لاحظت ان فى ضغط فى المنتدى على Pascal الفترة دى وكلام كتير عن Lazarus
فالمهم دى مجموعة اجزاء كنت مجهزها عن بسكال ولكن تحت التعديل لسه.. على كل حال بسكال فى رأيى لغة ممتعة (دا مش كان رأيى لما بدأتها لأنها اول لغة ليا)

بالمناسبة الدورة دى Windows/Linux عادى طالما فضلنا بصورة قياسية فسواء ويندوز او لينكس تقدر تتابع
الدورة متقسمة لكذا جزء.. الجزء الأول هو Pascal/Object Pascal واساسيتهم والجزء التانى مقدمة عن Lazarus والتعامل معاه

الجزء الأول Object Pascal

ودا الفهرس اللى هنشتغل عليه

<u>Chapter 1 (Programming Concepts)</u>

مامعنى كلمة Programming ؟

هى بتعنى القدرة على التخاطب مع الكمبيوتر وتنفيذ افكارك على ارض الواقع .. الكمبيوتر لايفهم اى شئ سوا 0 و 1 وصعب على البشر تعلمها إن لم يكن مستحيلا فنلجأ لخيارات بديلة وهى إستخدام لغات البرمجة

مامعنى Programming Language ؟

بكل بساطة هى وسيلة للتخاطب مع الكمبيوتر .. ولكننا قلنا إن الكمبيوتر لايفهم اى شئ سوا ال 0 وال 1 ومستحيل على الإنسان تعلمها! .. إذا الحل هو إستخدام لغات وسيطة .. على سبيل المثال واحد عربى وواحد فرنسى والعربى مش بيفهم فرنسى ولا الفرنسى بيفهم عربى .. فالحل هو إنهم يتكلمو إنجليزى مثلا... او يجيبو مترجم بين الإتنين مش كدا ؟
فهنا الحل إننا هنجيب مترجم يترجم افكارنا للغة الكمبيوتر 0 و 1 ويقوم المترجم بنفس الدور بتحويل رد فعل الكمبيوتر الى لغتنا المفهومة :)
وهنا دور ال Programming Language انت هتتعلم اللغة وكيفية التعامل معاها عشان تقدر تفهم المترجم "المقدم من اللغة" اللى إنت عايزه وهو يفهمه للكمبيوتر بدوره

ماهو ال Source Code ؟

بكل اختصار هو حلك لمسألة رياضيات وتفكيرك وإستنتاجاتك لما تكتبها فى ورقة ولكن هنا هو حلك لبرنامج مطلوب منك على ملف Text

مامعنى ال Debugging ؟

على فرض إنك بتحل مسألة رياضيات وفجأة إكتشفت خطأ فى طريقة حلك .. فإنت بتتبع المشكلة اللى حصلت وتشوف إزاى تصححها وهو دا معنى ال Debugging اى تصحيح الأخطاء :)

Compiled vs Interpreted
كتير منا إشتغلو على نظم Windows وكان ديما بيشوف ملفات إمتدادها .exe فإيه معنى ال exe ؟ معناها Executable او قابل للتنفيذ ..
فى لغات برمجة مثل ال C و Pascal بيتوافر الناتج النهائى بتاع برنامجك على صورة ملف exe وهو عبارة عن تعليماتك اللى إديتها للمترجم عشان يفهمها للكمبيوتر ولكن فى صورتها النهائية (الكلام اللى قاله المترجم لل كمبيوتر) فمستحيل على الإنسان إنه يقرا الملف دا وهنا معنى ال compiled فهى ملف ال exe يشمل التعليمات اللتى كتبتها ولكن بلغة الكمبيوتر وهو وحده القادر على فهمها

وإذا نظرنا من جانب آخر إلى لغات مثل Python, Perl هنجد إن الملف بيكون إمتداده .py او .pl ولكنك تقدر تفتحه فى اى Text Editor وتقراه -لفهمه لازم تكون عارف اللغة- والملف دا هو ال Source Code بتاعك نفسه بدون اى تحويلات ولا شئ ولكن لتنفيذه بنستدعى ال Interpreter فى كل مرة بحيث إنه يقرا ال Source Code ويبلغه للكمبيوتر ويتم التنفيذ

من مميزات ال Compiled Languages مثل ال C هى السرعة
ومن القصور هو إنك لازم تعمل Compile لل Source Code بتاعك على النظام اللى عايز تنفذ البرنامج عليه فبرنامج مكتوب على Windows محتاج يتعمله recompile على ال Linux وهكذا ..

من مميزات ال Intrepreted Languages هى انك ال Source Code بتاعك القياسى لايحتاج لعمل Recompile على مختلف النظم وال archS
ومن القصور البطء

ملحوظة:
لما بنتكلم على كلمة البطء فى ال Interpreted Languages بيكون المقصود البطء بالنسبة لل Compiled Language وليس البطء للمستخدم لأنك مش هتلاحظ الفرق لأن البطء فى شئ لايكاد يذكر

History
لغة Pascal ظهرت عام 1970 على يد العالم Niklaus Wirth وكان هدفها
1- صغر الحجم
2- وضوح كامل
3- تعليم المبرمجين ال Structured Programming بصورة جيدة ومنظمة
بعد دخول مفهوم ال OOP تم تطوير Object Pascal لدعم ال OOP بصورة متكاملة

Borland تعتبر اكبر داعم ل Pascal منذ ايام Turbo Pascal وقامت بتحسينها وتطويرها إلى الصورة الحالية

Delphi: هى IDE متكاملة لل RAD على Windows
Kylix: هى IDE متكاملة لل RAD على *NIX

حاليا تم إلغاء مشروع Kylix -ليست لديهم نية استمرار الدعم-

Borland: كل اللى نقدر نقوله هو
We don't mind paying for good software :)

لجذب المبرمجين لمنتجات Borland قامو بتطوير Turbo Explorer لتطوير C++, C#, Delphi for win32 and Delphi.net

ملحوظة: معظم مبرمجى Delphi مازالو غير مهتمين بال NET. ومازالو بيستخدمو Win32!

على كل حال فلسفة ال Open Source لازم تلعب دورها كالعادة فأخرجت لينا
Free Pascal Compiler او FPC للإختصار وهو Multi-Platform Compiler -وهو اللى هنستخدمه- يخضع ل GPL

GNU Pascal Compiler او GPC وهو Compiler من GNU ويعمل على معظم النظم ايضا

Lazarus: هو Class Libraries محاكية ل Delphi و IDE متكاملة تساعدك على ال RAD بكل بساطة فبكل إختصار Lazarus هو البديل الأقوى ل Delphi
سبب التسمية: على إسم Lazarus -بالكتاب المقدس- اللذى احياه المسيح من الموت. وهو ماينطبق على Lazarus حيث تم إحيائه من Megido

المقدمة من ويكيبديا و الويكى الخاص ب FPC, Lazarus

Downloading/Installing
كل المطلوب منك إنك تدخل هنا http://sourceforge.net/project/showf...group_id=89339
وتختار المناسب ليك
لو مش عندك FPC يبقة إختار Package يكون فيها ال FPC كمان

بعد ماحملت Lazarus وستبته
File -> New -> Program

اعمل save لل project بإسم hello
هتلقة ال code التالى بال Editor

كود:
```
program hello;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };

begin
end.
```
دا هو ال Skeleton بتاع برنامجك
(Run -> Build or (Ctrl + F9
(Run or (F9
هتلقة شاشة سودا فتحت وقفلت بسرعة فكل اللى اقدر اقوله مبروك ادى اول برنامج ليك فى Pascal :)

لاحظ فى معظم الأحيان برنامجنا هيكون بين .begin, end
ماهى .begin, end ؟
begin هى نقطة البداية لبرنامجك
end. هى نقطة النهاية لبرنامجك

اعتقد المفروض نبدأ Hello, World مش كدا ؟
WriteLn
هى ال Function الشهيرة للإخراج :)
بنستخدم WriteLn فى حال عايزين نطبع شئ للمستخدم كالتالى مثلا

كود:
```
program hello;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };

begin
     WriteLn('Hello, World!');
end.
```
لاحظ ال Generated Code بيبدأ ب program hello وهنا إن اسم البرنامج hello و uses هى section

بيتحط فيه ال units اللى هنستخدمها -هنتعرض ليها تفصيليا ان شاء الله-
اعمل Run للبرنامج هتلقة كلمة Hello, World ظهرت واختفت الشاشة بسرعة! ليه الشاشة اختفت ؟
السبب هو إن مش فى سبب تفضل لأن البرنامج انتهى التنفيذ!!
فكل اللى علينا إننا نخلي البرنامج يستنى لحد مدة معينة زى مثلا لحد مالمستخدم يضغط Enter او كدا
فبكل بساطة هنضيف ReadLn وهى هتساعدنا فى إن البرنامج ينتظر اى محاولة إدخال من المستخدم فيفضل البرنامج شغال :)

كود:

```
begin
     WriteLn('Hello, World!');
     ReadLn();
end.
```

رائع جدا كدا ال Screen بتستنى لحد ماتدوس على Enter

طب احنا البرنامج اللى كتبناه دا كانت وظيفته إيه ؟ او إيه كانت وظيفة كل سطر لأنى احتمال انسى ؟
begin هى البداية

كود:

```
WriteLn('Hello, World!');
```

بتطبع كلمة Hello, World

كود:

```
ReadLn();
```

بتخلى البرنامج مستنى المستخدم ليدوس Enter
end. هى نهاية البرنامج

طب ياسلام لو نقدر نكتب الكلام دا فى برنامجنا صح ؟ عشان نوضحه لينا ونوضحه للى هيقرا الكود بتاعنا ... كدا اديك عرفت Concept جديد وهو ال Commenting

كود:

```
begin   {The Start}
     WriteLn('Hello, World!'); {Writes Hello, World to the output}
     ReadLn(); {Waites for <Enter>}

end.    {The End}
```

تقدر تستخدم // بدل من ال { } كالتالى مثلا
//Writes Hello, World to the output

نعمل برنامج بسيط عددين هنشتغل فيه كالتالى
1- هنعرض رسالة للمستخدم نطلب منه إنه يدخل الرقم الأول
2- نقرا الرقم الأول
3- نعرض رسالة نطلب فيها الرقم التانى
4- نقرا الرقم التانى
5- نجمع الرقمين
6- نعرض حاصل جمع الرقمين للمستخدم

تحليلنا للبرنامج على خطوات محددة هو ال Algorithm اللى هنشتغل بيها لحل المشكلة

-البرنامج-

كود:

```
var
   First: Integer;
   Second: Integer;
   Result: Real;

begin
     Write('Enter the first: ');  // ask for a number
     ReadLn(First);               // Read it.

     Write('Enter the second: ');  // ask for a number
     ReadLn(Second);               // Read it.

     Result := First + Second;     // Summing.
     WriteLn('Result: ', Result) ; // Write out the Result.

     ReadLn();                     // Wait for <RETURN>
end.
```

ماهى المتغيرات ؟ هى عبارة عن alias ل data معينة بنستخدمها داخل برنامجنا ف First هنا بتمثل الرقم اللى هيدخله ال user و second كذلك و Result تمثل حاصل الجمع
بنقوم بتحديد كل المتغيرات بتاعتنا فى Section مخصوص بإسم var بنحدد فيه نوع ال Variable
لإعطاء قيمة لمتغير بنستخدم =: Operator

كود:

```
Result := First+Second;
```

لنعطى قيمة لمتغير هنستخدم ReadLn وهى هتاخد ال Input من المستخدم وتعمل implicit cast لل data type وهنا تم التحويل من ال Input (string) إلى Integer

كلمة بخصوص ال Data Types
Char-> لتمثيل الحروف
Integer -> لتمثيل الأعداد الصحيحة
Real -> للأعداد الحقيقية
String -> للسلاسل النصية
Boolean -> True, False
تابع هنا

ابع هذا المثال عن ال Data Types

كود:

```
var
  name: String[10];
  age : Integer;
  sex : Char;
  married: Boolean;
begin
  name := 'Ahmed';
  age  := 19;
  sex  := 'm';
  married := False;

  WriteLn('Name: ',name);
```

```
  WriteLn('Age : ',age);
  WriteLn('Sex : ',sex);
  WriteLn('Married? ',married);

  ReadLn();

end.
```

نخلية More Interactive :)

كود:

```
var
  name: String ;
  age : Integer;
  sex : Char;

begin
  WriteLn('Gathering data...');
  Write('Name: ');
  ReadLn(name);

  Write('Age: ');
  ReadLn(age);

  Write('Sex: ');
  ReadLn(sex);

  WriteLn('Data Entered Successfully');
  WriteLn('Your Profile');
  WriteLn(' name: ', name);
  WriteLn(' age : ', age);
  WriteLn(' sex : ', sex);

  ReadLn(); //Wait for <Return>

end.
```

Constants

هى عبارة عن aliases ل data لن تتغير خلال برنامجك لأى سبب.. على سبيل المثال يوجد Constant شهير وهو PI وقيمته 3.14

كود:

```
program constants1;

const
  PI: Real = 3.14;

var
  radius: Real;
  area: Real;

begin
  Write('Enter the radius: ');
  ReadLn(radius);
  area := PI*(radius*radius);
  WriteLn('Area: ', area);
```

```
  ReadLn();
end.
```

ال Operators

+ للجمع

- للطرح

* للضرب

/ للقسمة Real

div للقسمة Integer

mod لباقى القسمة

Pascal بتقدملك طرق للتحكم فى برنامج وإختبارات -زى كل اللغات- خلال البرنامج مثل if, then ,elseif, else, case, .. etc.

if <Condition is True> then
if_suite

تابع المثال التالى فى حال إذا كان ال Name اللى هيدخله المستخدم قيمته ahmed هيتنفذ ال Block اللى بعد then لو لأ مش هيتنفذ حاجة -لأننا مش علمنا برنامجنا إنه يعمل شئ معين لو Name مش قيمته احمد :)

كود:

```
var
  Name: String;

begin
  Write('Enter your name: ');
  ReadLn(name);

  if name = 'ahmed' then
   WriteLn('Access Granted.');
  {else
   WriteLn('Access Denied.');
  }
  ReadLn();

end.
```

ملحوظة:
:= -> لإسناد قيمة
= -> لإختبار قيمة

X := 5;

هنا اعطينا المتغير X ال 5

x = 5;

هنا بنختبر هل ال x = 5 او لأ!

if <True> then
if_suite
else
else_suite

كود:

```
var
  Name: String;

begin
  Write('Enter your name: ');
  ReadLn(name);

  if name = 'ahmed' then
   WriteLn('Access Granted.')
  else
   WriteLn('Access Denied.');

  ReadLn();

end.
```

لاحظ دور else هنا .. إن إذا تم إدخال اى قيمة غير ahmed هيتنفذ ال Block التابع ل else وهو Access Denied!

كود:

```
var
  age: Integer;

begin

  WriteLn('Enter your age: ');
  ReadLn(age);

  if age < 18 then
     WriteLn('Not old enough for driving test.');
  else
     WriteLn('GOOD LUCK!');
end.
```

لو العمر اقل من 18 هيتنفذ ال Block دا

كود:

```
WriteLn('Not old enough for driving test.');
```

لو لأ هيتنفذ

كود:

```
    WriteLn('GOOD LUCK!');
```

<u>if/else if/ else</u>

if <condition> then
if_suite
elseif <True> then
elseif_suite
else

else_suite

تابع المثال التالى
كود:

```
var
  Number: Integer;

begin
     Write('Enter a number in range 1, 5: ');
     ReadLn(Number);

     if Number=1 then
        WriteLn('One!')
     else if Number=2 then
        WriteLn('Two!')
     else if Number=3 then
        WriteLn('Three!')
     else if Number=4 then
        WriteLn('Four!')
     else if Number=5 then
        WriteLn('Five!')
     else
        WriteLn('It is not in range 1 to 5');

     ReadLn();
end.
```

لاحظ طالما بنعمل Fall فالأفضل نستخدم switch statement كالتالى
كود:

```
var
   dayNum: Integer;

begin
     Write('Enter: ');
     ReadLn(dayNum);

     case dayNum of
          1: WriteLn('Sunday');
          2: WriteLn('Monday');
          3: WriteLn('Tuesday');
          4: WriteLn('Wednesday');
          5: WriteLn('Thursday');
          6: WriteLn('Friday');
          7: WriteLn('Saturday');
          else //default
               begin
                    WriteLn('U R on EARTH!');
               end;

     end;

     ReadLn;
end.
```

لاحظ هنا اننا بنعمل check على dayNum ونشوف فى اى case هتتوافق معاه...
نقدر نستخدم case مع ranges -واضح من كلامنا ان range معناها D: range

كود:

```
var
   Age: Integer;

begin
     Write('Age? ');
     ReadLn(Age);

     case Age of
          1..5 : WriteLn('Cute Baby!');
          6..10: WriteLn('Mean kid!');
          11..14: WriteLn('Boy!');
          15..18: WriteLn('Teen!');
          19..23: WriteLn('Adult!');
          24..40: WriteLn('Regular guy');
          41..60: WriteLn('Gettin older!');
          else
              begin
                   WriteLn('R U Human?');
              end;
     end;
```

Looping

مممم اى Loop عبارة عن تكرار suite او مجموعة من ال statements ل
1 - عدد من المرات
2- لحد ما Condition يتحقق

طبعا كلام عجيب وزى الفل :D

التكرار لعدد من المرات معناه اقرب مثال .. نفذ 100 تمرين ضغط !
تكرار لحد ما Condition يتحقق مثلا
طالما مش فى اكل
اعمل اكل!

while <True> do
loop_suite

كود:

```
   while i <= 5 do
           begin
                WriteLn(i);
                Inc(i); // i := i + 1;
           end;
```

طالما i اقل من او تساوى 5 هيتم تنفيذ ال block التالى

كود:

```
begin
                WriteLn(i);
                Inc(i); // i := i + 1;
```

```
            end;
```

فى نهاية تنفيذ ال Block السابق i هتكون قيمتها 6!

كود:

```
     While i >= 0 do
            begin
                 WriteLn(i);
                 Dec(i); // i := i - 1;
            end;
```

طالما i اكبر من او تساوى صفر يتنفذ ال Block التالى

كود:

```
begin
                 WriteLn(i);
                 Dec(i); // i := i - 1;
            end;
```

فى loop تانية وهى

كود:

```
repeat
        loop_suite
until <True>;
```

لازم تتنفذ ولو loop_suite بتتنفذ قبل مايتم إختبارها.. فعشان كدا ال loop_suite الفرق هو إن ال مرة واحدة على الأقل!

فى loop ال وهى كتير هنشوفها for loop وهى كالتالى
for <start> to <end> do
for_suite

تصاعدية
for <high> downto <low> do
for_suite
تنازلية
كود:

```
var
   counter: Integer;

begin
     for counter := 0 to 10 do
          begin
               WriteLn('Counter: #', counter);
          end;

          ReadLn;
end.
```

تنبيه: إذا متعود على

كود:

```
for(start; cond; inc){
        for_suite
}
```

فالأفضل هنا إنك تستخدم ال while loop لتكون اسهل فى التحكم فى مقدار ال step اللى هتزيد

Functions vs Procedures

كثير من الناس للأسف لايعلم الفرق بين ال Function وال Procedure! فى الواقع معظم اللغات حاليا لاتفرق بينهم .. فى اللى بيعتبر ان ال procedure هو حالة خاصة من Function وفى بيعتقد العكس ان ال Function هى الحالة الخاصة من ال procedure وهكذا!

ملحوظة لمبرمجى C/C++/C# اى Function ال return type بتاعها void هى procedure !

نتكلم الأول إيه هى ال Function او ال Procedure ؟
بإختصار هى كود اتكتب مرة وهيستخدم بكثرة فبدل مايتكتب كل مرة .. كتبناه مرة واحدة وبقينا نستخدمه داخل برنامجنا.. على سبيل المثال .. اننا عايزين نعرض رسالة معينة فى برنامجنا بصورة دورية وليكن كالتالى مثلا
كود:

```
WriteLn('Hi');
     WriteLn('Hi');
     WriteLn('Hi');
```

هل تعتقد إن يعقل إنك تكتب ال 4 سطور دول فى كل مرة عايز تظهر فيها الرسالة ؟ او على فرض إنك غيرت الرسالة من Hi ل Hola مثلا هل هتقعد تعدل فى كل السطور ؟
رائع جدا فإحنا عايزين نكتب ال block دا مرة واحدة وننفذه كل ماحبينا بدون مانكرر نفسنا! فبكل بساطة هنكتب procedure او function تنفذلنا اللى احنا عايزينه كالتالى مثلا
كود:

```
procedure SayHi3Times;
    var
       counter : Integer;
     begin
           WriteLn('Hi');
           WriteLn('Hi');
          WriteLn('Hi');
    end;
```

الكود السابق يعمل 100% ولكن مش تصميم حلو ل procedure ليه ؟ على فرض إننا حبينا نغير ال implementation بتاعنا او عايزين نخليها 4 times او او او .. فلازم نراعى تصميم ال procedure بطريقة جيدة كالتالى مثلا
كود:

```
procedure SayHi3Times;
    var
       counter : Integer;
     begin
       for counter := 1 to 3 do
           begin
                WriteLn('Hi');
           end;
    end;
```

ونستدعيه فى اى مكان فى برنامجنا كالتالى
كود:

```
   SayHi3Times;
```

رائع جدا بالنسبة ل Procedure بيعرض رسالة hi ل 3 مرات! طب على فرض إنى عايز اغير ال 3 مرات دول اخليهم 4 او 5 ؟ قدامى اكتر من حل ... منها إنى اعدل ال implementation نفسها

واخلى ال counter يعد لحد ال 4 او ال 5 .. او إنى اخلي ال Procedure مصمم بطريقة افضل ..
بحيث إنى احدد عدد مرات الرسالة بدون ماصدع نفسى :)
كود:

```
procedure SayHi(NumberOfTimes: Integer);
     var
        counter: Integer;
     begin
          for counter := 1 to NumberOfTimes do
              begin
                   WriteLn('Hi!');
              end;
     end;
```

لاحظ هنا اننا حددنا ان عدد مرات تكرار الرسالة ك Parameter وهو NumberOfTimes وحددنا ان
ال Data Type بتاعه هو Integer
فكدا الموضوع اصبح اسهل :)
كود:

```
SayHi(5); //Say hi 5 times.
     SayHi(7); //Say hi 7 times.
```

وكدا اصبح كل المطلوب مننا هو اننا نباصى عدد مرات التكرار ك argument لل procedure SayHi!
جميل .. كدا نقدر نطلع بتعريف لل procedure .. هو code اتكتب ليتم استخدامه اكتر من مرة
بدون الحاجة لكتابته فى كل مرة
لاحظ الفرق بين ال Parameter وال argument: هو إن ال parameter هو اللى متحدد انه هيتباصى
لل Function فى التعريف بتاعها ولكن ال argument هو اللى تباصى بالفعل!

نيجى لل Functions
على فرض إننا عايزين نحسب قيمة مساحة مربع مثلا طول ضلعه 5 مثلا ولكن مش عايزين نكرر
نفسنا فى كل الكود ف إننا نكتب العلاقة
بالمناسبة العلاقة هى مربع طول الضلع :D
كود:

```
function AreaOfSquare5: Integer;
      begin
           Result := 5*5;
      end;
```

ونستخدمه كالتالى
كود:

```
WriteLn('The Area of Square (5 units length): ', AreaOfSquare5);
```

واحد هيسأل ايه قيمة AreaOfSquare5 ؟ اللى موجودة فى WriteLn ؟
الإجابة هى الشئ الوحيد اللى تقدر تخمنه :) هى قيمة ال Result
وبكل تأكيد ال Integer اللى موجودة فى السطر دا
function AreaOfSquare5: Integer;
بتعبر عن ال Data Type الخاص ب Result !
رائع جدا .. لاحظ إنك مش تقدر تباصى procedure لأن ال procedure مش ليها Result هو إجراء
بينفذ شئ معين مش اكتر ولكن ال Function بتنفذ شئ معين + ترجع ليك النتيجة النهائية :)
ملحوظة لمعظم المبرمجين: Result هى Return
ال Function بتاعتنا شغالة زى الفل بالنسبة ل Function تحسب مساحة مربع طول ضلعه 5مش
كدا ؟ :)

ولكن كناس يهمها انها مش تكرر نفسها! لازم نعيد تصميم ال Function بحيث إنها تساعدنا مع كل الأطوال مش ال 5 بس ؟
اها صح .. كدا انا تأكدت إنك فهمت ال Function Parameters بنسبة 100% :D
نحدد طول الضلع ك Parameter فى ال Function ونتعامل بناء عليه :D

كود:

```
function AreaOfSquare(Side: Integer): Integer;
      begin
           Result := Side*Side;
      end;
```

ونستخدمها بالصورة دى

كود:

```
     WriteLn('The Area of Square (3 units length): ', AreaOfSquare(3));
     WriteLn('The Area of Square (9 units length): ', AreaOfSquare(9));
```

تابع ال Function التالية..

كود:

```
function AreaOfRect(H: Integer; W:Integer): Integer;
      begin
           Result := H*W;
      end;
```

المفروض إنها تحسبلنا مساحة اى مستطيل طوله H وعرضه W (وهنا باصيناهم ك Parameters) ليكون الكود بتاعنا ابسط + لعدم الحاجة لكتابة 100 مليون Function بنفس الإسم لكل طول وكل عرض !
والإستخدام كالتالى

كود:

```
WriteLn('The Area of Rect   (7H, 4W)       : ', AreaOfRect(7, 4));
```

بسيطة ها ؟

نيجى لحاجة مهمة وهى ال Overloading .. مشكلة ال C انها كانت الأسامى بتخلص فيها بسرعة.. تخيل مثلا مش ينفع تكتب غير Function واحدة بإسم واحد! ؟ لكن اللغات الحديثة معظمها بيقدم ال Overloading وفيها بيكون ال Compiler/interpreter ذكى بيقدر يحدد اي Function المبرمج إستخدمها بناء على عدد ال Argument اللى هو باصاها لل Function!

<u>Overloading</u>
تعالى نشوف مثال ل overloading على procedure وليكن hola

كود:

```
procedure hola;
```

دا ال procedure الأساسى ولكن إكتشفنا إننا ممكن عايزين نحدد عدد المرات ك argument لل procedure !
فبكل بساطة

كود:

```
procedure hola(Times: Integer); overload;
```

معنى Overload اننا هنعمل implementation خاصة ل hola فى حال لو اتباصى ليه Integer

والكمبيلر مشكلته هو انه يعرف اى procedure هيستدعيه!

كود:

```
hola
```

هنا الكمبيلر هيستدعى الأولى

كود:

```
    hola(7);
```

وهنا هيستدعى التانية

مثال اخر

كود:

```
function Area(Side:Integer): Integer;
```

هنا هيتم استدعاء ال Function دى لو اتباصى ليها Argument واحدة معبرة عن ال Side لو إستخدمناها بالشكل دا

كود:

```
 WriteLn('The Area of Square (5 units length): ', Area(5));
```

كود:

```
function Area(H: Integer; W: Integer): Integer; overload;
```

وهنا تعريف تانى مختلف وبيتم إستدعاءه فى حال لو إتباصى 2 arguments ك H, W لمستطيل وهكذا ...

لاحظ القابليه لوجود اكتر من صورة ل Function او Procedure او او هى ال Polymorphism وخليها فى بالك لحد مانوصل لل OOP :)

Inline

انك تحدد ال Function على إنها inline بيعنى استخدامها ك macro لمستخدمى ال C (مع اختلاف جوهرى ان شاء الله هنتعرض ليه)
بإختصار .. اذا ال Function بتاعتك صغيرة فعرفها على انها inline لتضمن اداء افضل فى برنامجك

ولكن خلى بالك ان حجم ال executable هيزيد!

مثلا Function للحصول على ال ABS او القيمة المطلقة لعدد

القيمة المطلقة: هى القيمة الموجبة للعدد او العدد بدون الإشارة السالبة إذا كانت موجودة
مثلا 8 .. القيمة المطلقة ليها هى 8
مثلا -8 .. القيمة المطلقة ليها هى 8

كود:

```
function ABS(Num: Integer) :Integer;
    begin
         if Num > 0 then
            Result := Num
         else
            Result := -Num;
    end;
```

ولكنك المفترض إنك تعرفها على إنها inline وت implement ال Function على إنها inline كالتالى

مثلا

کود:

```
function ABS(Num: Integer) :Integer; inline; //Faster.
    begin
         if Num > 0 then
            Result := Num
         else
            Result := -Num;
    end;
```

Arrays

تخيل إنك بتكتب برنامج وليكن بتتعامل فيه مع الطلاب اللى معاك فى الفصل -خارج الكلام عن قواعد البيانات- مثلا
ممكن تعمل مثلا 30 متغير يعبرو عن ال 30 طالب مش كدا ؟

كود:

```
student1;
  student2;
  student3;
  student4;
  student5;
```

مممم تأكد انك لو عملت كدا والكود بتاعك اتراجع هتكون مطرود من الشغل فى تانى دقيقة :D

الحل الطبيعى إنك تعمل Array تربط فيها بين ال 30 طالب دول

كود:

```
 students: array[1..4] of string;
```

لاحظ لتقوم بتعريف اى array يلزمك
1- ال Range وهنا من 1 ل 4
2 نوع ال Items اللى فى ال Array وهنا String

ونتعامل مع ال Array اللى عرفناها كالتالى

كود:

```
students[1] := 'Ahmed';
     students[2] := 'Christina';
     students[3] := 'Rogina';
     students[4] := 'Youssef';
```

لاحظ اننا حددنا ان اول عنصر هو 1 واخر عنصر هو 4 (من ال Range)
تقدر تحصل على القيمة المخزنة فى index معين بإنك تحدد ال index فقط كالتالى مثلا
WriteLn('Students[1]: ', students[1]); //Ahmed;
ملحوظة: معظم اللغات بتبدأ ال arrays فيها من ال 0 مش 1 بعكس Pascal تقدر تحدد اول قيمة تبدأ فيها براحتك.. فانا عن نفسى مازلت بستخدم ال 0 كبداية مع Pascal زي باقى اللغات.. ولحل هذه المشكلة بنستخدم 2 Function يسهلو علينا الموضوع دا بغض النظر عن البداية والنهاية لل Array كالتالى مثلا

كود:

```
for index:=Low(students) to  High(students) do
         begin
              WriteLn('Student[', index, ']: ', students[index]);
         end;
```

Low هتعيد اول Index فى ال Array
High هتعيد اخر Index فى ال Array

احيانا قد نحتاج إلى Dynamic array بمعنى إننا نحدد عدد العناصر فى ال run-time فبكل بساطة
students: array of string; // we don't specifiy the range as it's zero based array.
لاحظ إن ال array هنا هتكون zero-indexed

بعد كدا نحدد ال length بإستخدام SetLength

كود:
```
SetLength(students, stNumber);
```
حيث ان stNumber هو عدد ال students ..

EnumS

بكل بساطة هى تصميم Data Type جديد مكون من قيم محددة مثلا الفصول او ايام الإسبوع او الإتجاهات!

كود:
```
Type
 TDay = (Saturday=1, Sunday=2, Monday=3, Tuesday=4, Wednesday=5, Thursday=6,
Friday=7);
```
مثلا نهاية الإسبوع هو Friday
فالأول نعلن إن نهاية الإسبوع هو من النوع Tday يعنى هياخد قيمة من اللى تم تحديدها فى Tday

كود:
```
var
  weekend: TDay; //friday..
نديلة القيمة Friday
     weekend := Friday;
```
بسيطة مش كدا ؟
;TDays = set of TDay
لاحظ ان TDays هى عبارة عن set من ال TDay اى مجموعة هتشمل بعض العناصر اللى من النوع TDay

وبكدا عملنا مجموعة خاصة بال أيام .. نقدر نعمل منها متغير وليكن بإسم busyDays ونحط فيه الأيام اللى الواحد بيكون مشغول فيها !

كود:
```
busyDays : Tdays;

busyDays:=[];
```
كود:
```
Include(set, VALUE);
```
ونقدر نضيف فيها بإستخدام Include

كود:
```
Exclude(set, VALUE);
```
نقدر نحذف منها بإستخدام Exclude

كود:
```
   busyDays := busyDays + [Saturday]; {Include(set of enum, value)}
   busyDays := busyDays - [Sunday];   {Exclude(set of enum, value)}
```
تابع المثال التالى

كود:
```
type
```

```
    TDay = (Saturday=1, Sunday=2, Monday=3, Tuesday=4, Wednesday=5,
Thursday=6, Friday=7);
    TAlphabet= 'a'..'z';
    TDays = set of TDay;


var
  weekend: TDay; //friday..
  today: TDay;
  c: TAlphabet;
  busyDays : TDays;
begin
     weekend := Friday;
     busyDays:=[];
     Include(busyDays, Saturday);
     Include(busyDays, Sunday);
     Include(busyDays, Friday);

     //busyDays := busyDays + [Saturday]; {Include(set of enum, value)}
     //busyDays := busyDays - [Sunday];   {Exclude(set of enum, value)}

     today := Saturday;

     if today = Saturday then
        WriteLn('Happy Saturday!');

     if Sunday in busyDays then
        WriteLn('I am busy -> Sunday');
```

لاحظ تقدر تختبر وجود value ما فى set معينة كالتالى

if val in someSet then
if_suite

Records

ال Record هو Type جديد بتنشأه بيكون فيه معلومات مترابطة عن شئ معين وليكن كتاب بدل ماتقول كود:

```
name_of_book_1 := 'Introduction to Ruby';
pages_of_book_1 := 147;
author_of_book_1 := 'ahmed';
```

ولا

كود:

```
book1.Name := 'Introduction to Ruby';
book1.nOfPages := 147;
book1.author := 'ahmed' ;
```

هل لاحظت الفرق ؟

الكود اكتر وضوح ومترابط اكتر !
طب .. نقول ل Pascal على اللى فى بالنا

كود:

```
type
    TBook = record
          Name: String;
          Pages: Integer;
          Author: String;
    end;
لاحظ إن فى convention اى نبدأ اننا Type ال بحرف جديد T
 var
   rubyBook: TBook;
   pythonBook: TBook;
   javaBook: Tbook;
```

هنا انشأءنا 3 متغيرات من النوع Tbook وهم rubyBook و PythonBook و javaBook
نديلهم شوية قيم

كود:

```
rubyBook.Name:='Introduction to Ruby';
   rubyBook.Pages:=147;
   rubyBook.Author:='Ahmed Youssef';

   pythonBook.Author:='Ahmed Youssef';
   pythonBook.Pages:=258;
   pythonBook.Name:='Introduction to Python';

   javaBook.Author:='Ahmed Youssef';
   javaBook.Name:='Introduction to Java';
   javaBook.Pages:=159;
```

جميل نقدر الوقتى نعرض كل Field فيهم مثلا

كود:

```
WriteLn(javaBook.Author);
WriteLn(javaBook.Name);
WriteLn(javaBook.Pages);
```

ولكن هل يعقل إننا نكتب الكلام دا 3 مرات ؟
بالظبط كدا انت فهمتنى لازم نعمل Extract ل Procedure يقوم بالمهمة دى عننا كالتالى مثلا

كود:

```
procedure ShowInfo(book: TBook);
begin
     WriteLn('Name: ', book.Name);
     WriteLn('Pages: ', book.Pages);
     WriteLn('Author: ', book.Author);

end;
```

هنا ال ShowInfo Procedure هياخد book ك argument من نوع Tbook ويعرض لينا ال Fields الخاصة بيه Name, pages, Author

نقدر نضم Records جوا Records بالطبع تابع المثال التالى هيجاوب كل اللى فى بالك عن ال Records

كود:

```
program records;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };

type
    TBook = record
          Name: String;
          Pages: Integer;
          Author: String;
    end;
    TSex = (MALE, FEMALE);

    TPerson = record
            Name: String;
            Age : ShortInt;
            Sex : TSex;
    end;

    TApartment=record
         Owner: TPerson;
         Area: integer;
         NumberOfRooms: ShortInt;
         PopulatedWith: ShortInt;
    end;


procedure ShowInfo(book: TBook);
begin
     WriteLn('Name: ', book.Name);
     WriteLn('Pages: ', book.Pages);
     WriteLn('Author: ', book.Author);

end;
var
   MyApartment:TApartment;
   Me : TPerson;
   rubyBook: TBook;
   pythonBook: TBook;
   javaBook: TBook;

begin
   rubyBook.Name:='Introduction to Ruby';
   rubyBook.Pages:=147;
   rubyBook.Author:='Ahmed youssef';

   pythonBook.Author:='Ahmed youssef';
   pythonBook.Pages:=258;
   pythonBook.Name:='Introduction to Python';
```

```
   javaBook.Author:='Ahmed Yousef';
   javaBook.Name:='Introduction to Java';
   javaBook.Pages:=159;


   Me.Age:=19;
   Me.Name:='Ahmed';
   Me.Sex:=MALE;

   WriteLn('My Name is: ', Me.Name);
   WriteLn('My Age  is: ', Me.Age);
   if Me.Sex = Male then
      WriteLn('And MALE!')
   else
      WriteLn('And a WOMAN!');

   MyApartment.Area := 400;
   MyApartment.Owner:=Me;
   MyApartment.NumberOfRooms:=10;
   MyApartment.PopulatedWith:=4;

   WriteLn('Name of the owner -> ', MyApartment.Owner.Name);
   WriteLn('Area  -> ', MyApartment.Area);
   WriteLn('Number of rooms -> ', MyApartment.NumberOfRooms);
   WriteLn('Populated With  -> ', MyApartment.PopulatedWith);

   ReadLn;

end.
```

Pointers

ال Pointers هى متغير بيحتوى على قيمة ولكن القيمة هى عنوان متغير تانى من نفس نوعه .. تابع المثال التالى

كود:

```
num: Integer;
```

هنا num هو متغير من نوع Integer

كود:

```
   p_num: Pinteger;  // Or you may use ^Integer;
```

هنا p_num هو pointer بيشير إلى متغير من النوع Integer

;num := 5

اعطينا ل num ال value مساوية ل 5

...p_num :=@num; //address of num

اعطينا ل p_num ال value وهى مساوية لعنوان num فى الميمورى ونقدر نحصل عليها بإستخدام ال @ قبل اسم المتغير.

كود:

```
    WriteLn('Address of num: ', Integer(@num));
    WriteLn('Value of num: ', num);
    WriteLn('p_num, points to the address: ', Integer(p_num));
    WriteLn('p_num^, the value contained at the address that p_num points to is: ',     Integer(p_num^));
```

للحصول على القيمة المخزنة فى المتغير اللذي يشير ليه p_num نقدر نحصل عليها كالتالى ^p_num

بكل تأكيد تستطيع ان تستخدم pointer ل anonymouse addresses! يعنى مش شرط تعمل متغير وتجعل ال pointer يشير ليه.. لأ، تقدر تحجز منطقة فى ال ميمورى وتخلى ال pointer يشير ليها بدون اهتمامك بإسم هذه المنطقة!

كالتالى مثلا

كود:

```
var
   p: PInteger; // is a pointer to integer
begin
   //allocate memory for it
   New(p); // Simple ha ?
   WriteLn('P points to: ', Integer(p));
   //set value
   P^ := 7;
   WriteLn('p^: ', p^);

   //Free memory!
   FreeMem(p);

   ReadLn;
end.
```

لاحظ ان Pascal حدد حجم الميمورى المطلوبة بناء على نوع p بإستخدام New(p)

يوجد طرق اخرى زى GetMem -هنتعرض ليها لاحقا-

خزنا قيمة فى الميمورى اللى p بيشير ليها وهى 7
وحررنا ال memory اللى حجزناها بإستخدام FreeMem

لاحظ فى البرنامج التالى إننا إستخدمنا P ليكون Pointer ل Integer و Char وتم ذلك بإننا عملنا Cast ليه فى الحالتين!

كود:

```
     program dynamicP;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };
var
   dp: Pointer;
   i : Integer;
   c : Char;
begin
     i := 5;
     c := 'a';

     //let dp points to i.
     PInteger(dp) := @i;   //cast dp into PInteger type.
     WriteLn('@i : ', Integer(@i));
     WriteLn('dp: ', Integer(dp));

     WriteLn('i: ', i);
     WriteLn('dp^ :', Integer(dp^));

     //let dp points to c.

     PChar(dp) := @c;  //cast dp into PChar type.
     WriteLn('@c: ', Integer(@c));
     WriteLn('dp: ', Integer(dp));

     WriteLn('c: ', c);
     WriteLn('dp^: ', Char(dp^));


     //Wait for <Return>
     ReadLn;


end.
```

OOP Concepts 1

ال Object Oriented Programming موضوع لذيذ جدا وهو مش فيه اى جديد عن حياتك.. اذا بصيت لأى شئ حواليك هتلقيه Object انسان عربية طيارة شجرة عمارة.. الخ
كلهم بيشتركو فى كلمة انهم Object! ولكن لكل شئ فيهم خصائص محددة ووظائف خاصة فيه نيجى الأول نفرق ال Class عن ال Object
على فرض عندنا لوحة مرسوم عليها تصميم عمارة => هى دى الكلاس
لكن اذا تم انشاء العمارة من التصميم دا => هو دا الأوبجكت
عندك تصميم لروبوت ومتحدد عليه خصائصه ووظايفه => هو دا الكلاس
اذا شركة اخدت التصميم دا ونفذته => هو دا الأوبجكت

على فرض اننا بنعمل تصميم لإنسان (على الورق) يعنى الكلاس
كود:

```
type THuman=class(TObject)

     Constructor Create(aName: String; aSex: Char; aColor: String);
     private
            fname: String;
            fsex: Char;
            fcolor: String;


     public

            Function GetSex(): Char;
            Procedure SetSex(aSex: Char);
            Function GetColor(): String;
            Procedure SetColor(aColor: String);
            Function GetName(): String;
            Procedure SetName(aName:String);
            Procedure Eat();
            Procedure Sleep();



end;
```

اول كلمة هتقلها WTH!!
ايه كل دا ؟
نمشى مع الكود واحدة واحدة
كود:

```
type Thuman=class(TObject)
```

معناها أننا بنعرف Type جديد بإسم Thuman (لاحظ ان اى Type بنبدأه ب T "مجرد عادة"
الإنسان دا ليه مميزات او صفات زى ان ليه اسم وليه لون وليه نوع ودى اسمها fields
(لاحظ ان اى field بنبدأه ب f)
ودى مش نسمح لأى حد انه يلمسها -يعدل فيها او يتعامل معاها- غيرى انا -الكلاس- وبموافقتى وشروطى ودا بيتم من خلال ال Getters/Setters methods لو لاحظت ان كل Field ليه Getter/Setter

fname ليها

كود:

```
Function GetName(): String;
            Procedure SetName(aName:String);
```

fsex ليها

كود:

```
Function GetSex(): Char;
            Procedure SetSex(aSex: Char);
```

fcolor ليها

كود:

```
                        Function GetColor(): String;
            Procedure SetColor(aColor: String);
```

ال Get/Set Methods بنبدأها بشئ مناسب Get/Set :)

نيجى بعد كدا للوظائف اللى بيعملها ال Human دا وهى مثلا Eat, Sleep و walk و Work وغيرهم دى اسمها methods

لاحظ اى Function/Procedure داخل كلاس بيسمى Method او تحديدا Instance Methods إلا فى حال اننا حددناها انها Class Method (مش تصعبها على نفسك حاليا)

جميل جدا كدا احنا بنخلى العالم الخارجى يتعامل مع ال Fields الخاصة بالكلاس عن طريق ال Getters/Setters ولكن اذا لاحظت انها غلسة شوية.. ناس كتير اشتغلت Vbاو Delphi وكان عندهم مفهوم كلمة ال Properties وهنا الموضوع بيكون الذ شوية كالتالى

كود:

```
type THuman=class(TObject)

 Constructor Create(aName: String; aSex: Char; aColor: String);
     private
            fname: String;
            fsex: Char;
            fcolor: String;
            Function GetSex(): Char;
            Procedure SetSex(aSex: Char);
            Function GetColor(): String;
            Procedure SetColor(aColor: String);
            Function GetName(): String;
            Procedure SetName(aName:String);
            Procedure Eat();
            Procedure Sleep();
     public


           property Name: String read GetName write SetName;
           property Sex: Char read GetSex write SetSex;
           property Color: String read GetColor write SetColor;
           Procedure InspectMe();
```

```
end;
```

لاحظ اننا خلينا ال Getters/Setters فى ال Private Section :S
طب العالم الخارجى هيتعامل ازاى مع ال Fields بتاعتنا كدا ؟ بكل بساطة هيستخدم ال Properties زى Name, Sex, Color

لعمل اى Property بتستخدم الصيغة العامة كالتالى

كود:

```
property property_name : Type read Getter write Setter;
```

كالتالى مثلا

كود:

```
      property Name: String read GetName write SetName;
```

هنا عندنا Property بإسم Name وهى من النوع String (بتتعامل بال String) سواء فى عملية ال Get او ال Set

كود:

```
.Name=val //set(write), sets fname to a string
.Name //Get(read), returns fname (a string)
```

جميل سامع واحد بيقول انت بتشتغلنا ياعم! فى الكود نفسه؟
هقله اتقل شوية احنا لحد الآن فى ال Interface (مبرمجى السى/++ بيحبو يسموه ال Header) لسه هنعمل ال Implementation فى قسم لوحدها كالتالى

كود:

```
Constructor THuman.Create(aName: String; aSex: Char; aColor: String);
begin
     //inherited;
     fname:=aName;
     fsex:=aSex;
     fcolor:=aColor;
end;

Function THuman.GetName(): String;
begin
     Result := Self.fname;
end;

Function THuman.GetSex(): Char;
begin
     Result := Self.fsex;
end;

Procedure THuman.SetName(aName: String);
begin
     Self.fname:=aName;
end;

Procedure THuman.SetSex(aSex: Char);
begin
     Self.fsex:=aSex;
end;

Function THuman.GetColor(): String;
```

```
begin
     Result:= Self.fcolor;
end;

Procedure THuman.SetColor(aColor: String) ;
begin
     Self.fcolor:=aColor;
end;
Procedure THuman.InspectMe() ;
begin

     WriteLn('Name: ', Self.fname);
     WriteLn('Color: ', Self.fcolor);
     WriteLn('Sex: ', Self.fsex);
end;
Procedure THuman.Eat()
begin
     //Code to eat.
end;
Procedure THuman.Sleep()
begin
     //Code to sleep.
end;
```

اوبس فى واحد بيقولى انت نسيت تقول ال Constructor دا يعنى إيه!

ال Constructor بكل بساطة هو Function ولكن ليها اسم مميز Constructor .. اشمعنة يعنى ؟
جميل احنا قلنا ان اى اوبجكت بيمر على مرحلتين
المرحلة الأولى التصميم والتانى الإنشاء
التصميم هو اللى احنا قاعدين نجهز فيه بقالنا فترة وتحديد ال Fields و ال Methods وال Properties وغيرهم لكن الإنشاء هو اننا نحول التصميم بقة دا لأوبجكت ودا بيتم عن طريق ال Constructor "المنشئ او المشيد" وفيه بيتجهز ال Data Members (مسمى تانى لل Fields) بالقيم الإفتراضية مثلا ال String ياخد Null وال Int ياخد Zero وهكذا * او * القيم اللى انت عايزها (ال Arguments اللى هتتباصى لل Constructor) فعلا انت راجل 10/10
زى ماحصل هنا بالظبط

كود:

```
Constructor THuman.Create(aName: String; aSex: Char; aColor: String);
begin
     //inherited;
     fname:=aName;
     fsex:=aSex;
     fcolor:=aColor;
end;
```

ملحوظة: مش لازم تسمى ال Constructor بإسم Create ولكنه تقليد بردو ^_^
جميل جدا الكلاس الجميل اللى احنا كتبناه دا مش هنسيبه متعلق فى الهوا كدا لازم نحطه فى مكان مناسب ليه

اعمل new unit
File -> New Unit
وسميها وليكن Creatures وضيف فيها اللى احنا كتبناه كالتالى
ملحوظة: ال UNIT هى ملف بنجمع فيه الأجزاء المترابطة من شغلنا مثلا لو بندرس ال OOP وبنعمل كلاسات للدراسة زى Human, Robot, Bird, .. etc نخليها Creatures او مثلا Employer,

Employee, Customer, .. etc تخليها Business وهكذا...

كود:

```
unit creatures;

{$mode objfpc}{$H+}


interface
type THuman=class(TObject)

     Constructor Create(aName: String; aSex: Char; aColor: String);
     private
            fname: String;
            fsex: Char;
            fcolor: String;
            Function GetSex(): Char;
            Procedure SetSex(aSex: Char);
            Function GetColor(): String;
            Procedure SetColor(aColor: String);
            Function GetName(): String;
            Procedure SetName(aName:String);
            Procedure Eat();
            Procedure Sleep();
     public


           property Name: String read GetName write SetName;
           property Sex: Char read GetSex write SetSex;
           property Color: String read GetColor write SetColor;
           Procedure InspectMe();



end;


implementation

Constructor THuman.Create(aName: String; aSex: Char; aColor: String);
begin
     //inherited;
     fname:=aName;
     fsex:=aSex;
     fcolor:=aColor;
end;

Function THuman.GetName(): String;
begin
     Result := Self.fname;
end;

Function THuman.GetSex(): Char;
begin
     Result := Self.fsex;
end;
```

```
Procedure THuman.SetName(aName: String);
begin
     Self.fname:=aName;
end;

Procedure THuman.SetSex(aSex: Char);
begin
     Self.fsex:=aSex;
end;

Function THuman.GetColor(): String;
begin
     Result:= Self.fcolor;
end;

Procedure THuman.SetColor(aColor: String) ;
begin
     Self.fcolor:=aColor;
end;
Procedure THuman.InspectMe() ;
begin

     WriteLn('Name: ', Self.fname);
     WriteLn('Color: ', Self.fcolor);
     WriteLn('Sex: ', Self.fsex);
end;
Procedure THuman.Eat()
begin
     //Code to eat.
end;
Procedure THuman.Sleep()
begin
     //Code to sleep.
end;

end.
```

واعمل save ليها
وتعالى نختبرها فى اى برنامج مثلا oopconcepts1
كالتالى مثلا

كود:

```
program ooConcepts1;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes, creatures
  { you can add units after this };

var
   ahmed: THuman ;
begin
```

```
     ahmed:=THuman.Create('Ahmed', 'm', 'white');
     ahmed.InspectMe();

     ahmed.Name:='youssef';

     ahmed.InspectMe();

     ReadLn;
end.
```

نكمل بمثال نشرح فيه الوراثة

اعمل برنامج جديد وضيف ليه unit بإسم creatures

كود:

```
type

{ THuman }

THuman=class(TObject)

     Constructor Create(aName: String; aSex: Char; aColor: String);
     private
            fname: String;
            fsex: Char;
            fcolor: String;
            function GetColor: String;
            function GetName: String;
            Function GetSex(): Char;
            Procedure SetSex(aSex: Char);
            Procedure SetColor(aColor: String);
            Procedure SetName(aName:String);



     public

           class Procedure About; virtual;
           property Name: String read GetName write SetName;
           property Sex: Char read GetSex write SetSex;
           property Color: String read GetColor write SetColor;
           Procedure Eat();
           Procedure Sleep();
           Function InspectMe(): String; virtual; //Should be overriden in
the child.

end;
```

كلاس جديد بإسم THuman

فى هنا fields لل name, sex, color وفى private getters/setters
وفى public functions/procedures ومجموعة من ال properties
about: يعرض كلمة عن الكلاس دا والمفترض انه يتغير فى ال child
ال prototype ليه كالتالى

كود:

```
          class Procedure About; virtual;
```

هنا لاحظ اننا بادئين التعريف ب class لنعبر على ان الميثود دى خاصة بالكلاس يعنى تستدعيها ب Thuman.About

بدون الحاجة لإنشاء object من ال Thuman class
فى Function بإسم InspectMe للحصول معلومات عن ال fields الموجودة بالكلاس ولكن المفروض نعيد تعريفها فى اى كلاس هيورثه

نيجى لل implementation بتاعت الكلاس دا

كود:
```
{ THuman }

constructor THuman.Create(aName: String; aSex: Char; aColor: String);
begin
     Self.fname:=aName;
     Self.fsex:=aSex;
     Self.fcolor:=aColor;


end;

function THuman.GetColor: String;
begin
     Result:=Self.fcolor;
end;

function THuman.GetName: String;
begin
    Result:=self.fname;
end;

function THuman.GetSex(): Char;
begin
   Result:=self.fsex;
end;


procedure THuman.SetSex(aSex: Char);
begin
  self.fsex:=aSex;
end;



procedure THuman.SetColor(aColor: String);
begin
     self.fcolor:=aColor;
end;


procedure THuman.SetName(aName: String);
begin
   self.fname:=aName;
end;

procedure THuman.Eat();
begin
    WriteLn('Eatin');
end;
```

```
procedure THuman.Sleep();
begin
    WriteLn('Sleepin');
end;

class procedure THuman.About;
begin
    WriteLn('Human class');
end;

function THuman.InspectMe(): String;
begin
    Result:='Name: '+self.fname + LineEnding + 'Sex: '+self.fsex +
LineEnding+ 'Color: '+self.fcolor;
end;
```

واضحة زى المثال السابق ولكن ال Thman.InspectMe محتاجة بعض التوضيح

كود:

```
function THuman.InspectMe(): String;
begin
    Result:='Name: '+self.fname + LineEnding + 'Sex: '+self.fsex +
LineEnding+ 'Color: '+self.fcolor;
end;
```

هى بتعمل دمج لمجموعة من ال strings + بتضيف LineEnding (لو متأقلم مع ال Escape Sequences فهى ال \n)

نيجى بقة لل TEmployer هو كلاس بيعبر عن انسان ليه اسم ولون ونوع + (مرتب)
فيإما نكتب كل الكلاس THuman داخل ال Temployer او نستخدم ال وراثة بإننا نورث ال Thuman لل Temployer ونضيف الجديد ليه

كود:

```
type

{ TEmployer }

TEmployer=class(THuman)
      Constructor Create(aName: String; aSex: Char; aColor: String;
aSalary:Integer);

      private
             fsalary:Integer;
             Function GetSalary():Integer;
             Procedure SetSalary(aSalary:Integer);

      public
            class Procedure About; override;
            Function InspectMe(): String;override;
            property Salary:Integer read GetSalary write SetSalary;


end;
```

بمجرد انك كتبت

كود:

```
Temployer=class(THuman)
```

فهنا ال Temployer هيورث كل اللى فيه ال Thuman

ال

كود:

```
 Function InspectMe(): String;override;
```

لازم نعيد تعريفها بحيث تتوافق مع الخصائص الجديدة للكلاس دا زى المرتب مثلا!
لاحظ اننا هنعيد تعريف ال About, InspectMe
نيجى لل Implementation بتاعته

كود:

```
{ TEmployer }

constructor TEmployer.Create(aName: String; aSex: Char; aColor: String;
  aSalary: Integer);
begin
     inherited Create(aName, aSex, aColor);
     self.fsalary:=aSalary;

end;
```

اولا ال Constructor
هنا الأول لازم نباصى ال قيم الخاصة بالكلاس الأب اللى هى ال aName, aSex, aColor
للكونستركتور الخاص بال THuman وال fields الجديدة نجهزها زى ال fsalary
لاحظ استخدام inherited هنا دى بتنقلك للسوبر كلاس الخاص بال Temployer وبعدها Create
"بتعبر عن ال Thuman.Create "

كود:

```
function TEmployer.GetSalary(): Integer;
begin
     Result := self.fsalary;
end;



procedure TEmployer.SetSalary(aSalary: Integer);
begin
       self.fsalary:=aSalary;

end;
```

كود:

```
class procedure TEmployer.About;
begin
  WriteLn('Employer class!');
end;
```

هنا اعادة تعريف ل About

كود:

```
function TEmployer.InspectMe(): String;
begin
     Result := inherited InspectMe + LineEnding + 'Salary:
```

```
'+IntToStr(self.fsalary);
end;
```

هنا اعادة تعريف ل InspectMe بنستدعى ال InspectMe بتاعت السوبر اللى هترجعلنا String

كود:

```
function THuman.InspectMe(): String;
begin
    Result:='Name: '+self.fname + LineEnding + 'Sex: '+self.fsex +
LineEnding+ 'Color: '+self.fcolor;
end;
```

وبعدها ندمج الناتج ب LineEnding و جملة

كود:

```
'Salary:' +IntToStr(self.fsalary)
```

هنا بنحول قيمة ال fsalary ل string لأنها integer ومش ينفع ندمج string مع integer

نيجى للبرنامج بقة

كود:

```
program ooConcepts1;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  sysutils, Classes, creatures
  { you can add units after this };

var
   ahmed: TEmployer;

begin

    ahmed:=TEmployer.Create('Ahmed', 'm', 'white', 90000);
    WriteLn(ahmed.InspectMe());

    //Set name
     ahmed.Name:='youssef';
    //Set salary
    ahmed.Salary:=100000;
    WriteLn(ahmed.InspectMe());
    TEmployer.About;
    ahmed.Eat;


    ReadLn;
end.
```

انشئ اوبجكت من ال Temployer

كود:

```
ahmed:=TEmployer.Create('Ahmed', 'm', 'white', 90000);
```

اعرض بيناته

كود:

```
WriteLn(ahmed.InspectMe());
```

تقدر تعدل فى ال Properties براحتك

اخيرا استخدام ال class method

كود:

```
Temployer.About;
```

وهى ميثود بتنتمى للكلاس مش للأوبجكت

File Handling

اهم الميثودز

كود:

```
Reset(F)
```

تفتح فايل فى مود القراءة

كود:

```
Rewrite(F)
```

تفتح فايل فى مود الكتابة

كود:

```
Append(F)
```

بتفتح فايل فى مود الإضافة

كود:

```
Assign(F, file_path)
```

بتسند الفايل المفتوح ل File Handler من النوع File

Flush

بت flush ال buffer

كود:

```
Close(F)
```

بتغلق ال File Handler

كود:

```
EOLN(F)
```

نهاية السطر؟

كود:

```
EOF(F)
```

نهاية الملف؟

كود:

```
Erase(F)
```

لحذف ملف ما

كود PHP:

Rename(F, new_name)

تسمية F ل new_name

كود:

```
Seek(F, pos)
```

بتنقل ال File pos ل pos معين فى ال File

كود:

```
SeekEOF(F)
```

بتنقل ال File pos لي نهاية الفايل

كود:

```
SeekEOLN(F)
```

بتنقل ال File pos ل نهاية السطر

التعامل مع ال text files

اولا تنشئ متغير من النوع Text فى ال var section

كود:

```
   F:Text;
```

2- تسند ليه فايل ما

كود:

```
Assign(F, 'textf.txt');
```

الكتابة

3- للكتابة استدعى ReWrite عليه

كود:

```
Rewrite(F);
```

4- اكتب فيه بإستخدام WriteLn

كود:

```
   For I:=0 To 10 do
       begin
            WriteLn(F,'Hello, World');
       end;
```

طبعا عرف ال I فى ال var section

كود:

```
   I: Integer;
```

الإضافة

3- استدعى ال Append عليه

كود:

```
Append(F);
```

4- ضيف اى حاجة انت عايزها وليكن سطر واحد مثلا كالتالى

كود:

```
   WriteLn(F,'a HELLO WORLD LINE!');
```

القراءة

3- استدعى ال Reset عليه

كود:

```
   Reset(F);
```

4- تقدر تقرا السطور بإستخدام ReadLn كالتالى مثلا

كود:

```
   //Loop till the EOF
   Repeat
         ReadLn(F, Line);
         WriteLn(Line);

   Until EOF(F);
```

هنا هيتم قرائة كل سطر وطباعته (فى متغير Line:String تم تعريفة فى ال var section)

اخيرا تقفل الهاندلر
5-اقفل الهاندلر

كود:

```
   Close(F);
```

بعض ال Helpers

كود:

```
ExtractFileName(F)
```

الحصول على اسم

كود:

```
ExtractFilePath(F)
```

الحصول على المسار الموجود فيه الفايل

كود:

```
ExtractFileExt(F)
```

الحصول على امتداده

كود:

```
ExtractFileDir(F)
```

الحصول على اسم الفولدر الموجود فيه الفايل

كود:

```
ExtractFileDrive(F)
```

الحصول على اسم الدرايف

عرف procedure كالتالى مثلا

كود:

```
Procedure GetInfo(F: String);
begin
    WriteLn('F    :', F);
    WriteLn('FileName    :', ExtractFileName(F));
    WriteLn('FileDirPath    :', ExtractFilePath(F));
    WriteLn('FileExt     :', ExtractFileExt(F));
```

```
    WriteLn('FileDir     :', ExtractFileDir(F));
    WriteLn('FileDrive   :', ExtractFileDrive(F));
end;
```

وجرب تباصي ليه حاجة زى كدا

c:\lazarus\bin\laz.exe

كود:

```
   GetInfo('c:\lazarus\bin\laz.exe');
```

اقتباس:

#output
F: 'c:\lazarus\bin\laz.exe'
FileName :laz.exe
FileDirPath :c:\lazarus\bin\
FileExt :.exe
FileDir :c:\lazarus\bin
FileDrive :c:

تعالى للكود

كود:

```
program Project1;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes, SysUtils
  { you can add units after this };

Procedure GetInfo(F: String);
begin
    WriteLn('F    :', F);
    WriteLn('FileName    :', ExtractFileName(F));
    WriteLn('FileDirPath    :', ExtractFilePath(F));
    WriteLn('FileExt     :', ExtractFileExt(F));
    WriteLn('FileDir     :', ExtractFileDir(F));
    WriteLn('FileDrive   :', ExtractFileDrive(F));
end;
var
   F:Text;
   I: Integer;
   Line:String;
begin
   Assign(F, 'textf.txt');
   //Rewrite
   {
```

```
   Rewrite(F);
   For I:=0 To 10 do
       begin
            WriteLn(F,'Hello, World');
       end;
   Close(F);
    WriteLn('Written!');
   }

   //Append
   {
   Append(F);
   WriteLn(F,'a HELLO WORLD LINE!');
   Close(F);
   WriteLn('Appended!');
   }

   //Reading
    {
   Reset(F);
   //Loop till the EOF
   Repeat
         ReadLn(F, Line);
         WriteLn(Line);

   Until EOF(F);


   WriteLn('Done Reading!');
   }

   //Close(F);

   //GetInfo('c:\lazarus\bin\laz.exe');

   ReadLn;

end.
```

طبعا هتحتاج تشيل ال كومينت من على كل سيكشن للتنفيذ

ناقص التعامل مع ال typed files

تقدر تظبطها اكتر من هنا
http://delphi.about.com/od/fileio/Fi...ith_Delphi.htm