الدرس الخامس:

في هذ ا الكتاب سوف نبني تطبيقاتنا باستخدام الدلفي 7 تعالو لنتعرف على نسخ الدلفي الاصدار السابع قبل الغوص في هذه اللغة:

1. The "Personal" edition : وهي نسخة للقادمين الجدد لهذه اللغة والمبرمجين العاديين ولايوجد فيها دعم لقواعد البيانات database ولا يوجد فيها أي من الميزات المتقدمة الاخرى.
2. "The "Professional Studio: وهي نسخة موجهة للمطورين المحترفين حيث تتضمن دعم لقواعد البيانات database وايضا دعم للويب (WebBroker) وتتضمن ايضا العديد من الادوات الخارجية
3. "The "Enterprise Studio: نسخة للمطورين حيث تتضمن دعم خاص لتقنية ال xml وتدعم الويب وتقنيات الهندسة المعمارية والعديد من الامور الاخرى
4. "The "Architect Studio: تعطي هذه النسخة دعم خاصا للمشاريع الضخمة حيث توفر العديد من الدعم لقواعد البيانات بواسطة تقنية ال UML ويوجد فيها الكثر الكثير من المكونات المتقدمة.

طبعا يوجد نسخ اخرى ل Delphi7 فهناك مثلا نسخة مصغرة منها وتتضمن فقط ادوات ال win32

التعليقات والطباعة الانيقة:

يقول ماركو في شرحه بهذا الصدد وذالك نقلا عن موقعه:

في باسكال، يتم ضمّ التعليقات في أقواس braces أو أقواس parentheses متبوعة بنجمة. دلفي تقبل ايضا نمط التعليقات المتبعة في س++ ، والتي يمكنها ان تمتد الى نهاية السطر:

```
{هذا تعليق}
(* هذا تعليق آخر *)
// هذا تعليق آخر يمتد حتى نهاية السطر
```

الشكل الأول أقصر و أكثر اتّباعا. الشكل الثاني كان مفضلا أكثر في اوربا بسبب عدم وجود رمز القوس السهمي في لوحات المفاتيح. الشكل الثالث من التعليقات تم استعارته من س++ و متوفر فقط في نسخ 32 بت من دلفي. التعليقات المحدودة بنهاية السطر مفيدة جدا للملاحظات القصيرة و لتلك الخاصة بسطر محدد في التوليف.

وجود ثلاثة اشكال مختلفة من التعليقات يمكن ان يساعد في بناء تعليقات متداخلة. إذا اردت التعليق على مجموعة أسطر من برنامج من اجل وقفها، وهذه الأسطر تحوي بعض التعليقات السابقة، فإنك لاتستطيع استخدام نفس علامة التعليقات:

```
{  ... code
{comment, creating problems}
... code }
```

مع علامة تعليق ثانية، يمكنك كتابة التعليمات الآتية، و التي هي صحيحة:

```
{  ... code
```

```
//this comment is OK
... code }
```

لاحظ أن القوس المفتوح او القوس_نجمة اذا كان تليه علامة الدولار ($)، فسوف يتحول الى توجيه للمجمّع compiler directive ، كما في {$X+}.

في الواقع، توجيهات المجمّع تعد تعليقات أيضا. مثال ذلك،{$X+ This is a comment} هي صحيحة. هي كلاهما توجيه و تعليق صحيحين، الا أن المبرمج المتعقل سوف يختار ان يفصل بين التوجيهات والتعليقات.

استخدام الأحرف العالية

مجمّع باسكال (بعكس اللغات الأخرى) يغضّ الطرف عن حالة الأحرف (عالية أو منخفضة). لذلك؛ فإن التعريفات التالية Myname، MyName، myname، myName، و MYNAME كلها متساوية. بشكل عام، هذا يعدّ أمرا ايجابيا، ففي اللغات الحسّاسة لحالة الأحرف، العديد من الأخطاء اللغوية قد تحدث بسبب الإهمال في مراعاة حالة الأحرف.

ملاحظة: ربما الحالة الإستثناء الوحيدة لقاعدة حساسية الأحرف في باسكال هي: إجراء *Register* في حزمة المكونات، لابد لها أن تبدأ بحرف *R* العالي، وذلك للمحافظة على التوافقية مع C++ Builder.

إلا أنه توجد بعض السلبيات. أولا، يجب أن تنتبه لأن تكون هذه التعريفات متساوية بالفعل، لذا يجب أن تتجنب استعمالها كعناصر مختلفة. ثانيا، يجب أن تكون متّسقا قدر الامكان عند استخدامك للأحرف العالية، لتحسين مقروئية برنامجك.

توحيد استخدام حالة الأحرف ليس ملزما من قبل المجمّع، و لكنها عادة حسنة يحبّذ اتّباعها. الاسلوب المتبع هو تكبير الحرف الأول فقط من كل معرّف identifier. و عندما يكون المعرّف مركّبا من عدة كلمات (لايمكنك حشر فراغ في المعرّف) ، فإن كل أول حرف من كل كلمة يجب أن يكون عاليا:

```
MyLongIdentifier
MyVeryLongAndAlmostStupidIdentifier
```

هناك حالات أخرى لايأبه لها المجمّع كالفراغات، و الأسطر الفارغة، و المسافات (tabs) التي تقوم بوضعها في البرنامج. كل هذه الحالات مجتمعة تسمّى بالفراغ الأبيض white space . الفراغات البيضاء تستخدم فقط لتحسين مقروئية البرنامج؛ و لا تؤثر في عملية التتجميع.

بعكس لغة بيسك BASIC ، فإن باسكال تسمح لك بكتابة تعليمة واحدة موزعة على عدة أسطر، فالتعليمة الطويلة يمكن تجزئتها لتكون في سطرين أو أكثر. السلبية الوحيدة (على الأقل بالنسبة لمبرمجى البيسك) لإمكانية أن تكون التعليمات في أكثر من سطر هي أنه عليك أن تتذكّر بأن تضع فاصلة منقوطة آخر كل تعليمة، أو بدقة أكثر، أن تفصل بين التعليمة والتي تليها. لاحظ أن القيد الوحيد هنا ان الجملة النصية الواحدة لايمكن مدّها لعدّة أسطر.

مرّة أخرى، لاتوجد قواعد ثابتة لاتستخدام الفراغات و التعليمات متعددة الأسطر، فقط بعض الأعراف:

محرّر نصوص دلفي له خطا عموديا تستطيع وضعه على بعد 60 أو 70 حرف. إذا استخدمت هذا الخطّ كمؤشر لتجنب تجاوزه، فإن برنامجك سوف يبدو أفضل عندما تقوم بطباعته على الورق. غير هذا فإن الأسطر الطويلة قد يتم تجزئتها من أي موضع حتى من منتصف الكلمة عند طباعة البرنامج.
عندما يكون للوظيفة أو الإجراء عدة محددات parameters ، فإن العادة المتّبعة هنا هي وضع هذه المحدّدات في سطر آخر.
تستطيع ترك سطر بكامله أبيضا (فارغا) قبل وضع أي تعليق أو ملاحظة، أو لتقسيم جزء كبير من التعليمات الى أجزاء أصغر. وحتى هذه الفكرة البسيطة بإمكانها تحسين مقروئية البرنامج، سواء على الشاشة أو عند طباعتها.
استخدم الفراغات لفصل محددات استدعاء وظيفة function call، وربما حتى فراغ قبل فتح الأقواس، ايضا حافظ على فراغات لفصل رموز العمليات في التعابير البرمجية. أنا أعلم أن بعض المبرمجين لن يوافقوا على هذه الفكرة، لكن أنا أصرّ: الفراغات بالمجان؛ لن تدفع شيئا مقابلها .(نعم، أعلم انها تستهلك مكانا للتخزين أو وقتا اضافيا في عملية اتصال الموديم لتحميل أو تنزيل ملف، لكن هذا أصبح لامعنى له في وقتنا الحاضر.)

الطباعة الأنيقة

الاقتراح الأخير فيما يخصّ استخدام الفراغات البيضاء له علاقة بالعرف المتبع لنسق تشكيل لغة باسكال، و الذي يعرف بالطباعة الأنيقة pretty-printing. القاعدة بسيطة: كل مرة تحتاج فيها لكتابة تعليمة مركّبة، قم بوضعها بعد هامش فراغين الى اليمين من باقي التعليمة الحالية. التعليمة المركبة داخل تعليمة أخرى مركبة يتم تهميشها بأربع مسافات، وهكذا:

```
if ... then
  statement;

if ... then
begin
  statement1;
  statement2;
end;

if ... then
begin
  if ... then
    statement1;
  statement2;
end;
```

الصياغة السابقة تعتمد نسق الطباعة الأنيقة، لكن المبرمجين لديهم تفسيرات مختلفة لهذه القاعدة العامة. بعض المبرمجين مثلا يقومون بتهميش تعليمات begin و end للمستوى التالي مع نفس التعليمات الداخلية، بعضهم يهمش begin و end ثم يقومون بتهميش التعليمات الداخلية لمستوى اضافي، مبرمجون آخرون يضعون begin في نفس سطر شرط if . هذا في معظمه أمر له علاقة بالذوق الشخصي.

نفس نمط التهميش يتّبع عادة عند سرد المتغيرات أو أنواع الببيانات، و لمواصلة تعليمة من سطر سابق:

```
type
  Letters = set of Char;
var
  Name: string;
begin
   { long comment and long statement, going on in the
     following line and indented two spaces }
   MessageDlg ('This is a message',
     mtInformation, [mbOk], 0);
```

بالطّبع، أي من هذه القواعد هي مجرّد إقتراح لجعل البرنامج مقروءا بشكل أفضل من قبل المبرمجين الآخرين، و هي تهمل بالكامل من جانب المجمّع. لقد حاولت استخدام هذه القاعدة بصورة متّسقة في كل اجزاء الأمثلة والبرامج في هذا الكتاب. كما يلاحظ أن البرامج، الأدلة، و أمثلة المساعدة التي تأتي مع دلفي كلها تتّبع نفس النسق في الصياغة.

تعليم الألفاظ

لتسهيل قراءة و كتابة توليف code باسكال، يملك محرر دلفي خاصية تسمّى تعليم الألفاظ syntax highlighting . فالكلمات التي تقوم بطباعتها في المحرر، يتم اظهارها باستخدام ألوان مختلفة بحسب معناها في باسكال. عرضا، الكلمات المفتاحية keywords تكون داكنة، النصوص و التعليقات تظهر ملونة (و غالبا مائلة)، وهكذا.

الكلمات المحجوزة، و التعليقات، و النصوص تقريبا هي العناصر الثلاثة الأكثر استفادة من هذه الخاصية. فمن أول نظرة يمكنك ملاحظة كلمة مفتاحية غير صحيحة، أو نصّ غير مقفل بصورة سليمة، أو طول الملاحظة المتعددة الأسطر.

بإمكانك بسهولة تعديل مواصفات تعليم الألفاظ باستخدام صفحة ألوان المحرر Editor page في لوحة خيارات البيئة Environment Options (انظر الشكل 2.1). إذا كنت تعمل بمفردك، يمكنك اختيار الألوان التي تفضل. أما إذا كنت تعمل بالتعاون مع مبرمجين آخرين، فالأفضل أن توافقوا جميعا على نسق ألوان نمطي. لقد وجدت ان العمل على حاسوب به تلوين الفاظ مختلف عمّا تعوّدت عليه أمر صعب بالفعل

واعتبره كافيا ووافيا.....

كما ورد في الدرس الثالث مصطلح البرمجة المهيكلة والاجرائية

سؤال هل الدلفي تسمح لنا بالبرمجة الاجرائية ام لا

الجواب اكيد ان الدلفي تسمح بذالك ان الدلفي هي امتداد للغة الباسكال وتطور لها فقط

ولذالك تقبل بلغة الباسكال تماما

مثال برنامج الاول بالطريقة المهيكلة برنامج hello World

افتح الدلفي

الان سوف ترى نموذج تنشئه الدلفي بشكل افتراضي لاعليك هذا النموذج هو للبرمجة تحت بيئة ويندوز

اما البرمجة المهيكلة فلانحتاجه

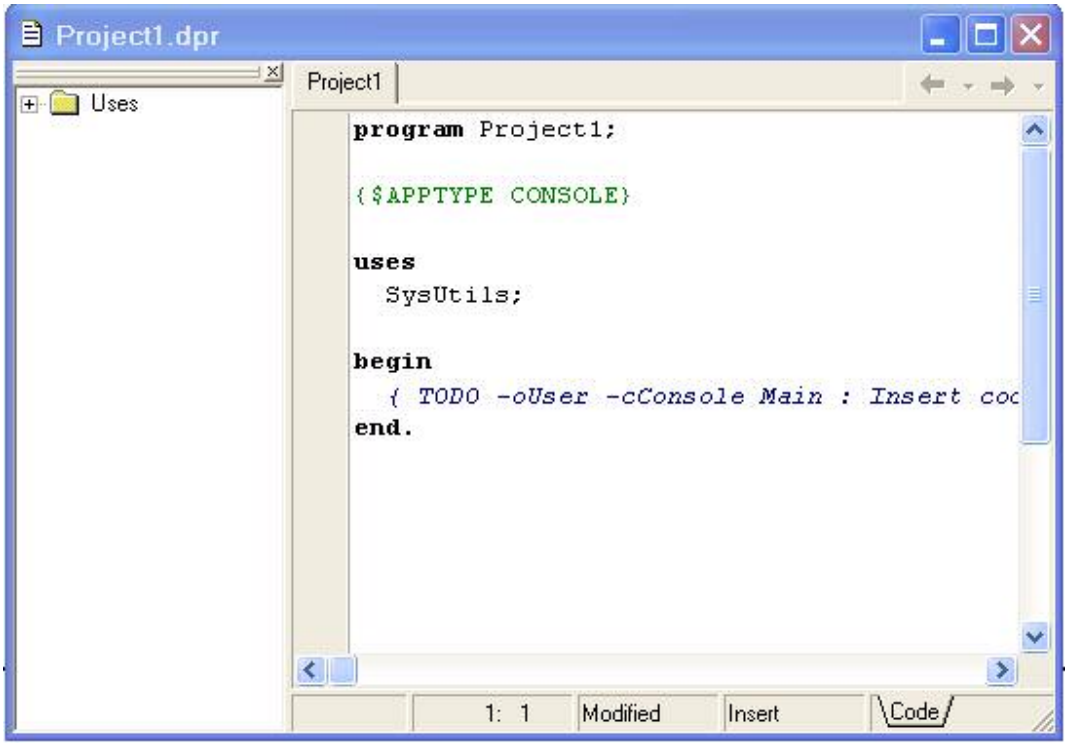
اغلق المشروع وذالك بالضغط على

file_____clse all

الان اضغط على

file_____new_____other_____applictian concole

سوف يظهر لك المشروع الاتي انظر



السطر الاول هو اسم المشروع اسمته الدلفي بشكل افتراضي طبعا بامكانك ان تغير اسم المشروع كما تريد سوف نسميه مشروع

hello world

السطر الثاني هو ارشاد للمجمع او المترجم سوف يتم شرح الارشادات فيما بعد

السطر الثالث

Usesهو عن الوحدات التي سوف يتك تضمينها بالمشروع وهنا تم وبشكل افتراضي تضمين وحدة النظام

السطر الرابع

Begin

end;

بين هذان السطران سوف تقوم انت بكتابة برنامجك

لاحظ التعليق بينهما حيث يخبرك الدلفي بانه عليك ادخال شيفرتك هنا

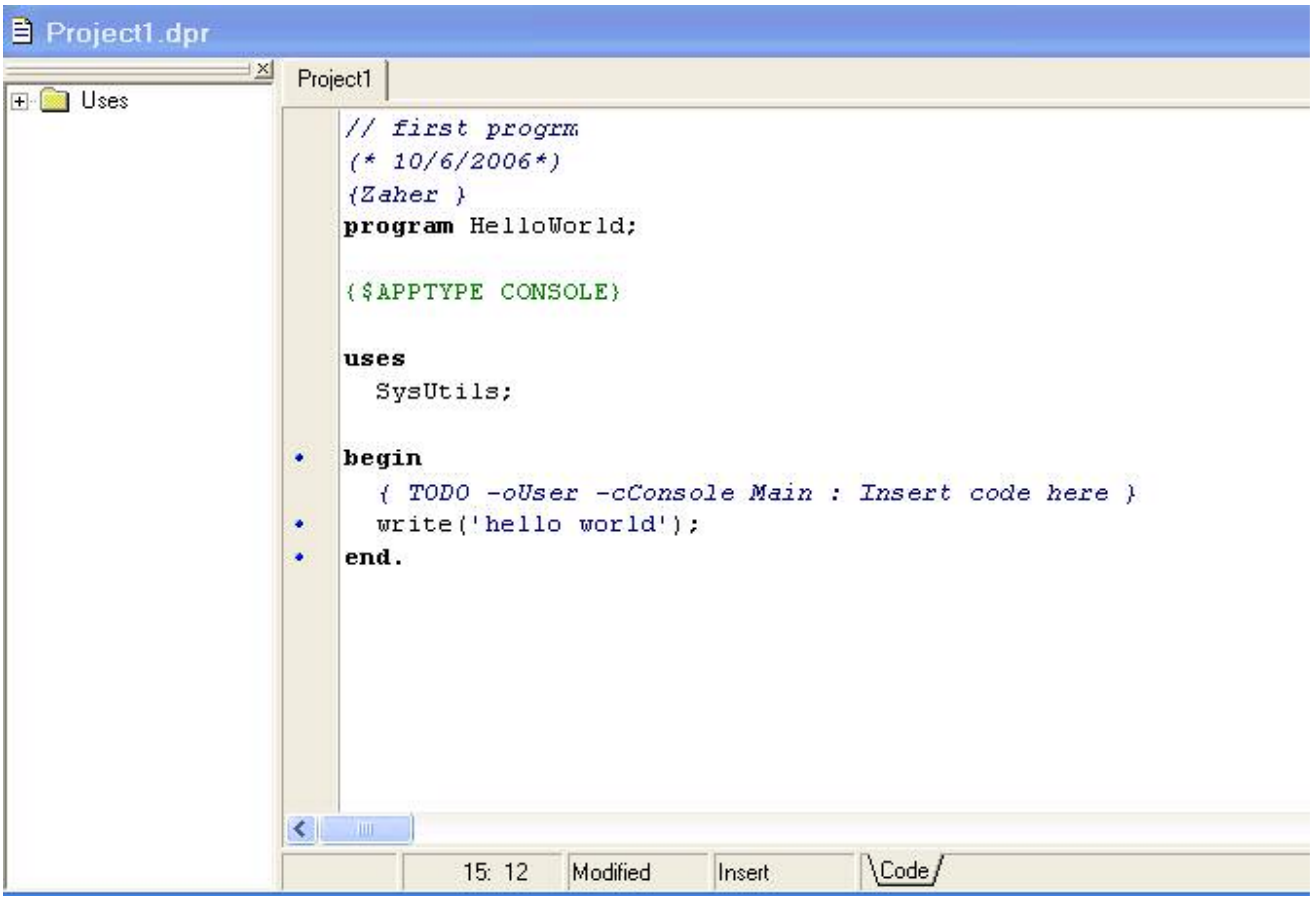
الان اطبع هذه الجملة

```
write('helloworld');
```

طبعا لاتنسى الفاصلة المنقوطة لتخبر المجمع ان تعليمتك قد انتهت

نفذ البرنامج بالضغط على مفتاح اف 9وسوف يكون الناتج كما في الصورة

لاحظو اصدقائي الفرق بين هذا التطبيق وبين التطبيق الاول في الدرس الثاني لنفس
المحاضرة....؟؟؟؟بقي علينا ان نكتب الصورة الكاملة للبرنامج وهي كما يلي



تنويه1 من المعروف بين المبرمجين كتابة باول كل برنامج تعليقات تتضمن تاريخ كتابته واسم المبرمج الذي كتبه ومعلومة عن البرنامج وكما ملاحظ اني كتبت هذه التعليقات بانواعها المختلفة

تنويه:من الجدير ذكره هو تسمية المشروع يجب ان يكون اول حرف كبير ولا يتخلله فراغات رغم ان الباسكال او الدلفي لاتميز بين الاحرف الكبيرة والصغير ولكن ينصح بها بشدة للتوافقية مع السي ++ وتحسين مقروئية البرنامج وايضا وضع الفراغات يسبب خطأ قواعديا

تنويه2كما تلاحظون مر معنا خلال شروحات ماركو في الاعلى تسمية تلوين الالفاظ وهذه ظاهرة في الالوان الاكلمات في البرنامج

تنويه3:كما تلاحظون في لائحة ال

uses

انه يمكن كتابة الوحدة بسطر الذي يليها او بنفس السطر وحت ان كان اكثر من وحدة يمكن كتابة كل واحدة بسطر او بنفس السطر او الاثنين معا مثال: هذه صحيحة

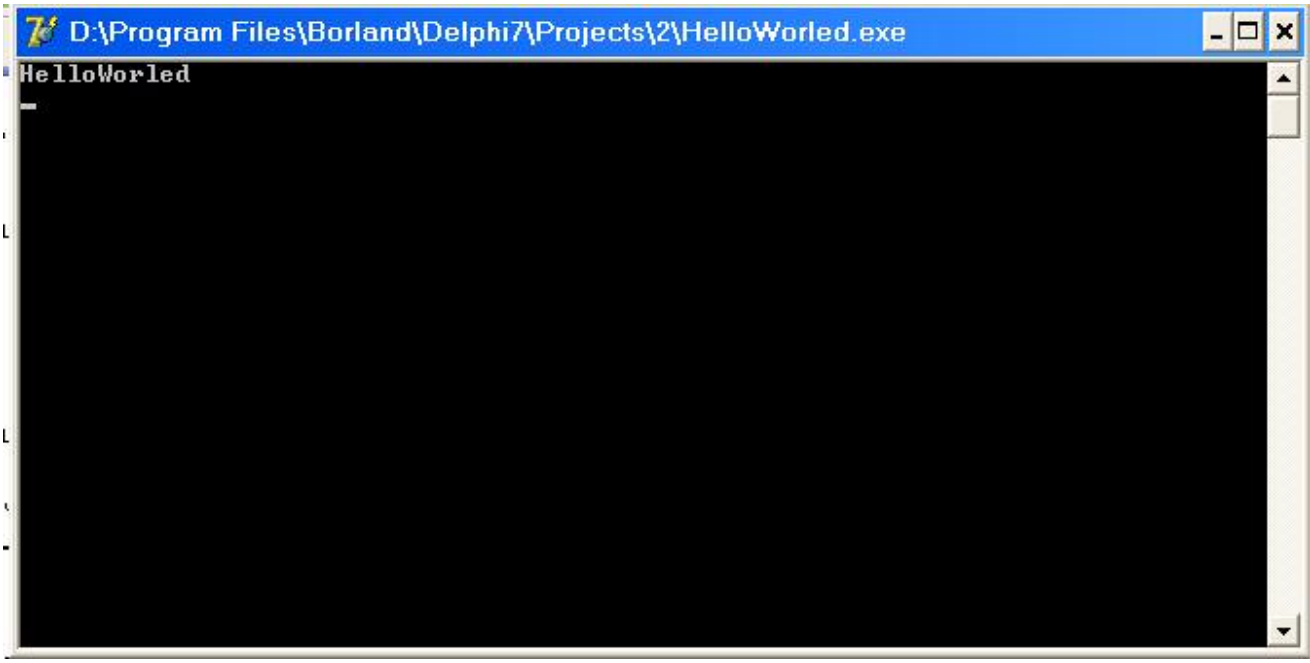
uses

SysUtils;

هذه صحيحة

uses SysUtils;

يمكننا تنفيذ نفس البرنامج السابق باستخدام الامر writelnوسوف يكون خرج البرنامج كما يلي



الفرق انه في خرج المثال الاول تم وضع المؤشر بنفس السطر اما في الخرج الثاني كما تلاحظون تم وضع المؤشر في السطر التالي يجدر ذكره وهو الهدف من المثال السابق في قابلية اعادة استخدام البرنامج وهذا يعد هدفا اساسيا في عملية التطوير البرمجي

تنويه4 ان الدلفي تحدد افتراضيا ان عمليات الدخل input هي لوحة المفاتيح وعمليات الخرج output هي شاشة العرض.

يمكنكم طباعة المثال السابق بعدة طرق فالمثال الاتي يستخدم عدة تعليمات لمجاري الخرج والتي تعطي نفس النتيجة

مثال:

```
begin
  { TODO -oUser -cConsole Main : Insert code here }
  write('Hello');
writeln('World');
end.
```

شرح البرنامج

تقوم التعليمة الاولى بطباعة كلمة Hello

ثم تتابع عملية الطباعة التعليمة الثانيى مباشرة

تسمح الباسكال والدلفي طبعا كما اغلب لغات البرمجة بالتعبير عن التعليمات بعدة طرق

يمكننا ايضا نطبع الجملة كل كلمة في سطر وذالك باستبدا تعليمة ال

writeln ب write

واذا اردنا وضع سطر فارغ يمكننا ذالك بسهولة كما في المثال

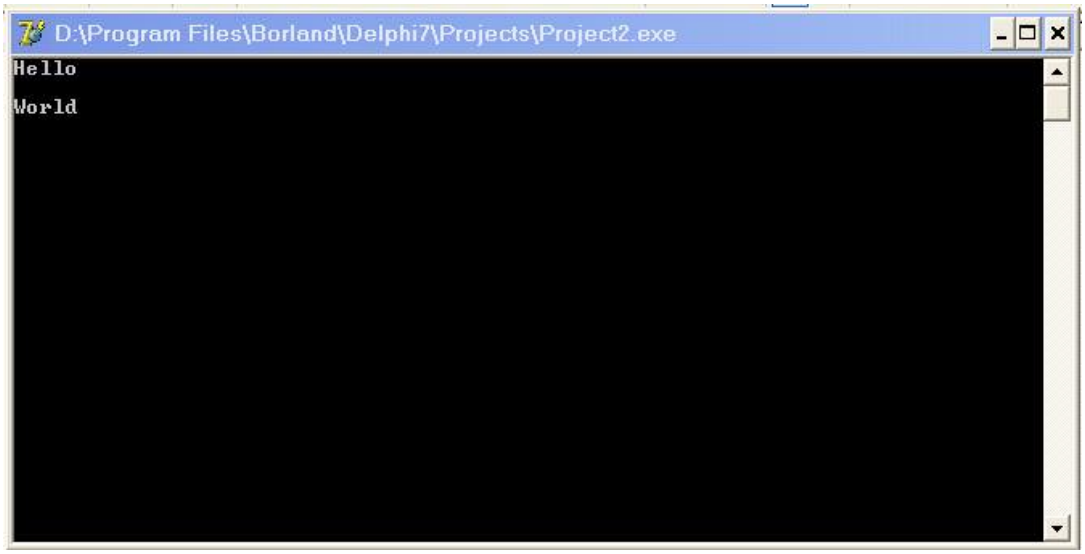
```
program HelloWorld;


{$APPTYPE CONSOLE}


{uses
  SysUtils;


begin
  { TODO -oUser -cConsole Main : Insert code here }
  writeln('Hello');
  writeln;
  writeln('World');
end.
```

ويكون الخرج كما في الصورة



انتهى الدرس الخامس

قمنا بهذا الدرس ذكر مفاهيم عديدة جديدة منها التعليقات وقد اوردت شرح لاحد خبراء البرمجة وهو ماركو عن التعليقات وقد بين في فقرة منفصلة شرح عن الطباعة الانيقة والتي كما ذكرنا في الدرس الثالث انها تدعم بقوة البرمجة المهيلكلة وتحسن من مقرؤية البرنامج كما وذكر ايضا تحت عنوان منفصل الاحرف العالية بضرورة مراعاة كتابة الاحرف العالية ايضا لتحسين مقرؤية البرنامج رغم ان مترجم اللغة يغض الطرف عليها في كثير من الاحيان وذكر ايضا تلوين الالفاظ عن خاصية جديدة بالدلفي وهي خاصية تلوين الالفاظ وقد قمت بالتنويه عليها كيف يتم تلوين التعليق بالازرق ووو....و

وقد انتقلت الى البرامج المهيكلة او التركيبية وكتابة البرامج بكود الباسكال والذي تعتمده الدلفي

وقدمت امثلة كافية للتعريف كيف يمكن كتابة برنامج الاول بالطريقة الباسكال طبعا للتنويه ليست كل الامثلة لاني سوف اعطي امثلة اخرى

كما وقدمت شرحا بسيطا عن مكونات المشروع ومايحتويه

الدرس لقادم سوف نتعلم الكلمة المفتاحية VAR

ونتعرف على مفهوم المتحولات او المتغيرات وسوف ننجز برامج وامثلة كافية ووافية عنها