 2:    15      // Press enter
 35
```

ملاحظة://

في كل تطبيق يجب أن يتأكد المبرمج من أن الكيان أو المتغير الموجود في البرنامج له قيمة قبل أن يتم استخدامة لأول مرة في البرنامج، في خلاف ذلك فأن المترجم سيستخدم متغيرا ليس له قيمة محددة من المبرمج او المستخدم، لذلك فان المترجم سيستخدم القيمة الموجودة في موقع الذاكرة الذي يشير عليه المتغير ودائما تكون قيم من برامج سابقة ليس لها علاقة ببرنامجك وبالتالي فستحصل على نتائج خاطئة او ربما تكون قيمتة صفرا اذا لم يتم استخدامة سابقا (اي خالي من القيم).

شرح البرنامج 2.3://

أولا://: تم استخدام المتغيرات (num1، num2، sum) وهي جميعا من نوع الأعداد الصحيحة لأن هذا البرنامج صمم للتعامل مع الأعداد الصحيحة (يقوم بجمع عددين صحيحين وأظهار النتيجة).

ثانيا://: يمكن الأعلان عن كل متغير بسطر منفصل، ويمكن وضعها جميعا بسطر واحد كما في هذا البرنامج على شرط أن تكون جميع المتغيرات من نفس النوع (هنا جميعها أعداد صحيحة) وذلك لغرض تقليل المساحة التي يكتب عليها البرنامج، على ان يتم الفصل بين متغير وأخر بفارزة. وطبعا العبارة تنتهي بفارزة منقوطة.

ثالثا://: بعد الدالة (main()) لاحظ العبارة التالية ({ no semicolon }) وهي تعني لا تستخدم فارزة منقوطة، وبما أنها وضعت بعد العلامة (//) فأن ذلك يعني أنها ملاحظة أو تعليق (Comment) للمستخدم أو القاريء بعدم استخدام الفارزة المنقوطة بعد كلمة ((main)) هذه العبارة التي أعتبرت تعليقا كتبت ووضعت بعد

العلامة (//)، وسوف لا يكون لها تأثير على تنفيذ البرنامج (أي أنها تهمل أثناء تنفيذ البرنامج)، عليه فسيكون عندك قاعدة وهي " أن أي عبارة تستخدم لغرض التوضيح أو التعليق ممكن كتابتها داخل البرنامج وحسب القواعد التي تم التطرق لها في الفصل الأول، وسوف لا تكون جزء من البرنامج أثناء التنفيذ (تهمل)".

ملاحظة://

التعليقات أو الملاحظات توضع لأي شرح معين لبعض الأوامر أو الإجراءات التي تكون معروفة لدى المبرمج وغير معروفة للمستخدمين، أيضا تستخدم لكتابة بعض المعلومات حول البرنامج (كوقت انشائة أو تحديثة) أو معلومات حول المبرمج نفسه (مثلا الأسم ، العنوان الالكتروني).

التعليقات ممكن أن توضع في أي مكان في برنامج C++، ولكن يفضل أن تكتب في بداية البرنامج (في حالة كون المعلومات عن وظيفة البرنامج أو معلومات عن المبرمج)، أو تكتب بجانب الأوامر التي تحتاج الى توضيح .

رابعا:// كما سبق وأن ذكرنا أن تنفيذ البرنامج يتم بالتسلسل من الأعلى الى الأسفل فيبدأ من الموجهة (#include) ثم العبارة (() main)، وبعدها أمر بداية البرنامج ({}) (والتي تعني أن ما بعدها هي أوامر برمجة مطلوب من الحاسوب تنفيذها، يلي ذلك قراءة المتغيرات، بعدها ينفذ أمر الطباعة (لاحظ الموجود بعد العلامة (>>) في أمر الطباعة هو محصور بين علامتي أقتباس لذا فأنه يطبع كما هو) هذه العبارة ستظهر على شاشة التنفيذ وهي تخبر المستخدم مايلي (أدخل الرقم الأول: input number1) وهي بشكل عام يمكن الأستغناء عنها دون أن يتأثر البرنامج.. ولكنها مفيدة حيث تخبر المستخدم عن الخطوة أو الخطوات الواجب أتباعها لأنجاز تنفيذ البرنامج، (يمكن ملاحظة مثل ذلك في البرامج التي تعملون عليها مثلا في برنامج للعبة (game) معينة فأن هناك ملاحظات ستظهر على الشاشة لأرشاد

المستخدم عن الخطوات الواجب أتباعها لتشغيل اللعبة أو أختيار درجة الصعوبة وغيرها).

خامسا:// هنا تبدأ عملية أسناد قيمة للمتغير (num1) وذلك باستخدام الأمر (cin>>)، عند الوصول الى هذه الخطوة فأن شاشة التنفيذ (الشاشة السوداء) ستظهر ويكون هناك مؤشر صغير على شكل شارحة (_) يظهر ويختفي (ينبض) في موقع على الجانب الأيسر من شاشة التنفيذ، هذا المؤشر يحفز المستخدم على طباعة قيمة على الشاشة (طباعة قيمة معينة باستخدام لوحة المفاتيح)، وبعد أن تطبع هذه القيمة يتم أعلام (المعالج) بانجاز العمل وذلك من خلال الضغط على الزر (Enter)، في هذه الحالة سيتم قراءة القيمة التي طبعت على الشاشة وخزنها في الموقع الذي يؤشر عليه المتغير الموجود بعد الأمر (cin>>) وبذلك تكون قد أسندت قيمة للمتغير (num1) (خزن قيمة) في الموقع الذي يؤشر عليه المتغير في الذاكرة بعد هذه الخطوة، وهذا ما أسمية الأسناد الذي يتم بواسطة المستخدم أثناء تنفيذ البرنامج.

سادسا:// الأمران اللاحقان هما مشابهان للخطوتين الرابعة والخامسة.

سابعا:// التعبير (sum = num1 + num2)، عند الوصول الى هذا التعبير فأن المترجم سيبدأ بالطرف الأيمن من التعبير ويقوض عن المتغيرات الموجودة بما يساويها من قيم (هذه القيم تم اسنادها الى المتغيرات من خلال الامر cin>> والذي اشرنا له)، بعدها يتم أجراء عملية الجمع على هذه القيم لينتج عن ذلك قيمة واحدة في الطرف الأيمن، هذه القيمة ستوضع (تخزن) في الموقع الذي يؤشر عليه المتغير الموجود في الطرف الأيسر، وبذلك فان المتغير (sum) ستسند له قيمة (تخزن في الموقع الذي يؤشر عليه في الذاكرة) من خلال المعادلة، وهذا ما أسميه أسناد قيمة اثناء كتابة البرنامج (أي أن المستخدم لا يتدخل في ذلك أثناء تنفيذ البرنامج).

ثامنا:// بعد أنجاز العمل المطلوب من البرنامج فلا بد من أعلام المستخدم بالنتيجة المتحصلة من أجراء البرنامج، ويتم ذلك من خلال طباعة القيمة

المتحصلة والتي هي الأن موجودة في المتغير(sum)، لذا تم استخدام أمر الطباعة ليطبع ما موجود بعد العلامة (<<) ولما كان ما موجود بعد هذا العامل غير محدد بعلامتي أقتباس لذا فان القيمة المخزونة في الذاكرة في الموقع الذي يشير عليه المتغير(sum) هي التي تظهر على شاشة التنفيذ (اي ان المترجم يعوض اولا قيمة المتغير sum في امر الاخراج وبعدها تتم طباعة القيمة).

تاسعا:// الأمر الأخير هو (}) الذي يمثل نهاية البرنامج.

ملاحظة://

بشكل عام فان استخدام القوس المتوسط المفتوح ({) والذي يشير الى البداية يجب أن يقابلة قوس متوسط مغلق يشير الى النهاية (}) ، عليه فأن عدد الأقواس المتوسطة المفتوحة في البرنامج الواحد تساوي عدد الأقواس المتوسطة المغلقة في ذات البرنامج، أما الأستثناءات فسنشير لها في موضعها .

ملاحظة://

في أدناه بعض القواعد التي يجب أن تلاحظ عند أدخال البيانات المطلوبة :

يجب أن يتطابق نوع القيمة المدخلة لمتغير معين مع النوع المعلن لهذا المتغير .

أذا كانت هناك رغبة في أسناد قيم لأكثر من متغير في أيعاز قراءة واحدة فيجب أن يفصل بين متغير وأخر بواسطة العامل (>>).

يجب أن يتطابق عدد البيانات التي يتم أدخالها مع عدد المتغيرات المدونة بعد العامل (>>) في أيعاز القراءة.

اذا كان أكثر من متغير واحد في ايعاز قراءة واحد فيمكن أدخالها جميعا ثم ضغط الزر (Enter) على أن يفصل بين قيمة وأخرى فراغ، أو تدخل القيم واحدة بعد الأخرى على أن تضغط الزر (Enter) بعد أدخال كل قيمة .

لا يجوز أن تكون القيم المدخلة صيغ رياضية (أي قيم بينها علامات رياضية)

ملاحظة://

من الممكن استخدام العوامل (<<)، (>>) بشكل متكرر مع عبارات الادخال والاخراج (cout OR cin) لتفيد تكرار أمر الادخال والاخراج. مثال

```
cout << x << y << z ;
cin >> x >> y >> z ;
```

2.4 بعض المواضيع الأخرى في عمليات الإدخال والأخراج

Formatted Consol for I/O Operations

دعم C++ عدد من الصفات التي من الممكن ان تستخدم لصياغة او تنظيم طريقة ظهور المخرجات والموضحة بالجدول (2.1)، هذه الدوال تستخدم مع الموجة (iostream) او مايكافئها مع (iomanip) وهي تستخدم بالترافق مع الأمر (cout)، والصيغة العامة لها هي:

cout.function

لاحظ هنا تم استخدام *النقطة* ((.) بدلا من <<).

جدول (2.1): بعض الصفات المهمة التي تستخدم لصياغة او تنظيم المخرجات

دوال مع الموجة #include<iostream>	دوال مع الموجة #include<iomanip>	وضيفة الدالة
width ( )	setw ( )	تحدد حجم الحقل المطلوب لعرض قيم المخرجات
precision ( )	setprecision ( )	تحدد عدد المراتب بعد الفارزة عند عرض القيم الحقيقية
fill ( )	setfill ( )	تحدد نوع الرمز الذي سيستخدم لملأ الجزء غير المستخدم في الحقل المحدد لعرض قيمة

setf ( )	setiosflags ( )	تحدد اشارة للمسيطر لتحدد نوع الصياغة المطلوبة (مثل طباعة القيمة من اليمين او اليسار,ملأ السطور)
unsetf ( )	setiosflags ( )	تستخدم لألغاء الصياغة المحددة بالأيعاز السابق

مثال:

```
cout.width (5) ;
cout << 345 ;
```


المخرجات ستكون كما يأتي:

		3	4	5

اي ان المترجم سيحدد خمس مواقع لطباعة القيمة، ويبدأ الطباعة من اليمين، لذلك سيكون هناك فراغين في اليسار.

ملاحظة://

تأثير الدالة width () يستمر لأمر طباعة واحد فقط، فاذا كان هناك اكثر من امر طباعة فنستخدم width ()) مع كل امر طباعة..

ملاحظة://

يستخدم الأمر fill()) (لملا الفراغات، ويجب ان تضع بين قوسي الأمر fill() الرمز المطلوب طباعتة (بما انه رمز فيجب ان يحدد بحاصرات مفردة). اما أذا لم يحدد ماهية الرمز المطلوب طباعتة في الحقول الفارغة (عند تحديد حجم الحقل لطباعة قيمة معينة) فأن المترجم سيتركها فارغة كما في المثال السابق .

مثال

```
cout.fill ( '*
```

```
cout.width ( 7 ) ;
cout << 345 ;
```

في هذه الحالة فان الحقول الفارغة ستملأ بالعلامة (*) وستكون النتيجة:

*	*	*	*	3	4	5

ملاحظة://

في حالة تحديد عدد المراتب بعد الفارزة فأن تأثير الدالة سيستمر على كل القيم اللاحقة لحين الغاء أو أعادة التحديد . مثال

```
cout.precision ( 10 ) ;
```

هذا يعني ان كل الأرقام الحقيقية اللاحقة سيحدد لها عشر مراتب بعد الفارزة .

ملاحظة://

أذا لم يحدد عدد المراتب التي بعد الفارزة للأرقام الحقيقية فأن المترجم سيفرضها ست مراتب .

ملاحظة://

من الملاحظ في جميع الأمثلة أعلاة أن الطباعة تبدأ من اليمين الى اليسار وهي الحالة الأفتراضية (default) للحاسوب، أما أذا كان المطلوب غير ذلك فهناك دالة خاصة لهذا الغرض سنأتي عليها ((setf ( ))، والتي لها استخدامات مختلفة .

* الدالة (setf()) تعمل مع الأمر (cout) كما بينا ولكنها تختلف بعض الشئ عن الدوال الأخرى المشار اليها أعلاه حيث أنها من الممكن أن تأخذ معامل واحد أو معاملين (وسيط او اثنين)، ووفقا لهذه المعاملات سيحدد واجبها وكمايأتي:

1. الدالة مع وسيط واحد تكون العملة العامة لها كما يأتي:

cout.setf (arg1 ،arg2) ;

ويكون استخدام هذه الدالة وفقا لما موضح في الجدول (2.2).

جدول (2.2): يبين وظيفة الدالة (setf()) مع استخدام اثنين من الوسائط

قيمة الوسيط الثاني bit-field (arg2)	قيمة الوسيط الأول(flagarg1)	وظيفة الـــــــدالة
ios::adjustifield	ios::left	ملأ السطور من اليسار
ios::adjustifield	ios::right	ملأ السطور من اليمين
ios::adjustifield	ios::internal	اظهار العلامات الرياضية (الاشارة الموجبة والسالبة)
ios::floatfield	ios::scientific	العلامة العلمية
ios::floatfield	ios::fixed	علامة النقطة الثابتة
ios::basefield	ios::dec	الأساس العشري
ios::basefield	ios::oct	الأساس الثماني
ios::basefield	ios::hex	الأساس السادس عشر

مثال://

```
cout.fill ('@') ;
cout.precision (3) ;
cout.setf( ios::internal ،ios::adjustifield) ;
cout.setf (ios:: scientific ،ios::floatfield);
cout.width (15) ;
cout << -12.34567 <<"\n" ;
```

نتيجة هذا المثال هي:

-	@	@	@	@	@	1	.	2	3	5	e	+	0	1

تلاحظ ان الأيعاز الأول هو لملأ الفراغات بالرمز(@)، اما الايعاز في السطر الثاني فهو يمثل عدد المراتب بعد الفارزة للرقم الحقيقي وهي هنا (3)، الايعاز الثالث فهو يستخدم معاملين او وسيطين لاظهار العلامة الرياضية، الايعاز في السطر الرابع يستخدم لأظهار العلامة العلمية، ثم تم تحديد عدد المواقع التي ستطبع بها القيمة والتي حددت (15 موقع).. واخيرا تم ادخال القيمة المطلوب طباعتها (لاحظ النتيجة).

2. استخدام وسيط واحد مع الدالة (setf ()) والصيغة العامة لها هي:

```
cout.setf (arg) ;
```

واعتمادا على قيمة الوسيط تقوم الدالة بعملها.

الجدول (2.3) يبين وظيفة الدالة (setf()) عند استخدامها وسيط واحد ووفقا لقيمة الوسيط المقابل لها

جدول (2.3): وظيفة الدالة (setf()) عند استخدام وسيط واحد

قيمة المعامل (flag)	وظيفة الـــــــدالة
ios::showbase	تستخدم base indicator في المخرجات
ios::showpos	تطبع العلامة الموجبة (+) قبل الارقام الموجبة
ios::showpoint	تظهر الفارزة والأصفار
ios::uppercase	تستخدم الحروف الكبيرة في المخرجات الممثلة بالنظام السادس عشري
ios::skipus	حذف الفراغات (white space) في المخرجات
ios::unitbuf	تدفق كل (stream) بعد الحشر
ios::stdio	تدفق (stdout and stderr) بعد الحشر

* برنامج لايجاد الجذر التربيعي للرقم 5 مع تنظيم المخرجات، وكذلك الجذر التربيعي للرقم 100 باستخدام العلامة العلمية.

```
// Example 2.4
#include <iostream>
using namespace std;
#include <math>
main()
{  cout.fill('*') ;
   cout.setf(ios::left ،ios::adjustfield );
   cout.width(10);   cout << "value";
   cout.setf(ios::right ،ios::adjustfield);
   cout.width(15);
   cout<<"sqrt of value"<<"\n";    cout.fill('.');
   cout.precision(4);
   cout.setf(ios::showpoint);
   cout.setf(ios::showpos);
   cout.setf(ios::fixed ،ios::floatfield);
    cout.setf(ios::internal ،ios::adjustfield);
    cout.width(5);
    cout<<5;
    cout.setf(ios::right ،ios::adjustfield);
    cout.width(20);
    cout<<sqrt(5)<<"\n";     }
    cout.setf(ios::scientific ،ios::floatfield);
    cout<<"\nsqrt(100)="<<sqrt(100)<<"\n";
    return 0;
}
```

مخرجات البرنامج 2.4://

```
value * * * * * * sqrt of value
+ . . . . 5 . . . . . . . . . . . . . +2.2361
sqrt ( 100 ) = +10000e+01
```

سيتم شرح ايعاز التكرار الوارد في المثال (2.4) في الفصل الرابع.

ملاحظة://

تستخدم setw() مع الأعداد والسلاسل الرمزية.

ملاحظة://

يستخدم الأيعاز (cin.get(ch)) لأسناد حرف للمتغير الحرفي (ch) أثناء تنفيذ البرنامج حتى وأن كان فراغ أو سطر جديد، مثال

```
cin >> m ;
cin .get (ch) ;
cin >> n ;
```


الأن لتلاحظ مثلي [illegible]

```
Input 1 :  25 w 34        // m is 25  ، ch is  w  ، n is  34
Input 2 :  33  41         //  m is 33  ، ch is blank  ، n is  41
Input 3 :  67 ( Enter ) 55  //  m is 67  ،ch is newline (\n)  ،n is 55
```

ملاحظة://

الأرقام تمثل داخل الذاكرة بالصيغة الثنائية (binary) وهي تحدد عدد البتات اللازمة لتمثيل ذلك الرقم، لذلك يجب ملاحظة تعريف المتغير بما يتناسب وحجمة، وفي خلاف ذلك فأن النتائج ستكون خاطئة.

* لغرض أخراج القيم العددية الصحيحة وفقا لأساس يتم اختياره مثل (decimal، octal، hexadecimal) فان بامكانك كتابة المختصرات التالية مع أمر الأخراج لتحصل على قيمة عددية وفقا لذلك الاساس:

dec = decimal

oct = octal

hex = hexadecimal

* برنامج لإدخال قيمة عددية وطباعتها بالنظام العشري، السادس عشر، والنظام الثماني.

```
// Example 2.5
# include<iostream>
using namespace std;

main( ) {
int value ;
cout<<" Enter number " << endl;
cin>>value ;
cout<<"Decimal base =" << dec<<value<<endl;
cout << " Hexadecimal base  =" << hex<<value <<endl ;
cout<<" Octal base=" << oct<<value << endl ;
return 0;
}
```


مخرجات البرنامج 2.5 :

```
Enter number
```

```
10
Decimal base =10
Hexadecimal base = a
Octal base = 12
```

لنفس الغرض اعلاه بالأمكان استخدام الأيعاز (setbase ()) والذي يستخدم لأخراج القيم العددية الصحيحة وفقا للأساس المحدد بين القوسين لهذا الأيعاز (بكلام أخر بالأمكان تحويل الاعداد من اساس الى أخر والمقصود بالاساس هنا هو ان الاعداد العشرية (decimal) اساسها (10)، والثماني (octal) اساسها (8)، والسادس عشر (hexadecimal) اساسها (16)). وهذه الدالة تستخدم مع الموجة (#include<iomanip >)

* سنعيد كتابة المثال (2.5) ولكن باستخدام الأيعاز (setbase())

```
// Example 2.6
# include<iostream>
# include<iomanip>
using namespace std;
main( ) {
int value;
cout<<" Enter number " << endl ;
cin>>value ;
cout<<" Decimal base = " << setbase ( 10 ) ;
cout <<  value << endl ;
cout << " Hexadecimal base  =" << setbase ( 16 ) ;
cout << value <<endl ;
cout<<" Octal base=" << setbase ( 8 )  ;
cout<< value << endl;
return 0;
 }
```

chapter 4

Bit Manipulations IN C++

2.5 التعامل مع البتات Bit Manipulations

تعلمنا من المواضيع السابقة عندما نعلن عن متغير فان المترجم يحجز مساحة في الذاكرة لهذا المتغير وحسب نوعة. في الحقيقة، وكما تعلمنا من دراسة البايتات والكلمات، المتغير المعلن عنه يشغل مساحة في الذاكرة عبارة عن مجموعة من الصناديق الصغيرة. فحسب فهمنا الانساني، ليس من السهل دائما ان نفهم كيف يتم خزن حرف مثل الحرف B بثمانية صناديق صغيرة عندما نعرف ان الحرف B هو حرف واحد. ان التعامل مع البتات تسمح لك للسيطرة على كيفية خزن القيم بالبتات. هذه ليست عملية تحتاج الى انجازها كل مرة، خصوصا ليس في المراحل المبكرة من رحلتك مع C++. على الرغم من ذلك، عمليات البتات (والعمليات المتطابقة ذات العلاقة) تقدم في كل بيئات البرنامج التطبيقي، لذا فانك يجب ان تهتم بماذا تعمل وماذا تقدم. في ذلك الوقت فانك يجب ان تهتم بماذا يعني البت، البايت، الكلمة. وقد سبق وان وضحنا في الفصل الاول العوامل المنطقية والتي هي تستخدم مع الشرط وسنستخدم هنا مايشبة ذلك قليلا ولكن تتعامل مع البتات.

2.5.1 عمليات البتات: العامل ~ Bitwise Not

واحدة من العمليات التي من الممكن ان تنجزها على البت تتمثل بعكس قيمتة. عليه فاذا كانت قيمة البت واحد فانها ستتغير وتكون صفر وبالعكس. هذه العملية سوف يقوم بها العامل Not والذي سيرمز له بالرمز (~). ان العامل Not هو عامل احادي اي يكون معه عامل واحد ويكون هذا العامل على الجانب الايسر كما في المثال:

```
~value
```

Bit	~Bit
1	0
0	1

لنفرض رقم بحجم بايت مثل الرقم 248. بالتاكيد فانك تعلم كيف تحول الارقام من نظام الى اخر، فمثلا ان القيمة الثنائية للرقم 248 هي 10001111 (والقيمة بالنظام السادس عشر هي xF80). فاذا نفذت العامل Not عليه لعكس قيم بتاتة، فانك ستحصل على النتيجة التالية:

Value	1	1	1	1	1	0	0	0
~value	0	0	0	0	0	1	1	1

2.5.2 عامل مقارنة البتات (و) The Bitwise AND Operator & :Comparing Bits

Bit1	Bit2	Bit1 & Bit2
1	1	1
1	0	0
0	1	0
0	0	0

العامل And هو عامل ثنائي اي يستخدم مع اثنين من المعاملات ويستخدم وفق الصيغة القواعدية التالية:

Operand1 & Operand2

هذا العامل ياخذ قيمتين ويقارن البت للقيمة الاولى مع البت الذي يقابلة في القيمة الثانية، والنتيجة ستكون وفقا لجدول الصدق المبين ادناه.

تخيل لدينا قيمتان البايت الاولى 187 والثانية 242. استنادا الى دراستنا لانظمة الاعداد فان القيمة الثنائية للعدد العشري 187 هي 1011 1011 (وقيمتة

بالنظام السادس عشر 0xBB)). القيمة الثنائية للرقم العشري 242 هي 00101111 (وقيمتها بالنظام السادس عشر هو(0xF2)، دعنا نقارن هاتين القيميتين بت بت، باستخدام عامل البتات And:

	ثنـــــائي								عشري
N1	1	0	1	1	1	0	1	1	187
N2	1	1	1	1	0	0	1	0	242
N1 & N2	1	0	1	1	0	0	1	0	178

في كثير من الاحيان تحتاج ان يقوم المترجم بانجاز هذه العملية واستخدام الناتج في البرنامج، هذا يعني امكانية الحصول على النتيجة لهذه العملية وعرضها على شاشة الحاسوب، هذه العملية من الممكن ان نوضحها في المثال التالي.

* برنامج لادخال قيمتين واجراء عملية (و) على بتاتهما.

```
// Example 2.7
#include <iostream>
using namespace std;

main(){
const int N1 = 187;
const int N2 = 242;
cout<<N1<<"&"<<N2<<"="<< (N1 & N2)<<"\n\n";
return 0;
}
```


مخرجات البرنامج 2.7 ://

```
187 & 242 = 178
```

2.5.3 عامل المقارنة او Comparing Bit: Bitwise OR Operator (|)

من الممكن ان تقوم بنوع اخر من المقارنة على البتات باستخدام عامل مقارنة البتات OR والذي يمثل بالعلامة (|) والصيغة القواعدية هي:

Value1 | value2

مرة اخرى، فان المترجم يقارن البتات المتقابلة في القيمتين. فاذا كان على الاقل واحد من البتات يساوي 1 فان نتيجة المقارنة ستكون 1. نتيجة المقارنة ستكون صفرا اذا كان البتان المقارنان قيمتيهما صفرا يمكن ملاحظة ذلك في الجدول ادناه:

Bit1	Bit2	Bit1 | Bit2
1	1	1
1	0	1
0	1	1
0	0	0

مرة اخرى دعنا نتعامل مع القيمتين 187 و 242 ونقارن بينهم باستخدام عامل مقارنة البتات OR

	الثنائي							العشري
N1	1	0	1	1	1	0	1	1	187
N2	1	1	1	1	0	0	1	0	242
N1 | N2	1	1	1	1	1	0	1	1	251

وكذلك من الممكن ان تدع المترجم ينجز هذه العملية وتستخدم الناتج في البرنامج.

* برنامج لادخال عددين صحيحين واجراء عملية (او)على بتاتهما وطباعة الناتج.

```
//Example 2.8
#include<iostream>

main(){
const int N1 = 187;
const int N2 = 242;
cout<< N1 << "|" <<N2<<"="<< ( N1 | N2 )<< "\n\n";
return o;
}
```


مخرجات البرنامج 2.8 ://

```
187 | 242 = 251
```

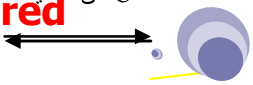
2.5.4 مقارنة البتات باستخدام العامل XOR

Comparing Bits: The Bitwise-Exclusive XOR Operator (^)

مثل العاملين السابقين فان هذا العامل يقوم بمقارنة كل بتين متقابلين في القيمتين، الصيغة القواعدية هي:

value1 ^ value2

Bit1	Bit2	Bit1 ^ Bit2
1	1	0
1	0	1
0	1	1
0	0	0



المترجم سيقارن البت لواحدة من القيم مع البت المقابل للقيمة الاخرى. نتيجة المقارنة تعتمد على الجدول اعلاه:

لناخذ مرة ثانية القيمتين 187 و 242، ونقارن بينهما باستخدام العامل XOR ونتيجة هذه المقارنة كما في ادناه:

	الثنـــــائي								العشري
N1	1	0	1	1	1	0	1	1	187
N2	1	1	1	1	0	0	1	0	242
N1 ^ N2	0	1	0	0	1	0	0	1	73

اذا ما نفذ المترجم هذه العملية فانه سيولد ناتج من الممكن ان يستخدم ضمن البرنامج.

* برنامج لادخال عددين صحيحين واجراء عملية XOR على بتاتهما وطباعة الناتج.

```
//Example 2.9
#include<iostream>
using namespace std;

main(){
const int N1 = 187;
const int N2 = 242;
cout<< N1<< "^" << N2<< "="<< N1 ^ N2 <<"\n\n";
return 0;
}
```

مخرجات البرنامج 2.9 ://

187 ^ 242 = 73

2.5.5 عامل تزحيف البتات لليسار Bit Shift Operators: The Left Shift <<

في المواضيع السابقة، تعلمت ان البتات تنظم بطريقة معينة لخزن البيانات التي تحتاجها. احد العوامل الذي بامكانك استخدامة على البتات يتكون من تحريك البتات باتجاه تختارة. لغة C++ توفر عامل التزحيف لليسار والذي يرمز له (<<) والصيغة القواعدية له هي:

Value << Constant Integer

عامل التزحيف لليسار، هو عامل احادي اي يعمل على قيمة واحدة تكون على يسار العامل ويجب ان تكون القيمة عدد صحيح ثابت. عند تنفيذ هذه العملية، فان المترجم سوف يدفع قيم البتات الى اليسار بعدد محدد مسبقا (Constant Integer) والذي يكون على يسار العامل <<. البتات التي تحرك الى اليسار سوف تختفي

عند التزحيف وعدد البتات التي ستختفي هي بعدد (Constant Integer) ، بعد تزحيف البتات الى اليسار فان الفراغ المتولد في مواقع البتات في الجانب الايمن سيملأ باصفار.

افرض لديك القيمة 42 حيث ان القيمة الثنائية لها هي 00101010 وترغب بتزحيفها الى اليسار مرتبتين كما يأتي:

```
const int N = 42;
N<<2;
```

هنا ستكون النتيجة كما في ادناه

	الثنــــائي								العشري
قبل التزحيف	0	0	1	0	1	0	1	0	42
بعد التزحيف: مرتبتين	1	0	1	0	1	0	0	0	168

لاحظ هنا ان البتان على اليسار اختفت واضيف صفران على اليمين. وهذه العملية من الممكن ان تستخدم ناتجها في البرنامج.

* برنامج لاجراء عملية تزحيف بتات الى اليسار (بمقدار بتان) على القيمة 42.


```
//Example 2.10
#include <iostream>
using namespace std;

main(){
const int value = 42;

cout << value<<"<<2="<<(value<<2)<<"\n\n";
return 0;
}
```


مخرجات البرنامج 2.10 ://

42 << 2= 168

2.5.6 عامل تزحيف البتات لليمين Bit Shift Operators:The Right Shift >>

وهو يعمل عكس عامل التزحيف لليسار، فهو يزحف بتات القيمة المعطاة الى اليمين وفقا [illegible] مشابهة للتزحيف

لليسار ماعدا التزحيف الى الاتجاه المعاكس، لذا لننفذ التزحيف على القيمة 42 الى اليمين بمرتبتين ونلاحظ مايحدث:

	الثنـــــائي								العشري
قبل التزحيف	0	0	1	0	1	0	1	0	42
بعد التزحيف مرتبتين	0	0	0	0	1	0	1	0	9

2.6 أمثله محلوله

* برنامج لتحويل (sec42200) الى ما يقابلها بالساعات والدقائق والثواني.

```
// Example 2.7
#include<iostream>
using namespace std;
main()
{
  int sec =42200 %60;
  int temp =42200 /60;
  int min =temp %60;
  int hour = temp / 60;
  cout<<"hour="<< hour<<" ,min="<< min<<" ,sec="<< sec;
return 0;
}
```


مخرجات البرنامج 2.7 ://

```
hour= 11, min=43, sec=20
```

* برنامج لأيجاد قيمة (y) من المعادلة $y = 4x^2 + 3x - 6$

```
// Example 2.8
#include<iostream>
using namespace std;
```



```
main()
{
  int x, y;
cout << " Enter number ";
  cin>>x;
  y =4*x*x +3*x-6;
  cout<< y;
return 0;
}
```


مخرجات البرنامج 2.8 ://

```
Enter number 10
424
```

* أكتب برنامج لتحويل درجة حرارة مقاسة بالفهرنهايت الى درجة مئوية.

```
// Example 2.9
#include<iostream>
main()
{
  int f;
  cout<<"Enter temperature degree in Fahrenheit "<<endl;
  cin>> f;
  float c =( 5/9)*(f+32);
  cout<< c ;
return 0;
}
```


مخرجات البرنامج 2.9 ://

Enter temperature degree in Fahrenheit

70

56.6666

* برنامج لأيجاد مساحة ومحيط دائرة.

```
// Example 2.10
#include<iostream>
using namespace std;
```

```
main()
{
  const float pi=3.141529 ;
  int r;
  float area,  perimeter;
  cout<<"enter circle radius \n";
  cin>> r;
  area =r*r*pi;
  perimeter =2*r*pi;
  cout<<"area= "<< area<<", perimeter=" <<perimeter;
return 0;
}
```


**مخرجات البرنامج 2.10 :// **

```
enter circle radius
4
area= 50.2644 , perimeter=25.1322
```


* برنامج لأيجاد حاصل ضرب ومعدل ثلاث أرقام.

```
// Example 2.11
#include<iostream>
using namespace std;

main()
{
  int prod, a, b, c;
```

```
 float average;
 cout<<"enter three numbers \n";
 cin>> a >> b >> c;
 prod = a*b*c;
 average =( a + b + c)/3;
 cout<<"prod= "<< prod<< endl;
 cout<<"averge= "<< average;
return 0;
}
```


مخرجات البرنامج 2.11 ://

```
enter three numbers
3 7 9
prod= 189
average= 63
```


اسئلة للحل ://

1. اكتب برنامج لايجاد مربع والجذر التربيعي لاي رقم.
2. صحح جزء البرنامج التالي

```
#include<iostream>
Main() {
Char    gap = ‘ ‘ ;
Int      m ; n ;
float   a, b;
```

```
char   c1, c2
int (a + m) = 12 ;
cin >> a >> b >> m >> n ;;
cout << a+b << c1 ;
gap = a + c2 ;
m = a / b ;
cin > c2 ;
cout << n = a * b ;
if ( a = b)     cout << " equal" ;
else        cout << a not equal b ;
}
```


3. اكتب برنامج لايجاد قيمة العلاقة التالية

$$Y = 3x^2 - 2x + 4$$

4. اكتب برنامج لايجاد مساحة مثلث.

5. اكتب برنامج لإبدال (swap) رقمين واحد بدل الآخر.

القرار والتكرار
DECISION AND REPEAT INSTRUCTIONS

3.1 المقدمة

الأن جاء دور دراسة القواعد الأكثر اهمية في البرمجة. وهي القرار **(if statement)** وكذلك الأيعاز المرافق لها **(else)** وعبارات التكرار التي هي **(for loop، do.. while loop، while loop)**، عالباً تعد هذه الأوامر من الأوامر الكثيرة الاستخدام في البرمجة لذا ننصح بعد الأنتهاء من دراسة هذا الفصل الشروع بكتابة برامج تستخدم فيها هذه القواعد وزيادة الخبرة العملية قبل الأنتقال الى موضوع جديد.

3.2 عبارة اذا (if) Statement

يستخدم هذا الأمر لأتخاذ قرار من قبل المترجم بناءا على بعض المعطيات التي ترد في البرنامج، هناك العديد من الحالات التي لايمكن التنبأ بها من قبل المبرمج أثناء كتابة البرنامج، فعلى سبيل المثال أننا نكتب برنامج لأيجاد الجذر التربيعي لأعداد صحيحة يتم ادخالها من قبل المستخدم أثناء تنفيذ البرنامج، في هذه الحالة وكما معلوم فأن العدد الصحيح يجب أن يكون موجب لأنه لايمكن أيجاد الجذر التربيعي للعدد السالب، السؤال هنا هل يمكن منع المستخدم من أدخال عدد سالب سواء كان بقصد أو سهوا، أن المبرمج سوف لايجد وسيلة أثناء كتابة البرنامج لمعالجة هذا الأشكال البسيط ألا أن يستخدم عبارة القرار (أذا) والتي ممكن أن تكون كما يلي (أذا كان العدد موجب أوجد الجذر التربيعي).. (وبالتأكيد فأن المترجم في الحاسوب لا يفهم عبارة موجب لذا نستبدلها بما يتناسب وقواعد لغة البرمجة C++ فنقول أذا كان العدد أكبرمن أو يساوي صفر فأوجد الجذر التربيعي).

ان استخدام عبارة (if) يكون كما يلي (أذا (شرط)).. (if (condition)) أذا تحقق الشرط الذي يرافق الأمر (if) فيتم تنفيذ العبارة التي بعده أما أذا لم يتحقق

هذا الشرط فيهمل ما بعده (اي تهمل العبارة التي بعده) أذن ستكون طريقة كتابة هذا الأمر كما يأتي:

```
if  conditional expression true     // لتنفيذ فعل واحد
        Statement ;
```

ملاحظة ://
توجد بعد الامر (if) مباشرة قلصف منقوطة .

عادة يكون تنفيذ البرنامج خطوة بعد الاخرى من الاعلى الى الاسفل حسب ترتيب خطوات البرنامج، عبارة (if) تمكنك من اختيار تنفيذ عمل معين وفقا للشرط المحدد (مثلا، فيما اذا كان متغيران متساويان) والتحول الى جزء مختلف من البرنامج حسب النتيجة، من الممكن اعادة كتابة الصيغة القواعدية للامر (if) كمايأتي:

```
if (expression)
 Statement ;
```

كل شيء يعوض بقيمة يسمى تعبير(expression) مثل pi، 23+

التعبير بين القوسين ممكن ان يكون اي تعبير ولكن عادة في هذه الحالة يكون احد التعابير العلاقية (اي التعابير التي يكون احد اجزاءها او اكثر متعلق بالاجزاء الاخرى للتعبير، وعادة يتم استخدام العوامل المنطقية)، فاذا كانت قيمة التعبير مساوية للصفر فسوف يعتبر التعبير (false) اما اذا كانت قيمتة لاتساوي الصفر فيعتبر التعبير (true) وتنفذ العبارة (واقعا المترجم هو الذي يحدد القيمة صفر ام لا استنادا الى كونها صحيحة ام لا)، مثال

```
if (bignumber > smallnumber)
bignumber = smallnumber;
```

نلاحظ هنا ان التعبير يقارن بين الرقم الكبير والرقم الصغير فاذا كان الرقم الكبير اكبر من الرقم الصغير فيتم تنفيذ العبارة التي تاتي بعد (if) مباشرة وهي

مساواة العددين في هذا المثال، واذا لم يكن اكبر فلا يتم تنفيذ عبارة المساواة (في هذا المثال هل سيتم تنفيذ المساواة ام لا ؟)، لاحظ هنا ان قيمة الشرط ستكون لاتساوي صفر اذا كانت التعبير صحيح اي ان الرقم الاكبر اكبر من الرقم الاصغر وتكون صفر اذا كان التعبير خاطيء.

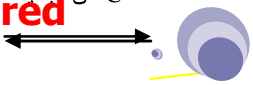
مثال اخر: من الممكن مثلا أن نطلب من أحدهم عملا ونقول له (أذا كان المحل مفتوحا فأجلب لي شراب الببسي)، (get me Pepsi, if shop opening) هذه العبارة ممكن صياغتها برمجيا كما يأتي:


```
if shop_opening
Drink  = Pepsi ;
```

لاحظ في هذا المثال أن الافعال المطلوب أنجازها هي فعل واحد (أن يجلب لنا شراب الببسي)، أما أذا كان ما مطلوب أنجازة هو أكثر من فعل واحد فأن الصيغة ستختلف حيث ستحدد الاعمال الواجب انجازها عند تحقق الشرط بين قوسي البداية والنهاية لتكون كتلة من العبارات التي تعامل على انها عبارة واحدة:

```
if conditional expression TRUE
{
Statements…
}        // لتنفيذ مجموعة من الأفعال
```


ماذا يعني ذلك.. ان الأمر (if) ينفذ عبارة واحدة فقط تأتي بعده والتي تمثل الفعل المطلوب انجازة عند تحقق الشرط، أما أذا كان هناك أكثر من فعل واحد مطلوب أنجازة عند تحقق الشرط فيجب أن تحدد هذه الأفعال للمترجم ويكون ذلك بأن تحددها بين الأمرين ({ }) (واللتان تمثلان البداية والنهاية) وبذلك سيكون واضحا أن الأفعال المطلوب تنفيذها عند تحقق الشرط تبدأ بعد الأمر ({) وتنتهي بالعبارة التي قبل (}).



لنعد الى المثال السابق ونطلب من أحدهم عملا ونقول (اذا كان المحل مفتوح فأجلب لي شراب الببسي وقطعة كيك)

(if shop_opening get me Pepsi, and cake)

الفعل المطلوب أنجازة هنا هو أكثر من واحد حيث المطلوب جلب شراب الببسي وقطعة من الكيك، لذا ستكون صياغة هذه العبارة برمجيا كما يأتي:

```
if shop_opening
{
drink = Pepsi ;
food = Cake ;
}
```


في حالة عدم وضع ({ }) فان أول عبارة ستأتي بعد الشرط الذي بعد الأمر (if) هي التي ستعامل على أنها تعود الى الأمر (if) وتنفذ في حالة تحقق الشرط وهي هنا ستكون (drink) أما العبارة الاخرى فسوف لاتعامل على انها تابعة للامر (if) والتي هي (food) وتنفذ في جميع الاحوال سواء تحقق الشرط ام لا، اما عند استخدام ({ }) فهي دلالة للمترجم على أن الايعازات المحصورة بين ({ }) جميعا مطلوب تنفيذها اذا ما تحقق الشرط.

اذن بالامكان استخدام عبارة واحدة او كتلة من العبارات (block) حيث ان كتلة العبارات تكون بين قوسي البداية والنهاية وكل عبارة تنتهي بفارزة منقوطة. الكتلة تعامل وكانها عبارة واحدة، فالعبارات الثلاثة التالية تعامل مع الامر (if) على انها مكافئة لعبارة واحدة فأما ان تنفذ جميعا او تهمل جميعا:

```
{ temp = a;    a=b;    b= temp;  }
```

مثال اخر

```
if (bignumber > smallnumber)
{
```

```
bignumber = smallnumber ;
cout << " bignumber: " << bignumber <<"\n";
cout<< "smallnumber: " << smallnumber <<" \n";
}
```

هنا لاحظ ان التعبير بعد (if) يقارن بين رقمين احدهما كبير واخر صغير فاذا كان الرقم الكبير اكبر من الرقم الصغير وهو الحال الطبيعي فيجب ان تنفذ العبارات المحددة بين قوسي البداية والنهاية والتي تمثل كتلة واحدة وهما مساواة العددين ثم طباعة العدد الاكبر بعدها طباعة العدد الاصغر اما في حالة كون التعبير (false) فتهمل الكتلة كلها اي العبارات الثلاث.

ملاحظة://

عند الحاجة لاستخدام المساواة في الشرط بعد (if) فلا تستخدم المساواة العادية (=) (assignment) وأنما تستخدم المساواة المزدوجة (= =) لأن استخدام الأولى سيؤدي الى عدم اكمال التنفيذ وظهور رسالة خطأ.

هناك حالة أخرى عند استخدام (if) وهي استخدام (else) لأختيار فعل واحد من أثنين فمثلا في مثالنا السابق ممكن أن يكون الطلب كما يلي (أذا كان المحل مفتوحا فأجلب لي شراب البيبسي وبخلاف ذلك (أي أذا كان المحل مغلقا) فأعمل لي قهوة (if shop_opening, get me pepsi, otherwise get me a coffee) هذه العبارة تنفذ برمجيا كما يأتي:


```
if shop_opening
  Drink  = Pepsi ;
else
  Drink  = coffee ;
```

لاحظ هنا أن حالة الشرط التي بعد (if) عادة أما أن تكون (صح، أو خطأ) (true OR false) أي أما أن يكون المحل مفتوحا أو مغلقا ولا يوجد أحتمال أخر. فأذا كان المحل مفتوحا فالمطلوب أن يجلب شراب وهو الببسي، في خلاف ذلك (else) أي أذا كان المحل مغلقا فليكن الشراب هو قهوة. الملاحظة المهمة هنا هي أنه لايمكن أن ينفذ العملان سوية أي لا يمكن أن يجلب ببسي وقهوة في نفس الوقت والسبب هو أنه لايمكن أن يكون المحل مفتوحا ومغلقا بذات الوقت. عليه فأذا تحقق الشرط (أي الشرط صح بمعنى أن المحل مفتوح) فأن العبارة التي تأتي بعد الشرط الذي بعد (if) ستهمل بينما العبارة التي بعد (else) ستهمل اما أذا كان الشرط غير متحقق (أي أجابة الشرط خطأ بمعنى أن المحل مغلق) فأن العبارة التي بعد (if) ستهمل وتنفذ العبارة التي بعد (else).

المثال التالي مقطع برنامج ممكن أن يكون جزء من لعبة بامكانك ان تضيف اليها أسئله أخرى لتكون لعبة متكاملة:

```
cout<< " Who has discovered the land of America?" ;
cin>> ans ;
if (ans == "Christopher Columbus")
 score = score + 1 ;                    // if condition is true
else                 // if condition is false
 cout << "sorry, you've got it wrong! " ;
```


* برنامج لادخال عددين والمقارنة بينهما (التحقق من قيمة العدد المدخل).

```
// Example 3.1
# include<iostream>
using namespace std;
int main()
{
int firstNumber, secondNumber;
```

```
cout <<"Please enter a big number:";
cin>>firstNumber;
cout<<"\n Please enter a smaller number: ";
cin >> secondNumber;
if ( firstNumber > secondNumber)
cout<<"\nThanks!\n";
else
cout<<"\n Oops. The second is bigger!";
return 0;
}
```


مخرجات البرنامج 3.1

```
Please enter a big number: 10
Please enter a smaller number: 12
Oops.  The second is bigger!
```


ملاحظة://

بالامكان استخدام اكثر من تعبير علائقي في الوقت الواحد بعد (if) مستخدمين العوامل المنطقية للفصل بينها وتحسب نتيجتها وفقا لنتيجة العوامل المنطقية

مثال

```
if ( (x == 5 ) && ( y == 5 ))
if ( (x==5) || ( y==5 ))
if ( ! (x==5))
```

هذه العبارة الأخيره صحيحة عندما (x) لاتساوي 5 و هي نفس العبارة التالية

```
if ( x !=5)
```

ملاحظة://

في لغة C++ فان نتيجة الشرط اذا كانت عبارة خاطئة فسيعيد المترجم القيمة صفر كما بينا واي قيمة لاتساوي الصفر تفسر على ان العبارة صحيحة.

كذلك:

تعني اذا كانت قيمة المتغير لاتساوي صفر اي صح

```
if (x)
x=0;
```


هذه العبارة تكون اكثر وضوحا اذا كتبت بالصيغة التالية

```
if (x!=0)
x=0;
```

كذلك فان العبارة التالية

```
if (!x)
```

تعني اذا كانت x تساوي صفر (false) وهي تكافيء

```
if (x==0)
```


والعبارة الاخيرة اكثر وضوح

ملاحظة://

يفضل استخدام الاقواس حول الاختبارات المنطقية لجعلها اكثر وضوحا كذلك يفضل استخدام الاقواس مع (if) المتداخلة (المركبة) لجعل عبارة (else) اوضح ولتجنب الاخطاء.

3.2.1 عامل الشرط الثلاثي (Conditional Ternary Operator :?)

عامل الشرط الثلاثي يفحص تعبير، ويعيد قيمة معينة اذا كان ذلك التعبير صح، ويعيد قيمة مختلفة اذا كان ذلك التعبير خطأ، هذا العامل هو اختصار لعامل الاختيار (if. else)

الصيغة العامة له:

condition ? result1: result2

فاذا كان الشرط (condition) صح فان التعبير سيعيد القيمة (result1) اما اذا كان خطأ فانه سيعيد القيمة (result2)

مثال:

7==5 ? 4: 3 // يعيد (3) حيث ان (7) لاتساوي (5)

7==5+2 ? 4: 3 // يعيد (4) لان (7) تساوي (2+5)

5>3 ? a: b // يعيد القيمة (a) لان (5) اكبر من (3)

a>b ? a: b // يعيد ايهما اكبر(a) او (b)

هذا التعبير الثلاثي يمكن ان نعبر عنه بما يأتي (اذا كان الشرط صحيحا فعلية ستكون النتيجة هي النتيجة الاولى وبخلاف ذلك اي اذا كانت نتيجة الشرط غير صحيحة فستكون النتيجة هي النتيجة الثانية) عادة هذه القيمة المعادة يجب ان تسند الى متغير. مثال


```
{
int min ،i=10 ،j=20;
min =(i < j ? i: j);
cout<<min;
}
```

* برنامج لأدخال عددين وطباعة الاكبر

```
// Example 3.2
```

```
# include<iostream>
using namespace std;

int main()
{
int x, y, z;
cout<<"Enter two numbers.\n";
cout<<"First: ";
cin>>x;
cout<<"\n Second: ";
cin>>y;
cout<<"\n";
if (x > y)
z=x;
else
z = y;
cout<<"z:"<<z;
cout<<"\n";
z= (x > y) ? x : y;
cout<<"z:"<<z;
cout<<"\n";
return 0;
}
```

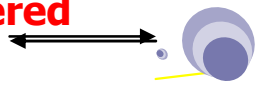

مخرجات البرنامج 3.2:

```
Enter two numbers. First: 5
Second: 8
z:8
z:8
```

3.3 اذا المركبة Compound if

من الممكن أن تستخدم (if) بشكل متداخل مع (if_OR_else) أخرى، وبهذه الحالة تسمى مركبة (أي ممكن أن يكون بعد الشرط الذي بعد (if) عبارة (if) أخرى وممكن أيضا بعد عبارة (else) وممكن أن تكون أكثر من عبارة (if) واحدة. **فمثلا** تريد أن تفحص نوعية رمز معين ووفقا لذلك تقرر ماهو الأجراء الواجب أتباعة وكمايأتي:

```
if (expression1)
  {   if (expression2)
        Statment1;
else
    {  if (expression3)
         Statment2;
       else
         Statment3;
     }
  }
else
   Statment4;
```



مثال

```
if (charkind == digit )
Readnumber ;
else
   if  (charkind == letter)
      Readname ;
   else
      send error message  ;
```


لتتأمل هذا المثال, في البداية يتم فحص الشرط لمعرفة نوع الرمز للمتغير (charkind) هل هو رقم (digit) أم لا، وكما تعلمت دائما أن الأجابة أما نعم (صح) أو لا (خطأ) ولا يوجد أحتمال اخر، فأذا كان صح معناه أن الرمز من نوع (digit)، عليه تنفذ العبارة التي بعد (if) مباشرة أي أقرأ رقم (هذا الأحتمال الأول)، أما الأحتمال الثاني فتكون اجابة الشرط خطأ أي أن نوع الرمز هي ليست أرقاما عليه فستهمل العبارة التي بعد (if) وتنفذ العبارة التي بعد (else)، عندما يحين الدور لتنفيذ العبارة التي بعد (else) لاحظ أن هذه العبارة هي أيضا عبارة (if) هذا يعني أنه لازال هناك أحتمالات أخرى يجب أن نفحص فممكن أن يكون الرمز هو (letter) أو شيء أخر وتطبق نفس القاعدة فأذا كانت أجابة الشرط صح تنفذ العبارة التي بعد (if) (الثانية) أما أذا كانت الاجابة خطأ فتنفذ العبارة التي بعد (else) (الثانية) والتي هي أصدار رسالة خطأ (أي أعلام المستخدم أن هذا الرمز هو ليس (digit OR letter). عبارات (if) هذه تسمى أيضا عبارات (if) المتداخلة (nested If statements).

مثال أخر:

```
if ((ch >= '0') && (ch <= '9'))
Kind = Digit;
```

```
else {
if ((ch >= 'A') &&( ch <= 'Z'))
Kind = upperletter;
else {
if ((ch >= 'a') && (ch <= 'z'))
Kind = lowerletter;
else
Kind = special;
}
```


ملاحظة://

دائما تستخدم (if) عندما تحتاج أن تختار بين أكثر من حالة (أي أختيار عمل أو حالة واحدة من بين أثنين أو أكثر) .

* برنامج لادخال عددين وايجاد امكانية قسمة العدد الاول على الثاني.


```
// Example 3.3
#include<iostream>
using namespace std;
int main() {
int firstNumber, secondNumber;
cout<<"Enter two numbers.\nFirst:";
cin>>firstNumber;
cout<<"\nSecond:";
```

```
cin>>secondNumber;
cout<<"\n\n";
if (firstNumber >= secondNumber)
{
if((firstNumber%secondNumber)==0)  //evenly divisible?
{
if(firstNumber==secondNumber)
cout<<"They are the same!\n";
else
cout<<"They are evenly divisible!\n";
}
else
cout<<"They are not evenly divisible!\n";
}
else
cout<<"Hey!The second one is larger!\n";
 return 0;
}
```


مخرجات البرنامج 3.3:

```
Enter two numbers. First:10
Second: 2 They are evenly divisible!
```

3.4 عبارة التكرار do.. while LOOP

يستخدم هذا الأمر لتكرار عبارة أو أكثر لعدد من المرات وفقا لمتطلبات البرنامج والتي يحددها المبرمج، في هذا الأمر فان البرنامج سينفذ العبارات بين

(do) و (while) على الأقل مرة واحدة.. ويكون توقف البرنامج اعتمادا على شرط يوضع بعد (while).

التكرار يبدأ بالأمر (اعمل أو كرر) (do) ثم مجموعة من الايعازات المطلوب تكرارها وتنتهي بالأمر (طالما) (while) الذي يكون بعده شرط (أي لغاية عدم تحقق هذا الشرط)، المترجم حين يجد العبارة (do) فأنه سيقوم بأعادة تنفيذ العبارات المحصورة بين هذا الأمر والأمر (while).. في كل مرة يصل المترجم الى الأمر (while) يفحص الشرط الذي بعده فأذا كان الشرط متحقق (أجابتة true) فأن المترجم سيعود الى الأمر (do) ويبدأ بالتنفيذ من الامر (do) نزولا من جديد الى الامر (while)، هذه العملية تستمر لغاية عدم تحقق الشرط وتكون أجابتة (false). الصيغة القواعدية لهذا الأيعاز هي:


```
do {
Instruction 1 ;
Instruction 2 ;
etc…      }
while (condition is true) ;
```


* برنامج بسيط واجبة اختبار الحرف (YN) وطباعتة اذا لم يكن (Y)، البرنامج لا يتوقف لغاية أدخال ألحرف (Y).

```
//Example 3.4
#include<iostream>
using namespace std;

main()
{
```

```
char YN ;
 cout<< " enter character?" ;
 cin>>  YN ;
 if (YN != 'Y')
   cout<<  YN ;
   cin>> YN ;
 if (YN != 'Y')
   cout<< YN ;
   cin>> YN ;
 if   (YN !=  'Y')
   cout<<  YN ;
   cin>>  YN ;
 if (YN != 'Y')
   cout<< YN ;
    …
    …
    …
```


هذا البرنامج ممكن أن يستمر بعدد كبير من الخطوات المتشابهة وحسب عدد الحروف المراد طباعتها، أن العبارات (اقرأ، اذا، وأكتب) (cin, if, and cout) تتكرر بأستمرار في البرنامج أعلاه، لذا فان لغة البرمجة C++ أوجدت البديل الذي يسهل العمل ويختصر عدد الخطوات، هذا البديل هو عبارات التكرار، واحدة من هذه الأوامر هو (do.. while) وأذا ما أعدنا كتابة البرنامج أعلاه ولكن مع استخدام (do.. while)، سينتج لنا البرنامج التالي:

```
// Example 3.5
```

```
#include<iostream>
using namespace std;

main()
{
 char YN;
  cout << " enter character? " ;
  do {      //repeat the code for at least one time
    cin>> YN ;
    cout<< YN  ;
     }
  while (YN != 'Y');
return 0;
}
```

ميزة هذا الأمر أن الشرط هو في نهاية التكرار ولذا فأنه سينفذ ولو لمرة واحدة قبل أن يتم فحص الشرط لأجل الأوامر لاحظ كيف أن البرنامج أصبح أكثر أختصارا وأسهل للمتابعة.

3.5 عبارة التكرار while LOOP

وهي أيضا من عبارات التكرار وهي تشابة الى درجة كبيرة الأيعاز (do.. while) اذ أن واجب الأيعازين هو التكرار لمرات غير محددة ابتداء، وأنما يعتمدان على تحقق شرط معين لأيقاف التكرار، الصيغة القواعدية لهذا التكرار هي:

```
while <condition is true> {
instruction 1;
```

```
instruction 2;
instruction 3;
etc...
}
```

ماذا يعني هذا الأمر (عندما يتحقق الشرط نفذ العبارات التي تلي الامر while) وفي كل مرة سينفذ الأيعاز او الايعازات التي بعدة مباشرة والمتعلقة بالامر (while) ليفحص الشرط هل هو متحقق أم لا فإذا كان متحققا ينفذ وأن كان غير متحقق سيهمل الأيعاز الذي بعد (while) وينفذ ما بعده.

ملاحظة://

كما هو الحال في (if and else) فان الأمر (while) ينفذ عبارة واحدة فقط والتي تأتي بعده مباشرة، أما أذا كان هناك أكثر من عبارة واحدة مطلوب تكرارها ضمن الامر (while) فيجب أن تحدد بين قوس البداية ({) وقوس النهاية (}) لتكون كتلة تنفذ جميعا .

اذن لمقارنة ((While) و (do..while).. لاحظ الجدول (3.1):

جدول (3.1): المقارنة بين أمري التكرار (while، do..while)

While	do _ while
الشرط في بداية التكرار	الشرط في نهاية التكرار
لا ينفذ أي أيعاز مالم يتم فحص الشرط والتأكد من تحققة	سيتم تنفيذ الأيعاز او الايعازات المشمولة بالتكرار على الاقل مرة واحدة قبل أن يتم فحص الشرط
تعيد تنفيذ الايعازات المشمولة بالتكرار عند تحقق الشرط	تعيد تنفيذ الايعازات المشمولة بالتكرار عند تحقق الشرط
غالبا ما يستخدم مع طلبات التكرار غير المحددة بعدد ثابت من التكرارات مسبقا	غالبا ما يستخدم مع طلبات التكرار غير المحددة بعدد ثابت من

	التكرارات مسبقا
يعتمد أستمرار التنفيذ على تحقق الشرط ويتوقف التنفيذ عند عدم تحقق الشرط	يعتمد أستمرار التنفيذ على تحقق الشرط ويتوقف التنفيذ عند عدم تحقق الشرط

تنفيذ عبارة (while) كما يأتي:

1. حساب قيمة الشرط بين القوسين لينتج (صح، او خطأ) (true, or false)
2. فأذا كانت نتيجة الشرط خطأ (false) فسوف لاينفذ المترجم ما موجود في جسم (while) اي لاتكون هناك عملية تكرار ويستمر تنفيذ العبارات التي تلي جسم (while).
3. أما اذا كان الشرط (صح) (true) فيتم تنفيذ كل العبارات داخل جسم (while) اي كل العبارات المحددة بين قوسي البداية والنهاية للامر (while) بعدها العودة الى الخطوة (1) اعلاه.

هذه العملية تسمى تكرار لان الخطوة (3) تعود وتكرر الخطوات (1..3)

ملاحظة://

يجب ان يتم في جسم while تغيير قيمة واحدة او اكثر من المتغيرات الموجودة في الشرط وذلك للمساعدة على ان يكون الشرط (false) وانهاء التكرار.

* برنامج لأدخال مجموعة أرقام وطباعتها بشرط يتم التوقف عند أدخال الرقم (0).

```
//// Example 3.6
#include<iostream>
using namespace std;
main()
{
```

```
int x;
  cout<< " Enter number";
  cin>> x ;
  while (x != 0)
    {
      cout<< x ;
      cin >> x ;
    }
return 0;
}
```


شرح البرنامج://

المطلوب من البرنامج أدخال مجموعة أرقام بشرط أن يتوقف عند أدخال الرقم (0)، أذن لما كان أدخال مجموعة أرقام فهذا يعني أنك ستكرر أمر الأدخال أكثر من مرة وفي كل مرة يجب فحص الرقم لغرض طباعتة أذا لم يكن يساوي (0) هذه العملية ممكن تكرارها 5 مرات 10 مرات 1000 مرة أو أكثر حسب طبيعة العمل (تصوروا برنامج يتكون من هذا الكم الهائل من الخطوات المتشابهة !!) لذلك لتجنب عملية تكرار كتابة مجموعة من الايعازات المتشابهة تم أيجاد ايعازات التكرار، فيمكن هنا أن تستخدم الأمر (While) لأختصار البرنامج، هذا الأمر يحتاج الى شرط لغرض العمل والتوقف، في هذا المثال البرنامج يتوقف عند ورود الرقم (0)، أي أنه يعمل مع الأرقام الأخرى ولما كان الشرط يجب أن يكون (true) لكي يعمل أذن أي رقم لايساوي صفرا سوف يجعل البرنامج يعمل لذا جعلنا (x != 0)، لقد سبق وأن بينا أن المترجم عندما يصل الى أي خطوة فيها متغير سيقوم بعملين الأول يتأكد من تعريف المتغير (اي الاعلان عن نوعة)، والثاني يتأكد من أن المتغير له قيمة وحسب النوع المعلن عنه. لذا فأنه عندما يصل المترجم الى الأمر (While) يجب أن يجد قيمة للمتغير (x) وهذا هو السبب الذي جعلنا نسند

قيمة للمتغير (x) قبل الأمر(While) وأن لم تقم بذلك فأن البرنامج سيفشل لعدم وجود قيمة محددة للمتغير (x). كذلك لما كانت هناك أكثر من خطوة مشمولة بالتكرار والتي هي الطباعة والقراءة عليه تم تحديدهما بين القوسين المتوسطين اللذان يمثلان البداية والنهاية ({ }).

ملاحظة://

في كل مرة يتم قراءة قيمة جديدة للمتغير (x) فأن القيمة السابقة ستزول وتحل محلها القيمة الجديدة وهذه قاعدة عامة يجب أن تلاحظ.

ملاحظة://

من السهل كتابة حلقة بشكل عفوي، شرطها يصبح متحققا دائما، هذا سيؤدي الى برنامج مقفل أو مغلق ويستمر بالتنفيذ الى مالانهاية .

ملاحظة://

يتم أختيار الشرط بعد الأمر (while) بحيث يساعد حلقة التكرار أن تستمر طالما كان هذا الشرط متحقق، وأن تتوقف الحلقة عن التكرار عندما لا يتحقق هذا الشرط .

في حالة الأمر (do..while) فأن الشرط ياتي بعد (while) في نهاية حلقة التكرار لذا يجب أن يتم أختيارة بحيث عندما يتم فحصة تكون النتيجة (true) أي متحقق، لكي يستمر التكرار بالعمل ومتى ما أصبحت نتيجة فحص الشرط (false) فأن التكرار يتوقف .

ملاحظة://

عند استخدام الامر (while) فيجب ملاحظة ان المتغير الذي يستخدم معها في الشرط يجب ان تكون له قيمة قبل الدخول الى حلقة (while) وهذه القيمة هي

بطاقة الدخول الى حلقة التكرار (while) وبعد الدخول الى حلقة التكرار يجب ان تتغير قيمة هذا المتغيرداخل الحلقة (حلقة التكرار) بما يساعد على انهاء التكرار.

* برنامج لطباعة كلمة معينة عدد من المرات

```
// Example 3.7
#include<iostream>
using namespace std;
int main()
{
int counter;
cout<<"How many hellos?";
cin>>counter;
do
{
cout<<" Hello\n";
counter - - ;
}
while(counter>0);
cout<<"Counter is:"<<counter<<endl;
return 0;
}
```


مخرجات البرنامج 3.7 :

```
How many hellos?3
```

```
Hello
Hello
Counter is:0
```

3.6 رماوأ التكرار for Loop

هذا الأمر يقوم بتكرار statement أو مجموعة statement لعدد من المرات المحددة مسبقا. والصيغة القواعدية له هي:


```
for (initialization ; test ; action)
  statement;
```

أو ان يكتب حسب الصيغة العامة التالية:

```
for (initial_value ; condition ; increment)
  statements ;
```

عبارة البدء (initialization) تستخدم لبدء حالة العداد اي اسناد قيمة ابتدائية للعداد (initial_value) او التحضير لحلقة التكرار، اما العبارة (test) فهي تعبير في لغة C++ وهي عبارة عن علاقة تمثل الحالة التي من المفروض ان يستمر فيها التكرار وبكلام اخر هي الشرط (condition) الذي عند عدم تحققة تتوقف عملية التكرار (اذا كان هذا الشرط (true) فسيتم تنفيذ العبارة التي بعد الامر for والتي تمثل جسم امر التكرار). اما الجزء الثالث من الايعاز هو action وهو يمثل العداد (عادة يتم زيادة او انقاص العداد حسب طبيعة التكرار والذي هو (increment)).

عبارة for عبارة ذات امكانيات كبيرة ومفيدة ومرنة لدرجة عالية ويمكن ان نوجز تنفيذها بثلاث خطوات:

1. تنفيذ العبارة الاولى في راس الامر for والتي هي اسناد قيمة ابتدائية للمتغير الذي سيعمل كعداد.
2. تقييم الشرط (condition) (true, or false).

اذا كانت قيمة الشرط (true) فيتم تنفيذ العبارة / العبارات (statement/s) والتي تمثل جسم الامر for، اذا كان جسم التكرار يتكون من اكثر من عبارة واحدة، عند ذلك يجب ان تحدد ككتلة بين قوس البداية وقوس النهاية. اما اذا كان الشرط خاطيء false فسيتم اهمال العبارة /العبارات في جسم امر التكرار والانتقال الى تنفيذ الاوامر التي بعده ان وجدت.

3. اما الخطوة الثالثة فهي تنفيذ الجزء الثالث من امر التكرار for والتي تمثل عداد يعد عدد مرات التكرار التي حدثت سواء كان العداد للزيادة او للنقصان حيث في كل مرة يتم فيها تنفيذ العبارات في جسم حلقة التكرار يتم زيادة او انقاص العداد حسب طبيعة الامر وحسب كمية الزيادة او النقصان المحددة لكل مرة.

وبعد كل عملية تنفيذ لجسم حلقة التكرار يتم العوده الى الخطوة (2).

ملاحظة://

في لغة C++ من الممكن أن يكون مكان أي تعبير (expression) في عبارة (for) فراغ، أمثلة :

```
for ( e1 ; e2 ;    )
for (     ; e2 ;    )
for (     ;     ;    )
```


* برنامج لطباعة كلمة معينة عشرة مرات

```
// Example 3.8A
#include<iostream>
using namespace std;
main()
{
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
```

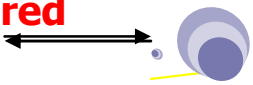
```
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
  cout<<" Hello ‹C++\n";
return 0;
}
```


نعيد كتابة هذا البرنامج باستخدام حلقة تكرار

```
// Example 3.8B
#include<iostream>
using namespace std;

main()
{
 int counter;
   for ( counter =1; counter<=10 ; counter++)
      cout<< " Hello ‹C++\n";
return 0;
}
```


* برنامج لاستخدام أكثر من قيمة ابتدائية وأكثر من عداد للزيادة او النقصان

```
// Example 3.9
#include<iostream>
```



```
using namespace std;

int main()
{
for(int i=0,j=0;i<3;i++,j++)
cout<<"i: "<<i<<"  j: "<<j<<endl;
return 0;
}
```


مخرجات البرنامج 3.9 :

```
i: 0  j: 0
i: 1  j: 1
i: 2  j: 2
```

ملاحظة://

من الممكن تعويض القيمة الابتدائية لعداد الحلقة for (أو خارج) الحلقة، مثال

لجمع خمسة أعداد صحيحة :

```
int   I = 1 ،sum = 0 ;
for (   ; I <= 5  ;  ++I  )
  sum + = I ;
```

وممكن اعادة كتابة ذات الخطوات اعلاه بطريقة أخرى :

```
int   I = 1   ، sum = 0 ;
for (   ; I <= 5 ;   )
  sum + = I ++ ;
```

ملاحظة://

لاتستخدم الفارزة المنقوطة بعد الأمر (for)، الأمر (while)، والأمر (do).

ملاحظة://

كما في (if ، else، while) فان الأمر (for) لاينفذ أكثر من أيعاز أو عبارة واحده تاتي بعده مباشرة، لذا كان هناك اكثر من ايعاز يجب ان يكرر ضمن الأمر (for) فيجب أن يحدد بين ({ }) ليكون كتلة.

3.7 استخدام (for) المتداخلة Nested for

ممكن استخدام الأمر (for) بشكل متداخل ولأكثر من مرة وبهذه الحالة فان حلقة (for) تكرركاملة بعدد مرات التكرارالمحددة في (for) الخارجي. فمثلا لو كان لديك عدد من الطلاب في صف معين (30 طالب مثلا) وترغب أن تطبع أسماء الطلبة مع الدرجات التي حصل عليها كل منهم في كل الدروس التي يدرسوها في تلك المرحلة (8 دروس مثلا). هنا يجب طباعة أسماء الطلبة وهي 30 أي أن أمر الطباعة سيكرر 30 مرة لذا استخدم (for) لهذا الغرض لأن عدد مرات التكرار محدد، وفي كل مرة (أي لكل طالب) يجب أن تطبع الدرجات (8 درجات) أي أن أمر طباعة الدرجات يكرر 8 مرات، عليه استخدم (for) أيضا لطباعة الدرجات لكل طالب، وسيكون البرنامج (تم استخدام الحرف الأول للأسم بدل الاسم ليتناسب البرنامج مع ما تم تعلمة في هذا الكتاب لغاية الان) كما يأتي:


```
//// Example 3.10
#include<iostream>
using namespace std;
main()
 {
```

```
int degree ،i ،j;
char name ;
for (i=1 ; i<=30 ; i++)
  {
    cout<<"enter student name and his/her degrees\n";
    cin>>name;
    cout<<name;
    for (j=1;j<=8;j++)
      {
        cout<<"Enter degree:"<<j;
        cin>>degree;
        cout<<degree;
      } //second for
  }    //first for
return 0;
}
```


شرح البرنامج://

في البرنامج أعلاه فأن (for) الأولى تستخدم لطباعة أسماء الطلبة، ولما كان كل طالب له 8 درجات فأن أمر تكرار لهذه الدرجات سيكون من ضمن (for) الأولى (أي عند طباعة أسم طالب معين يجب أن تطبع معه درجاتة الثماني قبل الأنتقال الى الطالب التالي). وبما أن عدد الخطوات المشمولة بالتكرار ضمن (for) الأولى هي أكثر من واحدة لذا تم تحديدها بين ({ }) ونفس الشيء بالنسبة للأمر (for) الثانية. وفي كل مرة تنفذ (for) الأولى ستنفذ (for) الثانية كاملة قبل أن تنتقل الى زيادة العداد (i) (أي أن العداد (j) يبدأ بقيمة البداية ويستمر بالعمل حتى ينتهي بقيمة النهاية هنا 8 بعدها تكون زيادة واحدة للعداد (i)). هذا مشابهة لعقارب الساعة

فلكي يتحرك عقرب الساعات خطوة واحدة فأن عقرب الدقائق يجب أن يتحرك 60 خطوة، وكأنما عقرب الساعات هو (for (i=1; i<=60; i++) وهو حلقة تكرار خارجي وعقرب الدقائق هو حلقة التكرار الداخلي for(j = 1; j<=60; j++).

ملاحظة://

يعمل الأمر (exit ()) على أيقاف تنفيذ (أو الخروج) من البرنامج في مكان محدد من البرنامج وتكون قيمة الدالة (exit(0)) عند الخروج من البرنامج بنجاح ، وألا فأن قيمة الدالة تكون واحد (exit(1)) وهذا يعني أن البرنامج توقف نتيجة حدوث خطأ .

وفي كلتا الحالتين يعود البرنامج الى نظام التشغيل .

ملاحظة://

يستخدم الأمر (break) والأمر (continue) مع حلقات (for) وكافة حلقات التكرار الأخرى مثل (while ,do..while) وكما يلي :

الأمر (break) ويستخدم للسيطرة على تدفق تكرار العبارات وهي تؤدي الى أنهاء أو توقف التكرار عند تحقق شرط معين، مثال


```
for ( i=1 ; i <= 10 ; i++)
{     cin >> x ;
      if  x < 0
         break ;
      else
         cout <<  sqrt ( x ) ;     }
```

في هذه الحالة يتوقف التنفيذ عند ورود عدد سالب لعدم أمكانية أيجاد الجذر التربيعي للعدد السالب

الأمر (continue) ويستخدم أيضا مع حلقات التكرار وهو يعني تجاوز تنفيذ بقية الجمل في التكرار خلال الدوره الحالية والأنتقال الى الدوره التالية (أي أستمر مع حلقة تكرار جديده مع أهمال تنفيذ الأوامر التي بعد الأمر (continue) عند تحقق شرط معين حيث سيعيد المؤشر الى الأمر (for))، مثال

```
for ( i=1 ; i <= 10 ; i++)
{    cin >> x ;
     if  x < 0
        continue ;
     cout << sqrt (x ) ;   }
```


في هذه الحالة عند ورود عدد سالب فأن الأمر (continue) سيمنع متابعة تنفيذ العبارات الأخرى في حلقة التكرار والمتمثلة بأمر الطباعة في هذا المثال ويعيد المؤشر الى الأمر (for) ليبدأ بتكرار جديد .

* برنامج لاستخدام حلقة التكرار (for) مع اهمال عبارتين من عبارات رأس الحلقة، البرنامج يطبع عبارة Looping!

```
// Example 3.11
#include<iostream>
using namespace std;
int main()
{
int counter =0;
for( ;counter<5; )
{
 counter++;
 cout<<"Looping!";
}
cout<<"\nCounter:"<<counter<<".\n";
return 0;
}
```


// مخرجات البرنامج السابق

```
Looping!
Looping!
Looping!
Looping!
Looping!
Counter: 5.
```

* برنـامج يسـتخدم حلقـة التكـرار (for) مـن دون عبـارات رأس البرنـامج، البرنامج يطبع عبارة !Hello.

```
// Example 3.12
#include<iostream>
using namespace std;
int main()
{
int counter = 0;  // initialization
int max;
cout<<"How many hellos? ";
cin>>max;
for ( ; ; )              // for loop that doesn't end
{
if ( counter<max)      // test
{
cout<<"Hello!\n";
counter++;       //    increment
}
else
break;
}
return 0;
}
```


مخرجات البرنامج 3.12 :

```
How many hellos?3
Hello!
```

```
Hello!
Hello!
```

اذا اردت حلقة التكرار (for) لاتعمل شيء فيجب عليك ان تضع فارزة منقوطة بعد عبارة (for) (التنفيذ داخل قوس for) ممكن ان تعتبر مثل هذه الحلقة هي حلقة تأخير الوقت.

* برنامج لاستخدام حلقة التكرار لطباعة الرمز i بحيث يكون امر الطباعة داخل قوس for


```
// Example 3.13
#include<iostream>
using namespace std;
int main()
{
for ( int i=0;i<5;cout<<"i:"<<i++<<endl);

return 0;
}
```


المخرجات:

```
i:0
i:1
i:2
i:3
i:4
```

لاحظ ان هذه الطريقة غير جيدة و الافضل ان تكتبها كما يأتي:

```
for (int i =0 ; i<5 ; i++)
cout<<"i:"<<i<<endl;
```

ملاحظة://

من الممكن أن يكون التداخل بين عبارات التكرار جميعا سواء المتشابهات أو المختلفات، مثلا بين (for ،and for) ،(for ،and while) ،(for ،and do..while) ،(while and do..while) ،(do..while ،and while) ،(while, and while)

3.8 عبارة أختيار الحالة The switch Case Statement

استخدام (if, and if..else) تصبح مضللة بشكل كبير وتزيد التعقيد عند تداخلها وخصوصا التداخل العميق، C++ وفرت البديل وذلك من خلال استخدام (switch) التي تسمح للتفرع لاي عدد من القيم المختلفة لغرض تقييمها على عكس (if) التي تقيم قيمة واحدة.

switch تفحص التعبير وتقارن النتيجة مع كل قيمة من القيم المرافقة للامر (case) وهنا يجب ملاحظة ان المقارنة هي لاغراض المساواة فقط ولايجوز استخدام العلاقات العلائقية او العبارات المنطقية

فاذا تطابقت احدى عبارات (case) مع التعبير فان المسيطرسيقفز الى تلك العبارة المرافقة للأمر (case) ويستمر بالتنفيذ لغاية نهاية كتلة (switch) مالم يتم ايقاف التنفيذ عن طريق الامر (break) اما اذا لم يحدث تطابق مع اي من عبارات (case) فان التنفيذ يتفرع الى عبارة (default) الاختيارية وفي حالة عدم وضع عبارة (default) وعدم حدوث تطابق فان التنفيذ سينتهي دون تنفيذ اي شيء.

ملاحظة://

يفضل استخدام (default) واستخدامها للحالات التي تعتقد انها مستحيلة ويمكن ان تطبع عبارة خطأ

ملاحظة://

اذا لم تضع الامر (break) فان التنفيذ سيستمر للعبارة اللاحقة وهكذا الا اذا كان المبرمج يقصد ذلك وفي هذه الحالة يفضل وضع ملاحظة

في بعض الاحيان تستخدم (if) المتداخلة ولمرات عديدة بشكل ممكن أن يكون مطولا أو مملا، ولتسهيل العمل فأنه يمكن أن تستعيض عنها بعبارة (switch..case)

والصيغة القواعدية لها هو:

```
switch (variable){
case valueOne: statement; break;
case valueTwo: statement; break;
....
case valueN: statement; break;
default: statement;
}
```


ملاحظة://

الأمر (switch..case) دائما يحتاج الى بداية ونهاية ({ })

* استخدام الأمر (switch.. case) لطباعة عبارة معينة مقابلة للرقم المدخل دون انهاء العبارات بالأمر (break).

```
// Example 3.14
#include<iostream>
using namespace std;
int main()
```

```
{
unsigned short int number;
cout<<"Enter a number between 1 and 5: ";
cin>>number;
switch(number)
{
case 0: cout << "Too small ‘sorry!"; break;
case 1: cout << "Good job.\n";    //fall through
case 2: cout<<" Nice Pick!\n";    //fall through
case 3:cout<<"Excellent!\n";     //fall through
case 4:cout<<"Masterful!\n";    //fall through
case 5:cout<<"Incredible!\n"; break;
default:cout<<"Too large!\n"; break;
}
cout <<"\n\n";
return 0;
}
```


مخرجات البرنامج 3.14:

```
Enter a number between 1 and 5:     3
Excellent!
Masterful!
Incredible!

Enter a number between 1 and 5: 8
```

Too large!

ملاحظة://

يأتي بعد الأمر (case) قيمة ثابتة وهذة القيمة من نوع الأعداد الصحيحة أو الحروف فقط ولا يمكن أن نستخدم السلاسل الرمزية والاعداد الحقيقية هنا .

ملاحظة://

يفضل استخدام الأمر (case) في البرامج التي تحتاج الى ثلاثة عبارات (if) متتالية أو أكثر .

لتوضيح الفرق بين استخدام (if) و (case) لاحظ البرنامجين التاليين.

* برنامج يحاكي عمل الحاسبة الجيبية ذات العمليات الأربعة (Calculator) باستخدام if...else

```
// Example 3.15
#include<iostream>
using namespace std;
main()
 {
  int num1 ،num2;
  float Result;
  char ch;
  cout<<" enter two numbers\n";
  cin>> num1>> num2;
  cout<<" enter one of operators + ، - ، * ، / \n ";
```

```
 cin>>ch;
 if (ch =  ' + ')
   Result = num1 + num2 ;
 else
   if (ch = ' – ')
     Result = num1 – num2 ;
   else
     if (ch = ' * ')
       Result = num1 * num2 ;
     else
       Result = num1 / num2   ;
 cout<< result;
return 0;
}
```


البرنامج أعلاه برنامج بسيط اذ يتم أدخال عددين وادخال العملية الرياضية المطلوب أجراؤها عليهم، ثم يقوم المترجم بفحص العملية التي تم أدخالها لينفذ ما مطلوب فيها على الاعداد، وأخيرا تطبع النتيجة.

* برنامج يحاكي عمل الحاسبة الجيبية ولكن باستخدام (switch.. case).

```
// Example 3.16
#include<iostream>
using namespace std;
main()
{
 int num1 ، num2 ;
 char ch ;
```

```
float Result;
cout<< "enter two numbers\n ";
cin>>num1 >> num2;
cout<< "enter one of operators  + ، -  ، * , /     \n" ;
cin>>ch ;
switch ( ch ) {
case  '+ ': result = num1 + num2; break;
case  '– ': result = num1 – num2; break;
case  '* ': result = num1  * num2; break;
case  '/ ': result = num1 / num2; break;
default   : cout<<"not correct character\n";
}
cout<< result ;
return 0;
}
```


ملاحظة://

لا تستخدم (if) بعد (else) عندما يكون هناك احتمال واحد متبقي، وتستخدم بعد (else) أذا كان هناك أكثر من أحتمال واحد ويجب اختيار احدهما .. لأن استخدامها مع وجود أحتمال واحد يعتبر غير منطقي بالرغم من أن البرنامج ممكن أن ينجز العمل.

شرح البرنامج://

البرنامج أعلاه (3.16) أكثر بساطة من البرنامج السابق (3.15).

لاحظ كيف تم استخدام الأمر (switch case) حيث يتم أسناد قيمة

للمتغير (ch)، يتم التحول الى احدى العبارات المتطابقة مع احدى الحالات المحددة بواسطة (case) وكأن العبارة تترجم (أذا كانت قيمة ch تطابق الحالة أعمل الخطوات التي تقابلة) ,حيث ان كل واحدة من عبارات (case) تحمل قيمة من القيم التي ممكن أن تكون عليها (ch) حسب متطلبات البرنامج، وكل (case) توضع في سطر مفرد وتوضع بعدها النقطتين المتعامدتين (:) (colon) بعد ذلك اكتب الاجراء الذي يجب أن يحصل عند تحقق هذه الحالة. فمثلا أذا كانت قيمة المتغير (ch) هي (*) فأن المترجم يفحص أولا (+) سوف يجدها لا تساوي قيمة المتغير (ch) أي أن الأجابة هي خطأ (false) فيتركها ليقارن القيمة اللاحقة وهي (–) وأيضا سيجد أن الأجابة (false) فيستمر بفحص القيمة التي بعدها وهي (*) هنا ستكون النتيجة (true)، لذا سينفذ العبارة أو العبارات التي بعدها وهي أجراء عملية الضرب ووضع النتيجة بالمتغير (result)، ونظرا لوجود الأمر (break) فان تنفيذ هذه الحالة سيتوقف عند الأمر (break) وينتقل المسيطر الى نهاية الأمر (switch)، ليأتي بعدها أمر طباعة النتيجة.

لاحظ هنا استخدام الامر (break) لمنع استمرار التنفيذ الى عبارات (case) الأخرى.

ملاحظة://

دائما الحروف والسلاسل الرمزية عند استخدامها وكتابتها في البرامج على أساس أنها حروف أو سلاسل حرفية وليس لغرض أخر فأنها تحدد بين علامتي أقتباس .

3.9 أمثلة محلولة

* برنامج لأيجاد الرقم الأكبر بين رقمين.

```
// Example 3.17
#include<iostream>
```

```
using namespace std;
main()
{
  int x, y;
  cout<<"Enter two numbers\n";
  cin >> x >> y ;
  if (x > y)
    cout <<"the largest number =" << x ;
  else
    cout <<"the largest number =" << y ;
return 0;
}
```

* برنامج لأيجاد قيمة (z) حيث أن

$$Z = \begin{cases} 5x^2 + 3x/y & \text{when } x = y \\ Y^2 - 3x & \text{when } y > x \end{cases}$$

```
// Example 3.18
#include<iostream>
using namespace std;
main()
{
    int x , y;    float z;
cout <<"Enter x and y ";
```

```
cin >> x ;    cin >> y ;
 if (x == y)
   Z =5*x*x + 3* x / y ;
 else
   Z =y*y - 3*x;
   cout << z ;
return 0;
}
```


* برنامج لطباعة الأرقام الفردية المحددة بالرقمين (55 – 35).

```
/ Example 3.19
#include<iostream>
using namespace std;
main()
{
 int i ;
 for ( i=35 ; i<=55 ; i++ )
   if ( i %2 == 0)
     continue;
     cout << i ;
return 0;
 }
```


* برنامج لأيجاد مجموع الأرقام الزوجية المحددة بين الرقمين (100– 2).

```
// Example 3.20
#include<iostream>
```

```
using namespace std;
main()
{
 int i ، sum;
 sum =0;
 for (i =2 ; i<=100 ; i++)
   if (i %2==0)
     sum += i ;
 cout << sum ;
return 0;
}
```


* برنامج لأيجاد أكبر وأصغر عدد من بين (15) عدد.

```
// Example 3.21
#include<iostream>
using namespace std;
main()
{
 int x,max,min;
 cout << "Enter first number\n";
 cin >> x ;
  max =x;        min =x;
 for (i =1 ; i<=14 ; i++ )
   {
     cin >> x ;
     if(x>max)
```

```
      max =x;
    else
      if(x<min)
        min =x;
  }
 cout << "max number="<< max;
 cout << "min number=" << min;
return 0;
}
```


* برنامج لأيجاد مجموع عدد من الأرقام التي تقبل القسمة على (7)، وأخر رقم فيها يساوي (0).

```
// Example 3.22
#include<iostream>
using namespace std;
main()
{
 int  Sum = 0 ، x;
 do
   cout <<"Enter new number";
   cin >> x;
   if ( x % 7 == 0 )
   Sum =Sum+x;
 while(x!=0)
 cout << Sum;
return 0;
```

```
}
```

*برنامج لأيجاد معدل مجموعة من الأرقام أخر رقم فيها هو(12).

```
// Example 3.23
#include<iostream>
using namespace std;

main()
{
 int sum=0 ، x ، count=0 ;
 cout <<"Enter first number in group\n ";
 cin >> x ;
 while(x!=12)
  {
   sum =sum+x;
   ++count ;
   cin >> x;
  }
 cout << sum/count ;
return 0;
}
```


* برنامج لطباعة الأرقام (0..9) و (0..9) بعمودين منفصلين ومتجاورين

```
// Example 3.24
# #include<iostream>
using namespace std;
 main( )
```

```
{
        for ( int i=0 ، j=9 ; i<=9 ; i++ ،j--)
        cout <<"i=" << i << '\t' << "j=" << j << endl ;
        return 0;
 }
```

* برنامج لأيجاد ناتج (n) من العناصر في العلاقة التالية:

2/1*2/3*4/3*4/5*6/5*6/7……


```
// Example 3.25
#include<iostream>
using namespace std;
main()
{
 int i ، n ; float sum = 1.0 ;
 cout <<"Enter number of elements\n ";
 cin >> n ;
  for (i =1 ; i<=n; i++ )
    {
      if (i %2==0)
        sum =sum * i/ (i+1);
      else
        sum =sum * (i+1)/i;
    }
  cout << sum ;
return 0;
}
```


*برنامج لأيجاد ناتج العمليات بين ناتج

```
// Example 3.26
#include<iostream>
using namespace std;
main()
{
  int x, y, z ;
  cout <<"Enter three numbers\n";
  cin << x << y << z ;
  if (x<y) && (x<z)
    cout <<"min number=\n"<< x;
  else
    if (y<x) && (y<z)
      cout <<"min number=\n'<<y ;
    else
      cout <<"min number=" << z ;
return 0;
}
```

* برنامج لطباعة جدول الضرب للأرقام المحددة (1.. 10)

```
// Example 3.27
#include < iosream>
using namespace std;
main() {
 int i ، j ;
 for ( i =1 ; i <= 10 ; i++ )  {
 for ( j =1 ; j <= 10 ; j++ )
```

```
 cout << i * j << '\t' ;
 cout << endl ;
}
return 0;
}
```

* برنامج لقراءة عدد ثم أوجد مجموع أرقامة والرقم الأكبر بين أرقامة.

(مثلا العدد 5472 فأن مجموع أرقامة هي (18) والرقم الأكبر فيه هو (7))


```
// Example 3.28
#include<iostream>
using namespace std;
main()
{
int  x, z ، max =0  ، sum=0 ;
  cout << "Enter number";
  cin >> x ;
  do
    z = x  % 10;
    sum = sum + z ;
    if (z > max)
      max = z ;
    x =x / 10;
  while(x!=0);
  cout << "max number=\n"<< max ;
  cout << "sum of number digits\n" << sum ;
return 0;    }
```

* برنـــامج لتحويـــل الـــرقم العشـــري (decimal number) الـــى ثنـــائي (binary number) دون استخدام الدوال الجاهزه.

```
// Example 3.29
#include<iostream>
using namespace std;
main()
{
  int sum=0 ، i=1 ، x ، b;
  cout << "Enter decimal number\n" ;
  cin << x ;
  while (x != 0)
    {
      b =x %2;
      sum: =sum+ i*b;
      x =x / 2;
      i =i*10;
    }
  cout << sum ;
return 0;
}
```


* برنامج لأيجاد عدد القيم الموجبة في مجموعة من القيم تنتهي بالرقم (0)

```
// Example 3.30
#include < iostream>
```

```
using namespace std;
main () {
int counter =0;
do
{
cin >> x ;
if ( x > = 0 )
  counter ++;
}
while ( x ! = 0 ) ;
cout << " Number of positive numbers in set = \n " << counter ;
return 0;
}
```


برنامج لطباعة مايأتي:

0 3 6 9 … n

3 6 9 … n

6 9 … n

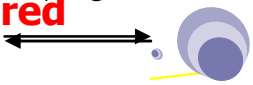
9 … n


```
// Example 3.31
#include<iostream>
using namespace std;
main()
{
 int n, x=0;
```

```
cout <<"Enter the last number n\n" ;
 cin>> n ;
 if (n %3 == 0)
   {
     while(x<=n)
       {
        for ( int i=0; i<=n; i+3){
         if ( i <  x)
          cout<<"  ";
         else
          cout<< i ;
         } //  end for
         cout << endl ;
         x += 3;
       } //while
   }  // end if
   else
     cout << "Error number, n should divided by 3 \n";
return 0;
}
```


* برنامج لأدخال قيمة أقل من (10) أو اكبر من (100) وطباعتها

```
// Example 3.32
#include <iostream>
using namespace std;
 int main()
```



```
{
int x;
cout << "Enter a number less than 10 or greater than 100: ";
cin >> x;
cout << "\n";
if (x > 100)
cout << "More than 100 ،Thanks!\n";
else
if(x < 10)
cout << "Less than 10 ،Thanks!\n";
return 0;
}
```


* برنامج لأيجاد جذور معادلة من الدرجة الثانية باستخدام الدستور

```
// Example 3.33
#include<iostream>
#include<math>
using namespace std;


main(){
int a ، b ، c ، x1 ، x2;
cin >> a >> b >> c ;
int z =   b*b– 4 *a * c   ;
if  ( z < 0 )
cout << " Error  ،square root with negative value \n ";
else
{
```

```
 Z = sqrt ( z ) ;
x1 = (  b + Z ) / 2 * a  ;
x2 = ( b – Z ) / 2 * a ;
cout << " first root = " << x1 << endl ;
cout << " the second root = " x2 << endl ;
  }
return 0;
}
```


* برنامج لتنفيذ لعبة والتي تتضمن ادخال رقمين من قبل المستخدم احدهما كبير والأخر صغير، الرقم الصغير يتم زيادتة بمقدار واحد والرقم الكبير يتم انقاصة بمقدار اثنين، هدف اللعبة هو تخمين متى يلتقي الرقمان

```
// Example 3.34
#include <iostream>
using namespace std;
 int main()
 {
 unsigned short small;
 unsigned long large;
 const unsigned short MAXSMALL=65535;
 cout << "Enter a small number: ";
 cin >> small;
 cout << "Enter a large number: ";
 cin >> large;
 cout << "small "<<small<<"..";
```

```
// for each iteration ، test three conditions
while (small < large && large > 0 && small < MAXSMALL)
{
if (small %5000 == 0) // write a dot every 5k lines
cout << ".";
small++;
large - =2;
}
cout << "\nSmall: " << small << " Large: " << large << endl;
return 0;
}
```


شرط السيطرة على التكرار من الممكن ان يكون اي عبارة C++ مقبولة، هو ليس بحاجة الى متغير سيطرة على التكرار. في المثال التالي، فان التكرار سيستمر بالتنفيذ لغاية ان يضغط المستخدم احد مفاتيح لوحة المفاتيح. المثال يقدم ايضا دالة مكتبية مهمة هي (kbhit()) هذه الدالة تعيد القيمة المنطقية خطأ (false) اذا لم يتم ضغط اي مفتاح وتعيد القيمة المنطقية صح (true) اذا ما تم ضغط مفتاح ما. هي لاتنتظر الضغط على المفتاح، وبذلك ستسمح لدائرة التكرار بالاستمرار في التنفيذ. ان الدالة (kbhit()) غير معرفة بلغة C++ القياسية، ولكن هناك امتداد عام يوفر بواسطة غالبية المترجمات. وهذه الدالة تستخدم مع الملف الرأسي (conio).

* برنامج للاستمرار بطباعة اعداد لحين الضغط على زر من ازرار لوحة المفاتيح.

```
// Example 3.35
#include <iostream>
#include <conio>
using namespace std;
```

```
int main()
{
int i;
// print numbers until a key is pressed
for(i=0; !kbhit(); i++)
cout << i << ' ';
return 0;
}
```


في كل مرة خلال عملية التكرار، فان الدالة (kbhit()) سوف تستدعى. اذا ما تم ضغط مفتاح فان القيمة المنطقية صح ستعاد والتي ستجعل الدالة (!kbhit()) ستكون خطأ، وبذلك فان التكرار سيتوقف.

* برنامج لعمل لعبة تعتمد على تخمين رقم ومقارنتة بالرقم الذي يولدة الحاسوب.

```
// Example 3.36
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
int magic; // magic number
int guess; // user's guess
magic = rand(1000); // get a random number
cout << "Enter your guess: ";
cin >> guess;
```

```
if (guess == magic) {
cout << "** Right **\n";
cout << magic << " is the magic number.\n";
}
else {
cout << "...Sorry ، you're wrong.";
// use a nested if statement
if(guess > magic)
cout <<" Your guess is too high.\n";
else
cout << " Your guess is too low.\n";
}
return 0;
}
```


* برنامج لايجاد اكبر رقم بين خمسة ارقام (باستخدام عبارة if)

```
// Example 3.37
#include<iostream>
using namespace std ;
main(){
int  a ،b ،c ،d ،e ،h ;
cout<<"Enter five numbers\n";

cin>> a>> b>> c>> d>> e ;
int max=a;
if (max < b)
```

```
max=b ;
if (max < c)
max=c ;
if (max < d)
max = d;
if (max < e)
max = e;
cout << "Maximum number is=" << max << "\n\n" ;
return 0;
}
```


* برنامج لطباعة الشكل التالي

```
*******
*****
***
*
```

```
\\ Example 3.38
#include<iostream>
using namespace std ;

main( ) {
int I ،j ،k ;
for ( i=4 ; i >=1 ; i-- )
for ( j =1 ; j <= 4-i ; j++ )
cout << “  “ ;
for ( k = 1 ; k <= 2 * i-1 ; k++ )
cout << “*” ;
cout << “\n” ;
}
return 0 ;
}
```


* برنامج لطباعة الشكل التالي

```
:0
#:1
**:2
###:3
****:4
#####:5
******:6
########:7
**********:8
#########:9
**********:10
```

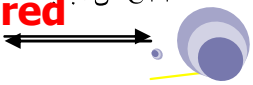
```
// Example 3.39
#include < iostream>
using namespace std ;
```

```
main ( ) {
int x ,y ;
for ( x= 0 ; x <= 10 ; x++ )
{
for ( y = 0 ; y <= x ; y++ )
{
if  ( x %2 == 0 )
cout << "#" ;
else
```

```
cout <<"*" ;
}
cout << ":" << x << endl ;
 }
return 0 ;
    }
```

* برنامج لحل المعادلة $y = x^2 + x + 1/x$ حيث ان قيمة x تتغير بين (2..4) مع زيادة مقدارها 0.2 في كل خطوة، استخدم حلقة التكرار for.

```
// Example 3.40
```



```
#include<iostream>
#include<iomanip>
using namespace std ;

main ( )  {
double x ,y ;
cout << "x value " << "     " << "f(x) \n\n" << setiosflags(ios::fixed) ;
for( int I = 20 ; I <= 40 ; I+=2 )
{
x = I / 10.0 ;
y = x*x + x + 1/x ;
cout << setw(7) << setprecision(1) << x << setw(16) << setprecision
(10) << y << "\n" ;
}
cout << "\n\n" ;
return 0 ;
}
```


* برنامج لطباعة الشكل التالي:

```
1
121
12321
1234321
123454321
12345654321
```

```
// Example 3.41
#include<iostream>
```

```
#include<iomanip>
using namespace std ;
main( )  {
int n = 6 , k, i, j ;
for (i = 1 ; i < n ; ++i )
{
k = 0 ;
cout << endl << setw(n-i+1) ;
for (j = 1 ; j <= (2*(i-1)+1) ; ++j)
{
if ( j<= i)
{   ++k ;
cout << k ;
}
else
{
--k ;
cout << k ;
}}}
cout << "\n\n" ;
return 0 ;
}
```


اسئلة للحل:

1. اكتب برنامج لقراءة ثلاثة ارقام ومن ثم ايجاد الرقم ذات القيمة الصغرى.

2. اكتب برنامج لقراءة ثلاث ارقام ومن ثم ايجاد الرقم ذات القيمةالوسطى بينهما.

3. اكتب برنامج لايجاد قيمة (R) حيث ان:

$R = \sqrt{a^2 + b^2}$

$a = x/ (y^2 + z^2)$

$b = (x+y) /z$

4. اكتب برنامج لايجاد قيمة Z حيث ان

$Z = 5A^2 + 3B / A$ when A>= B

$Z = B^2 – 3A$ when B>A

5. اكتب برنامج لايجاد قيمة X حيث ان (استخدم طريقة (switch-case))

$X = A + B$ if j = 1

$X = A – b$ if j = 2

$X = A * B$ if j = 3

$X = A / B$ if j = 4

6. اكتب برنامج لقراءة عدد واحسب عدد الارقام التي يتكون منها.

7. اكتب برنامج لقراءة عدد واحسب الارقام التي اكبر من 3.

8. اكتب برنامج لقراءة عدد وجد اكبر رقم فيه.

9. اكتب برنامج لايجاد قيمة D حيث ان

$D = 3x^3 – 2x + 5$ if x> 5

$D = \sqrt{x - 9}$ if $5 \geq x \geq 2$

$D = |x - 7|$ if x< 2

10. اكتب برنامج لاختبار عدد فردي فيما اذا كان يقبل القسمة على 3, 5 بذات الوقت.

11. اكتب برنامج لايجاد مضروب (factorial) اي رقم .

12. اكتب برنامج لايجاد اكبر واصغر رقم في مجموعة ارقام تتكون من 200 رقم.

13. اكتب برنامج لايجاد معدل الارقام الزوجية ضمن مجموعة ارقام اخر رقم فيها (-1) .

14. اكتب برنامج لطباعة الشكل التالي

0 3 6 9 12 15

3 6 9 12 15

6 9 12 15

9 12 15

12 15

15

15. اكتب برنامج لايجاد قيمة Y للعلاقات التالية:

3.

a) $Y = 1 + 2 + 3 + \ldots + N$

b) $Y = 1 + 2^2 + 3^3 + \ldots + N^n$

c) $Y = X + X/2 + X/3 + \ldots + X/n$

d) $Y = x + x^2/2 + x^3/3 + \ldots + x^n/n$

e) $Y = x + x^2/2! + x^3/3! + \ldots + x^n/n!$

f) $Y = 1 + 3^3 + 5^5 + \ldots + n^n$

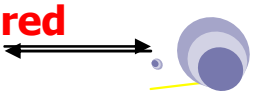
g) $Y = 1/3 - 1/5 + 1/7 - \ldots + 1/33$

h) $Y = 3/70 + 6/63 + 9/56 + \ldots + 30/7$

i) $Y = (x^3 - 1)/2! + (x^6 - 4)/5! + (x^9 - 7)/8! + \ldots + (x^{18} - 16)/17!$

j) $Y = 2/4 + 4/6 + 6/8 + 8/10 + \ldots + 18/20$

k) $Y = (15 + 7 + 13 + \ldots + 121) - (110 + 104 + 98 + \ldots + 8)$



l) $Y = (x-1/x) + ½ (x-1/x)^2 + 1/3 (x-1/x)^3 +1/4(x-1/x)^4 + \dots + 1/12(x-1/x)^{12}$

m) $Y = 2!/3! + 3!/4!+ 4!/5!+\dots+ 60!/61!$

16. اكتب برنامج لايجاد قيمة K من العلاقة التالية:

$$K = \sum_{N=1}^{10} N! + \prod_{i=1}^{10} xi$$

17. اكتب برنامج لقراءة مجموعة من الاعداد الصحيحة اخر عدد فيها 55, واحسب مايلي:

a) معدل قيم هذه الاعداد.

b) حاصل ضربها.

c) اكبر قيمة بينها.

d) عدد الاعداد التي تقبل القسمة على 3 ولاتقبل القسمة على 7.

e) عدد الاعداد الزوجية.

f) تسلسل العدد الاصغر في المجموعة.

chapter 5

FUNCTIONS IN C++

الدوال
FUNCTIONS

4.1 المقدمة

تقسيم البرنامج الى دوال هي احدى المباديء الرئيسية للبرامج المهيكلة باتباع اسلوب من الاعلى الى الاسفل (Top Down)، وهي مفيدة نظرا لأمكانية استدعائها واستخدامها في اماكن مختلفة في البرنامج.

4.2 الدوال

الدوال هي واحدة من كتل البناء الاساسية في لغة C++، فهي مجموعة من الخطوات (الايعازات) تحت أسم واحد..والدالة تسمح لك بخلق مجاميع منطقية من الشفرات، فهي جزء من برنامج يعمل على البيانات ويعيد قيمة، وكل دالة لها اسمها الخاص وعندما يتم تمييز الاسم في البرنامج اثناء التنفيذ فان البرنامج سيولد تفرع الى الدالة التي تحمل هذا الاسم ليقوم بتنفيذها، وبعد الانتهاء يعود المسيطر الى ذات المكان الذي تفرع منه في البرنامج لاكمال تنفيذ باقي الايعازات.

4.2.1 فوائد استخدام الدوال:

* تساعد الدوال المخزنة في ذاكرة الحاسوب أو التي يكتبها المبرمج على تلافي عملية التكرار في خطوات البرنامج التي تتطلب عملا مشابها لعمل تلك الدوال.
* تساعد الدوال الجاهزة على تسهيل عملية البرمجة نفسها.
* من شأن أستعمال الدوال التوفير في المساحات المطلوبة في الذاكرة.
* ومن شأنها أيضا أختصار زمن البرمجة وزمن تنفيذ البرنامج.
* أمكانية استخدام الدوال مع برامج أخرى تتطلب تنفيذ أو انجاز ذات

المهمة.

* عندما يكون برنامج C++ مكون من أجزاء (دوال) مستقلة واضحة المعالم، فأن البرنامج نفسة يكون واضحا لكل من المبرمج والقاريء والمستخدم على حد سواء.

4.2.2 تعريف الدالة

تتكون الدالة من رأس وجسم، والدالة تأخذ الصيغة او الشكل العام التالي

type *function-name* (argument-list)

{ // codes to execute inside function }

وهذا يسمى **تعريف** الدالة اي الدالة التي تحتوي على شفرة البرنامج اللازمة لانجاز عمل معين اضافة الى راس الدالة. بينما **الاعلان** عن الدالة هو كتابة راس الدالة فقط.

وكما هو واضح يتكون رأس الدالة من ثلاث اجزاء هي:

1. النوع (type)
2. اسم الدالة (function-name)
3. قائمة المتغيرات (argument-list)

ولتوضيح هذه النقاط سنبدأ من النوع، والنوع هو اي نوع من الانواع المعروفة في لغة C++ مثل (int, float, char…etc) ودائما عند استخدام الدوال يجب ان يحدد النوع للدالة وهذا النوع يمثل نوع القيم التي ستعاد بواسطة الدالة (كل دالة تعيد قيمة تمثل نتيجة معالجة الايعازات في الدالة)، وفي حالة عدم اعادة اي قيمة من الدالة بعد انتهاء تنفيذ الدالة عندها سيكون النوع (void) وهو يعني لاشيء، وكما تعلم ان النوع يسبق المتغيرات وهو يمثل عنوان المساحة الخزنية التي يجب ان تخصص في الذاكرة لقيم هذا المتغير, والنوع (void) يعني عدم حجز اي مكان للمتغير في الذاكرة.

اما الجزء الثاني فهو اسم المتغير وهو الاسم الذي تستدعى به الدالة وبما ان

اسم الدالة مسبوق بنوع فهذا يعني ان اسم الدالة هو معرف او متغير ولذلك فان هذا المعرف سيحتاج الى قيمة وفقا للقواعد المعروفة لك حول التعامل مع المتغيرات والتي تتضمن الاعلان عن المتغير واسناد قيمة له، اما الاعلان عن المتغير فهو النوع السابق له (السابق لاسم الدالة)، اما قيمة هذا المتغير (اسم الدالة) فسيتم اسنادها له من القيمة المعادة من تنفيذ الدالة.

الجزء الاخير من راس الدالة هو الوسائط التي تمرر الى الدالة، وهو عبارة عن المدخلات الى الدالة (القيم التي ترسل الى الدالة من خلاله الدالة لغرض معالجتها في الدالة)، هذه المدخلات ممكن ان تكون وسيطا واحدا، اكثر من وسيط، وممكن ان لايكون هناك اي وسيط وعندها تكون الاقواس اما خالية، او نضع فيها كلمة (void).. والوسائط هي متغيرات تكتب اسماؤها في داخل القوسين كل منها يكون مسبوقا بنوعة كما سترى لاحقا.

الجزء الثاني من الدالة هو جسم الدالة، وهو يحتوي على الايعازات او الشفرة اللازمة لانجاز العمل الذي من اجلة كتبت الدالة، وتكون هذه الشفرة محددة بين القوسين المتوسطين واللتان تمثلان البداية والنهاية للبرنامج، وممكن ان تكون هذه الشفرة ايعازا واحدا او اكثر.

كل دالة يجب ان تحتوي على الاقل على زوج من هذه الاقواس (الأقواس المتوسطة) على الاقل تبدأ بالقوس المفتوح ({) وتنتهي بالقوس المغلق (}).

وعادة ينتهي التنفيذ باعادة قيمة بواسطة عبارة الاعادة (return)، اذ ستعيد او تسند القيمة الناتجة من تنفيذ الدالة الى اسم الدالة وهي تمثل المخرجات للدالة.

مثال:

```
float volume (int x ،float y ،float z)
```

لاحظ هنا ان كل وسيط يتم الاعلان عن نوعة بشكل منفصل ولايجوز الدمج فمثلا الاعلان التالي يعتبر غير صحيح

```
float (int x ،float y ،z)   // illegal
```

ملاحظة://

في الأعلان عن الدوال فان أسماء الوسائط اختياري لانها لاتمثل الأسماء الحقيقية، لذلك يمكن حذف هذه الاسماء والاكتفاء بنوعها فقط مثل

float volume (int ،float ،float) ;

بينما في تعريف الدالة فان الاسماء ضرورية لامكانية الاشارة لها او استخدامها داخل الدالة، مثل

```
 float volume ( int a ،float b ،float c )
{
float v = a * b * c ;
return v;
}
```


* برنامج لاستخدام دالة لطباعة عبارة معينة، يوضح استخدام النوع (void)

```
// Example 4.1
#include <iostream>
using namespace std;
void printmessage ()
{
  cout << "I'm a function!";
}
int main ()
{
  printmessage ();
  return 0;
}
```

4.3 الدالة الرئيسة Main Function

الدالة main () هي الدالة الرئيسة لاي برنامج (كل برنامج بلغة C++ يجب ان يحتوي على الدالة (main()، وعند تنفيذ البرنامج فان اول عبارة يتم تنفيذها هي العبارة الاولى في الدالة الرئيسة (main()، وعادة الدالة الرئيسة تعيد قيمة من نوع الاعداد الصحيحة الى نظام التشغيل، هذه القيمة تكون صفر عند اكمال تنفيذ البرنامج بشكل صحيح وتكون اي قيمة اخرى عند حدوث خطأ لذلك عند كتابة هذه الدالة فأن أخر عبارة هي (return 0) للدلالة على اكمال التنفيذ دون أخطاء، اما اذا لم تكتب عبارة الاعادة فان المترجم سيصدر رسالة تحذير ويستمر بالتنفيذ وفي بعض الاصدارات يصدر رسالة خطأ ولايتم التنفيذ، لغة C++ تعرف هذه الدالة على انها من نوع الاعداد الصحيحة.. لذلك فان نظام التشغيل يعامل الدالة main () على انها من نوع الاعداد الصحيحة بالأفتراض في حالة عدم كتابة النوع لهذه الدالة (في حالة الرغبة ان تكون الاعادة من نوع أخر فيجب ان يشار لها..أي يكتب النوع مقابل اسم الدالة (main ()). البرامج في لغة C++ ممكن ان تتكون من عدد من الاصناف، الدوال، وعناصر البرنامج الأخرى ولكن عند بداية التنفيذ للبرنامج فان المسيطر دائما يذهب الى الدالة الرئيسة (main()). وفي حالة عدم وجود دالة في البرنامج باسم (main() فان المترجم سيصدر رسالة خطأ

الاقواس التي تتبع اسم الدالة هي صفة مميزة للدالة وبدون هذه الاقواس فان المترجم ممكن ان يعتقد ان الايعاز (main ()) ممكن ان يشير الى متغير او عنصر اخر في البرنامج.

ملاحظة://

لاينتهي رأس الدالة بفارزة منقوطة، كما هو معمول مع عبارة (for)، الا في حالة الاعلان عن الدالة لأسباب سنشير لها في موضعها فأنها تنتهي بفارزة منقوطة لأننا عند الأعلان لا نكتب الا رأس الدالة اما تعريف الدالة فسيكون في مكان اخر من البرنامج.

4.4 اعادة القيم Return Values

في كل مرة يتم استدعاء الدالة فان هناك نتائج او مخرجات يجب ان تخرج نتيجة تنفيذ الدالة، فكل الدوال عدا تلك من نوع (void) تعيد قيم، هذه القيم تحدد بواسطة عبارة الارجاع (return).

في C++ فان اي دالة يجب ان تحتوي عبارة الأرجاع التي يجب ان تعيد قيمة، عدا طبعا تلك من نوع (void)، كذلك فان الدالة ممكن ان تعيد مؤشرات والتي سنوضحها في الفصل الخاص بالمؤشرات.

ملاحظة://

اذا ما تم تنفيذ عبارة الأعادة (return)، فانها ستكون اخر عبارة تنفذ في تلك الدالة ولا تنفذ اي عبارة بعدها وبذلك ينتهي تنفيذ الدالة.

ملاحظة://

اعادة قيمة في الدالة ممكن ان تتم باستخدام عبارة الاعادة كما بينا، او من الممكن ان تستخدم طريقة الاسناد وذلك باسناد قيمة الى اسم الدالة داخل جسم الدالة، في ادناه امثلة لاعادة قيم :


```
int square ( int x ،int y )
        {   int s = x * y ;
            return s;      }


int square ( int x ،int y )
        {    return ( x * y ) }


int square ( int x ،int y )
        {
          s = x * y ;
```

```
    square = s ;
  }
```

ملاحظة://

من الممكن استخدام اكثر من عبارة اعادة في الدالة الواحدة ولكن واحدة منها سوف تنفذ وتنهي البرنامج والاخرى او الأخريات سوف تهمل مثال :


```
int max (int x, int y)
 {
    if (x > y)
      return x ;
    else
      return y ;
 }
```

ملاحظة://


```
return (x > 5);
```

ستعيد صح او خطا وليس قيمة المتغير (x)، فاذا كانت العبارة صح ستعيد القيمة (1) اما اذا كانت خطأ ستعيد القيمة (0) .

جملة (return) لها وظيفتان:

1. تعد مخرجا طبيعيا في نهاية الدالة، وتعيد نتيجة الدالة الى العبارة التي استدعت الدالة في البرنامج.
2. تستعمل لعمليات حساب واستخراج قيم تعابير بداخلها.

4.5 اين تكتب الدالة في البرنامج:

من المعلوم ان البرنامج يتكون على الاقل من دالة واحدة رئيسة هي دالة (main()) وممكن ان تكتب دوال اخرى في البرنامج فضلا عن الدالة الرئيسة، السؤال هنا اين تكتب الدوال الاخرى؟ قبل الدالة الرئيسة ام بعدها.. واقع الحال يمكنك كتابة الدالة (تعريف الدالة) قبل او بعد الدالة الرئيسة. عند كتابة الدالة او الدوال قبل الدالة الرئيسة فيتم بعد كتابة الموجهات وبالامكان استدعاء هذه الدوال من داخل الدالة الرئيسة او اي دالة اخرى بعدها وفقا لطريقة الاستدعاء التي سنبينها لاحقا. اما اذا ما تم كتابة تعريف الدالة بعد الدالة الرئيسة فهنا نحتاج الى الاعلان عن الدالة قبل ان يتم استدعائها داخل الدالة الرئيسة كما هو الحال مع المتغيرات، ويتم الاعلان عن طريق كتابة رأس الدالة فقط منتهية بفارزة منقوطة (الأعلان عن الدالة) بعد الموجة الرأسي، بعدها يمكن ان يتم استدعائها.

* برنامج لاستخراج مربع عدد باستخدام دالة تكتب بعد الدالة الرئيسة

```
// Example 4.2
#include <iostream>
using namespace std;
int square (int i) ;
main() {
int i=10;
cout<<"\n"<<(square(i))<<" is the quare value of "<<i<<endl;
return 0;
}
int square(int i) {
i *=i;
 return i;
}
```

في المثال (4.2) تم الأعلان عن الدالة اولا وانتهت بفارزة منقوطة (لأنه أعلان فقط)، وهذا الأعلان ضروري بسبب انك كتبت الدالة بعد الدالة الرئيسة ولذلك عند استدعاء الدالة (square) من داخل الدالة الرئيسة فان المترجم سينظر للخطوات السابقة ابتداء من الموجهات واذا لم يجد الدالة فانه سيصدر رسالة خطأ، لذا لابد من أعلام المترجم بوجود دالة من خلال الاعلان عنها حيث سيؤدي هذا الاعلان الى البحث عن الدالة قبل وبعد الدالة الرئيسة.

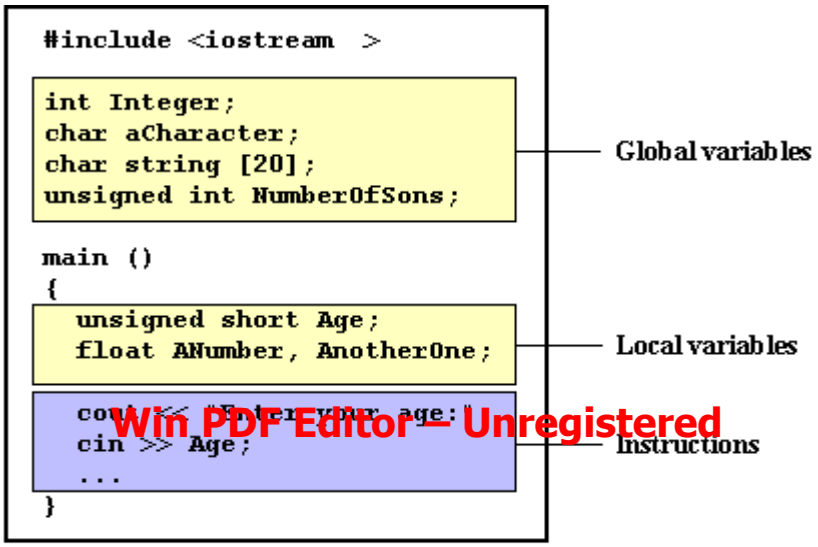
ملاحظة://

دائما يجب تعريف الدالة قبل ان يتم استدعائها من قبل اي دالة اخرى تكتب بعدها.

4.6 المتغيرات

ليس بالامكان تمرير متغيرات الى الدالة فقط ولكن بالامكان الاعلان عن متغيرات داخل جسم الدالة ايضا، وهذا يتم باستخدام المتغيرات المحلية (Local variables) وسميت كذلك لتواجدها محليا في الدالة نفسها فقط، اذ ان هذه المتغيرات سوف لاتستمر فعاليتها بعد انتهاء تنفيذ الدالة (اي بعد اعادة القيمة من الدالة). المتغيرات المحلية تعرف مثل المتغيرات الاخرى.. كذلك فان الوسائط التي تمرر الى الدالة تعد متغيرات محلية وبالامكان استخدامها بالضبط كما لو كانت معرفة داخل جسم الدالة.

اما المتغيرات التي تعرف خارج جميع الدوال فلها تاثير عام على كامل البرنامج بكل دوالة وتسمى المتغيرات العامة (global variables) وبالرغم من كون المتغيرات العامة متغيرات مقبولة في C++ لكنها غالبا لاتستخدم.. وبشكل عام فهي ضرورية وخطرة، ضرورية لان المبرمج احيانا يحتاج الى بيانات تكون متوفرة لعدد من الدوال ولايرغب ان يمررها كوسائط من دالة لاخرى.. وهي خطرة لانها بيانات مشتركة وبامكان دالة ان تغير المتغير العام بطريقة ربما تكون غير مرئية من الدوال الاخرى، وهذا يولد خطأ ربما يكون من الصعب ايجاده



شكل (4.1): يوضح مدى عمل المتغيرات المحلية والعامة

كما واضح من الشكل (1.4) فان المتغيرات العامة تكتب بعد الموجهات وقبل الدالة الرئيسية.

- برنامج لتحويل درجات الحرارة من الفهرنهايت الى المئوي، باستخدام الدوال


```
// Example 4.3
#include <iostream>
using namespace std;
 float Convert (float);
 int main()
 {
 float TempFer;
 float TempCel;
 cout << "Please enter the temperature in Fahrenheit: ";
```

```
cin >> TempFer;
TempCel = Convert(TempFer);
cout << "\nHere's the temperature in Celsius: ";
cout << TempCel << endl;
return 0;
}
float Convert (float TempFer)
{
  float TempCel;
  TempCel = (TempFer - 32) * 5 / 9;
  return (TempCel);
}
```


مخرجات البرنامج4.3://

```
Please enter the temperature in Fahrenheit: 212
Here's the temperature in Celsius: 100
```


في هذا البرنامج فان متغيرين من نوع الاعداد الحقيقية استخدما، احدهما لدرجة الحرارة مقاسة بالفهرنهايت والثاني لدرجة الحرارة مقاسة بالمئوي، يتم ادخال درجة الحرارة بالفهرنهايت لترسل كوسيط ضمن عبارة الاستدعاء التي تستدعي الدالة (convert)، المسيطر سيقفز الى الدالة المستدعاة ليتم تنفيذ عباراتها ثم العودة الى البرنامج الرئيس، لاحظ هنا ان المتغير (Tempcel) هو متغير موقعي او محلي ضمن الدالة (convert) كذلك فان المتغير (Tempfer) هو متغير محلي ضمن الدالة الرئيسة وسترسل الى دالة التحويل لتحل بالمتغير (Tempcel)

ملاحظة://

المتغيرات المحلية التي لها نفس الاسم للمتغيرات العامة سوف لاتغير المتغيرات العامة، فاذا كانت هناك دالة فيها متغيرات لها نفس اسم المتغيرات العامة فان الاسم يشير الى متغيرات محلية وليس العامة عند استخدامها داخل الدالة، وهي تمثل المتغيرات التي يعلن عنها داخل جسم الدالة اي بين القوسين المتوسطين.

* برنامج يوضح مدى عمل المتغيرات المحلية والعامة وذلك بطباعة المتغيرات العامة والمحلية التي لها نفس التسمية.

```
// Example 4.4
#include <iostream>
using namespace std;
void myFunction(); // prototype
int x = 5  ,y = 7; // global variables
int main()
{
cout << "x from main: " << x << "\n";
cout << "y from main: " << y << "\n\n";
myFunction();
cout << "Back from myFunction!\n\n";
cout << "x from main: " << x << "\n";
cout << "y from main: " << y << "\n";
return 0;
}
void myFunction()
{
int y = 10;
cout << "x from myFunction: " << x << "\n";
cout << "y from myFunction: " << y << "\n\n";
}
```


مخرجات البرنامج 4.4 :

```
x from main: 5
y from main: 7
```

```
x from myFunction: 5
y from myFunction: 10
Back from myFunction!
x from main: 5
y from main: 7
```

ملاحظة://

* تذكر ان وسائط الدالة تعمل كمتغيرات محلية ضمن الدالة.

تذكر ان الوسائط التي تمرر بالقيمة لايمكن ان تؤثر على المتغيرات في دالة الاستدعاء.

تذكر ان تغيير المتغير العام في دالة معينة سيؤثر على قيمة هذا المتغير في جميع الدوال.

4.7 استدعاء الدالة

يقصد باستدعاء الدالة، هي العملية التي يتم فيها الطلب من الدالة لتنفيذ الشفرة الخاصة بها، ويتم ذلك من خلال كتابة اسم الدالة مع القوسين اللذين يحملان الوسائط الواجب تمريرها الى الدالة لتستخدمهما بانجاز عملها.. ويجب ان تلاحظ ان اسم الدالة عند الاستدعاء لايسبق بتعريف النوع، اما الوسائط فيجب ان يكون عددها مساويا الى عدد الوسائط في الدالة المستدعاة (عدا حالة سنأتي عليها لاحقا)، كذلك يجب ان تكون انواع الوسائط الممررة الى الدالة من نفس نوع وسائط الدالة وحسب ترتيبها (اي ان الوسيط الاول في دالة الاستدعاء يكون من نفس نوع الوسيط الاول في الدالة المستدعاة والثاني في دالة الاستدعاء نفس نوع الثاني وهكذا (عدا بعض الحالات المحدودة التي سنأتي عليها لاحقا). بعد استخدام هذه الوسائط في الدالة فان مخرجات الدالة ستعاد باستخدام عبارة الاعادة الى اسم الدالة, ومن اسم الدالة تنتقل القيمة الى دالة الاستدعاء (اي ان دالة الاستدعاء بالنتيجة ستحمل

قيمة ولذلك فهي يجب ان تخزن في الذاكرة وعملية الخزن تتم باسنادها الى متغير يمثل موقع في الذاكرة، او في حالة عدم الحاجة الى الخزن فيتم طباعتها مباشرة على الشاشة اذا لم تكن بحاجة لها في عمليات اخرى.

* برنامج لجمع عددين باستخدام الدوال، يوضح كيفية خزن نتائج الدالة

```
. // Example 4.5
#include <iostream>
using namespace std;
int addition (int a ،int b)
{
  int r;
  r=a+b;
  return (r);
}
int main ()
{
  int z;
  z = addition (5,3);
  cout << "The result is " << z;
  return 0;
}
```



```
The result is 8
```