

Exercices sur sous-programmes

Exercices sous Lazarus

Eric Thirion - 04/07/2015

Les exercices suivants sont en majorité des projets à compléter dont l'interface graphique est déjà réalisée, ce qui vous permettra un gain de temps important. Ces projets sont disponibles par téléchargement. Pour savoir comment y accéder cliquez [ici](#).

D'autre part, ce document fait partie d'un ensemble de cours du même auteur (programmation objet, programmation web, bases de données) auxquels vous pouvez accéder en cliquant [ici](#).

Exercice 1 : Frais de déplacement

Projet: Exo-Sous-Programme/Pascal/FraisDeplacement/ProjetFraisDepl.lpi

En vous inspirant de l'exemple vu en cours réécrivez le code de ce projet en associant une procédure sans paramètre à chaque traitement intermédiaire du calcul des frais de déplacement.

Pour l'exemple du cours, la version de départ se trouve dans le dossier

Exemple-Premieres-Notions/Peinture

La version avec procédure sans paramètre se trouve dans le dossier

Exemple-Sous-Programme/Peinture

Exercice 2 : Calcul de Moyenne

Projet: Exo-Sous-Programme/Pascal/Moyenne/ProjetMoyenne.lpi

Le formulaire

	Semestre1	Semestre2	Moyenne	
Informatique	<input type="text"/>	<input type="text"/>	<input type="text"/>	Moyenne en Informatique
Mathématiques	<input type="text"/>	<input type="text"/>	<input type="text"/>	Moyenne en Mathématiques
Moyenne	<input type="text"/>	<input type="text"/>	<input type="text"/>	Moyenne des colonnes
Toutes les moyennes en même temps				

Noms des zones de textes (dans le sens de la lecture):

- ZT_Info1, ZT_Info2, ZT_MoyenneInfo
- ZT_Math1, ZT_Math2, ZT_MoyenneMath
- ZT_Moyenne1, ZT_Moyenne2, ZT_MoyenneG

Règle du jeu

Pour toutes les questions de cet exercice, il est interdit d'utiliser une autre variable globale que la variable globale **Moyenne** déjà déclarée.

Question 1 : Déclaration de la procédure CalculerLaMoyenne

Déclarez une procédure, nommée **CalculerLaMoyenne**, avec deux paramètres de type **double**. Cette procédure doit calculer la moyenne des deux paramètres et affecter le résultat à la variable globale **Moyenne**.

Dans les questions suivantes, tous les calculs de moyenne devront se faire en utilisant cette procédure.

Question 2: Calcul de la moyenne en informatique

Lorsque l'utilisateur clique sur le bouton **Moyenne en Informatique** après avoir saisi les deux notes d'informatique, le programme doit afficher la moyenne de ces deux notes dans la zone de texte **ZT_MoyenneInfo**.

Pour obtenir ce résultat, n'écrivez pas directement le code dans la procédure événementielle associée à ce bouton (c'est à dire **BT_MoyenneInfoClick**).

Comme cet exercice porte sur la déclaration et l'appel de procédure, on vous impose ici de programmer en utilisant ces nouveaux concepts.

Vous allez donc tout d'abord écrire une nouvelle procédure, sans paramètres, nommée **MoyenneEnInfo** qui effectuera le traitement souhaité, ainsi que l'instruction d'appel de cette procédure (et rien d'autre !) dans la procédure **BT_MoyenneInfoClick**.

Dans la procédure **MoyenneEnInfo**, utilisez obligatoirement la procédure **CalculerLaMoyenne**.

Indication: dans une procédure non-événementielle, les composants de formulaire doivent être préfixé par **Form1**. Par exemple, la zone de texte **ZT_Info1** s'écrit **Form1.ZT_Info1**.

Question 3: Calcul de la moyenne en mathématique

Même principe pour le bouton **Moyenne en Mathématiques**.

Question 4: Moyenne des colonnes

Dans cette question, on suppose que l'utilisateur à déjà calculé les moyennes en informatique et en mathématiques.

Les zones de texte **ZT_MoyenneInfo** et **ZT_MoyenneMath** contiennent donc ces moyennes.

S'il clique à ce moment là sur le bouton **Moyenne des Colonnes**:

- La moyenne des notes du premier trimestre s'affichera dans la zone de texte **ZT_Moyenne1**.
- La moyenne des notes du second trimestre s'affichera dans la zone de texte **ZT_Moyenne2**.
- La moyenne générale (moyenne des moyennes en informatique et en mathématiques) s'affichera dans la zone de texte **ZT_MoyenneG**.

Encore une fois, on vous impose d'obtenir ce résultat en utilisant des procédures.

Vous allez donc déclarer une nouvelle procédure sans paramètres nommée **MoyenneDesColonnes**, qui effectuera le traitement souhaité ainsi que l'instruction d'appel de cette procédure (et rien d'autre !) dans la procédure **BT_MoyenneColonneClick**.

Dans le corps de la procédure **MoyenneDesColonnes**, tous les calculs de moyenne doivent se faire en utilisant la procédure **CalculerLaMoyenne**.

Question 5: Tous les calculs en un seul clic

Si l'utilisateur clique sur le bouton **Toutes les moyennes en même temps** après avoir saisi les quatre notes, il obtient directement les cinq moyennes:

1. Moyenne en informatique
2. Moyenne en mathématique
3. Moyenne du premier trimestre
4. Moyenne du second trimestre
5. Moyenne générale.

Essayez d'obtenir ce résultat en écrivant une nouvelle procédure sans paramètre nommée **ToutesLesMoyennes**. Le code de cette procédure doit être le plus court possible et réutiliser au maximum ce qui a déjà été fait.

La procédure événementielle **BT_ToutesLesMoyennesClick** doit contenir un appel de cette procédure et rien d'autre.

Question 6: Deuxième version de la procédures ToutesLesMoyennes

Les instructions d'affichage que nous avons utilisés jusqu'à présent sont en fait des appels à des procédures que j'ai écrites. En particulier **AfficherNombre** est une procédure dont le premier paramètre est de type **Double**. Le premier paramètre effectif dans un appel de la procédure **AfficherNombre** peut donc être une expression numérique quelconque.

Par exemple, si **A** et **B** sont de type numérique et **ZT** une zone de texte, l'appel de procédure

`AfficherNombre (A+B, ZT)`

affichera la somme de **A** et **B** dans la zone de texte **ZT**.

En utilisant cette nouvelle possibilité d'appeler la procédure **AfficherNombre**, écrivez une nouvelle version de la procédure **ToutesLesMoyennes**, nommée **ToutesLesMoyennes2**.

Le début du code de cette procédure est déjà écrit. Ajoutez y cinq appels à la procédure **AfficherNombre**, de manière à obtenir le même effet que celui de la procédure **ToutesLesMoyennes**.

Exercice 3 : Frais de déplacement d'une voiture

Notions utilisées: appel de fonctions, variables locales, déclaration de procédures, déclaration de fonctions.

Objectif

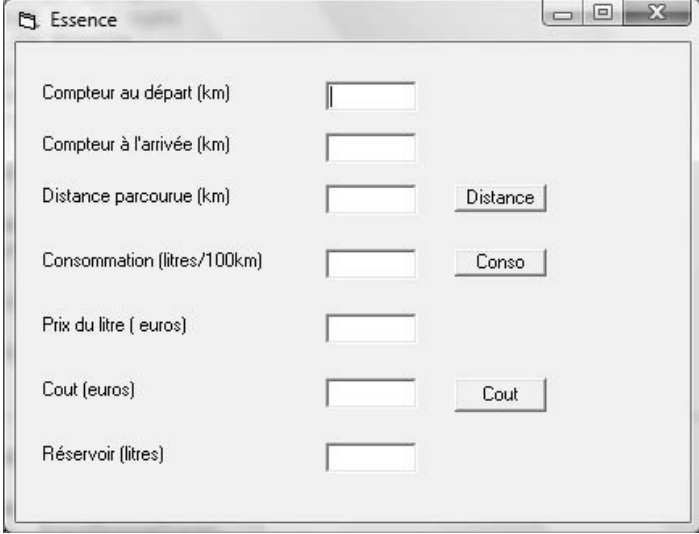
Il s'agit de réaliser un programme en Pascal permettant d'estimer les frais de déplacement d'une voiture à partir de la distance parcourue, sa consommation (nombre de litres pour 100 km) et du prix de l'essence.

Le programme permet également d'estimer la consommation de la voiture à partir de la distance parcourue avec un plein.

Dans l'exercice 1 nous avons déjà réalisé un projet similaire (**FraisDéplacement**) utilisant uniquement des variables globales et des procédures sans paramètres. Dans cette nouvelle version du projet, on vous demande au contraire de n'**utiliser aucune variable globale**.

Notez également que l'interface graphique a changée.

Projet à ouvrir: Exo-Sous-Programme/Pascal/Essence2/ProjetEssence2.lpi

Le formulaire	Zone de textes (de haut en bas)
	<ul style="list-style-type: none"> ● ZoneTexteDep ● ZoneTexteArr ● ZoneTexteDist ● ZoneTexteConso ● ZoneTextePL ● ZoneTexteCout ● ZoneTexteRes

Question1 : Fonctions de calcul

Les calculs de distance , cout et consommation sont effectués par trois fonctions, que l'on vous demande d'écrire :

Nom de la fonction	Paramètres	Résultat
CalculDistance	d : valeur du compteur au départ	Distance parcourue
	a : valeur du compteur à l'arrivée	
CalculCout	d : distance parcourue	Cout du déplacement
	c : consommation (en litres pour 100km)	
	p : prix du litre d'essence	
CalculConso	r : capacité du réservoir	consommation (en litres pour 100km)
	d : distance parcourue	

Question 2 : estimation de la consommation

La consommation de la voiture est estimée à partir de la distance parcourue avec un plein d'essence. Le bouton **Conso** est utilisé à cet effet:

The screenshot shows a window titled 'Essence' with the following fields and buttons:

- Compteur au départ (km):
- Compteur à l'arrivée (km):
- Distance parcourue (km):
- Consommation (litres/100km):
- Prix du litre (euros):
- Cout (euros):
- Réservoir (litres):

Ecrivez la procédure événementielle associée à ce bouton en utilisant la fonction **CalculConso**.

Question 3 : calcul de la distance

Lorsque l'utilisateur clique sur le bouton **Distance** après avoir saisi les valeurs du compteur au départ et à l'arrivée, la distance parcourue est affichée:

The screenshot shows the 'Essence' window with the following values:

- Compteur au départ (km):
- Compteur à l'arrivée (km):
- Distance parcourue (km):
- Consommation (litres/100km):
- Prix du litre (euros):
- Cout (euros):
- Réservoir (litres):

Le code de la procédure événementielle associée à ce bouton est le suivant:

```
procedure TForm1.BoutonDistClick(Sender: TObject);
begin
  DistanceViaCompteur ( ) ;
end;
```

Compléter le code de la procédure **DistanceViaCompteur** en utilisant la fonction **CalculDistance** de manière à obtenir le résultat souhaité.

Question 4 : Calcul du cout

Le calcul du coût de déplacement se fait en cliquant sur le bouton **Cout** après avoir saisi les valeurs du compteur, la consommation et le prix du litre:

La distance parcourue est calculée à partir des valeurs du compteur.

Donnez le code de la procédure événementielle associée au bouton **Cout** en faisant obligatoirement appel à la fonction **CalculCout** ainsi qu'à la la procédure **DistanceViaCompteur**.

Exercice 4: Conjugaison

Objectif: en utilisant les fonctions sur les chaînes de caractères, réalisez un programme permettant de conjuguer un verbe du premier groupe (imparfait, présent et futur) à partir de l'infinitif de ce verbe.

Projet à ouvrir: Exo-Sous-Programme/Pascal/Conjugateur/ProjetConjugateur.lpi

Le formulaire

Nom des contrôles (dans le sens de la lecture):

- ZT_Verbe
- ZL_Imparfait , ZL_Présent , ZL_Futur

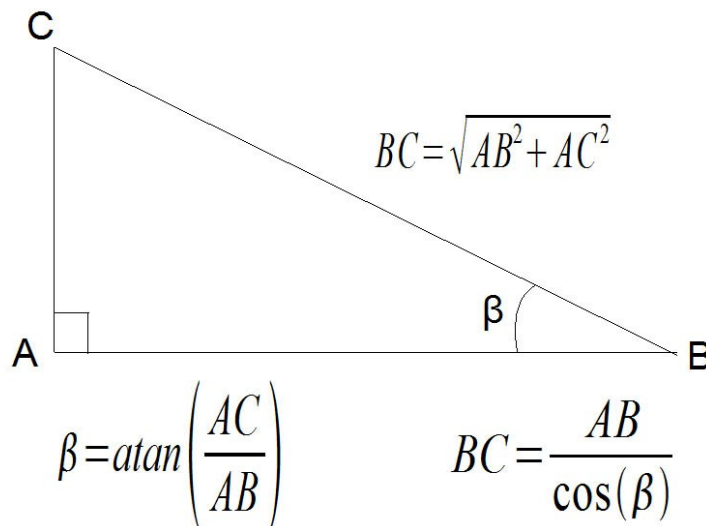
Exercice 5: Random pythagore

Objectif: utiliser les fonctions mathématiques pour calculer l'hypothénuse d'un triangle **ABC** rectangle en **A** dont les dimensions sont générées aléatoirement.

Projet à ouvrir: Exo-Sous-Programme/Pascal/RandomPythagore/ProjetRandomPythagore.lpi

Le formulaire

Noms des zones de texte (dans le sens de la lecture): ZT_AB, ZT_AC, ZT_Angle, ZT_BC.

Formules mathématiques utiles**Question 1 - Générer AB et AC**

En utilisant la fonction **Random**, générez aléatoirement un nombre entier entre 1 et 10 et affectez le à une variable **AB** de type **Double**.

Faites de même pour **AC**, puis affichez les valeurs de ces deux cotés.

Question 2 - Calculer BC par Pythagore

Calculer **BC** en utilisant le théorème de Pythagore et affichez sa valeur arrondie au dixième (fonction **RoundTo**).

Question 3 - Calculer l'angle

Calculer l'angle β (en radian) et affichez sa valeur en degrés arrondie au dixième. Pour convertir un angle exprimé en radians en degrés, il faut le multiplier par $180 / \pi$. Pour π , vous pouvez utiliser la constante PI déjà déclarée.

Question 4 - Calculer BC par l'angle

Calculer BC en utilisant l'angle β .

Exercice 6 : Etat civil

Notions utilisées : variables locales, déclaration de procédure, fonctions prédéfinies

Projet à ouvrir : Etudiant/Pascal/Naissance/ProjetNaissance.lpi

Le formulaire

Nom des contrôles (dans le sens de la lecture)

- ZT_Jour , ZT_Prenom
- ZT_Mois , ZT_Nom
- ZT_Annee , ZT_Lieu
- ZL

Question 1 - La procédure AjouterNaissance

Ajoutez une procédure nommée **AjouterNaissance** au projet avec les paramètres suivants:

- Jour: de type **Integer**
- Mois: de type **String**
- Annee: de type **Integer**
- Prenom: de type **String**
- Nom: de type **String**
- Lieu: de type **String**

Cette procédure doit afficher la date de naissance sur une seule ligne, comme dans l'exemple ci-dessus.

On utilisera une variable locale de type chaîne de caractères pour construire cette ligne avant de l'afficher.

La procédure sera appelée dans **FormCreate** avec les valeurs de paramètres que vous choisirez vous même.

Question 2 - Procédure RemplirZoneTexte

Ajoutez une procédure nommée **RemplirZoneTexte** au projet avec les mêmes paramètres que la procédure précédente, mais cette fois-ci il s'agit d'afficher leurs valeurs dans les zones de texte. On appellera cette procédure dans **FormCreate**, juste après l'appel de la procédure **AjouterNaissance**.

Voilà par exemple l'aspect du programme lorsque l'appel de la procédure **AjouterNaissance** se fait avec *Pierre Simon Laplace né à Baumont le 5 Aout 1749*.

Question 3 - le bouton Ajouter

Complétez la procédure événementielle associée au bouton **Ajouter** afin qu'un clic sur ce bouton transfère toutes les informations contenues dans les zones de textes dans une ligne de la zone de liste.

Par exemple, si l'utilisateur clique sur **Ajouter** lorsque le programme se présente comme à la figure précédente, il devrait obtenir ceci:

Règles du jeu: pour écrire ce code on utilisera **obligatoirement** :

- les variables locales déjà déclarées dans cette procédure.
- un appel de procédure judicieusement choisi.

Exercice 7 : Calcul de vitesse, durées et distances

Notions utilisées: appel de fonctions, variables locales, déclaration de procédures, déclaration de fonctions.

Objectif

Il s'agit d'écrire un programme permettant des calculs de vitesse, de durée et de distance en utilisant les deux représentation du temps définies ci-dessous.

Représentation du temps

Le temps est représenté de deux manières dans le projet:

- En **Heure/Minutes** : par exemple 2H15. Une durée exprimée en Heure/Minutes comporte deux parties: la **partie heure** et la **partie minutes**. Par exemple, la partie heure de 2H15 est 2 et la partie minutes 15.
- En **Minutes**: c'est le nombre de minutes contenues dans la durée. Par exemple: $2H15 = 2 \times 60 + 15$ minutes = 135 minutes

L'heure de départ (ou d'arrivée) est le temps écoulé depuis 0H.

Pour simplifier, on supposera que le départ et l'arrivée ont lieu dans la même journée.

Projet à ouvrir: Exo-Sous-Programme/Pascal/Vitesse/ProjetVitesse.lpi

Le formulaire

Noms des zones de texte (dans le sens de la lecture):

- ZT_Heure_Dep , ZT_Min_Dep
- ZT_Heure_Arr, ZT_Min_Arr
- ZT_Heure_Dur, ZT_Min_Dur
- ZT_Distance
- ZT_Vitesse

Question 1: Conversion entre les deux représentations

Dans le formulaire, le temps est représenté en **Heure/Minutes**. Par contre, pour effectuer les calculs (durée, vitesse, etc) on utilisera la représentation en minutes. On aura donc besoin de fonctions de conversion permettant de passer d'une représentation à l'autre.

Dans cette première question, on vous demande de compléter le code des fonctions suivantes :

- **HeureMinuteEnMinute**: cette fonction permet de convertir une durée exprimée en Heure/Minutes en une durée exprimée en minutes. Elle possède deux paramètres de type entier: la partie heure et la partie minute de la durée. Le résultat retourné par la fonction est le nombre de minutes. C'est donc également un entier.
- **PartieHeures** et **PartieMinutes** ces deux fonctions permettent de convertir une durée exprimée en minutes en une durée exprimée en Heure/minutes. Plus précisément:
 - **PartieHeures** donne la partie heure. Elle possède un paramètre de type **Integer**: la durée exprimée en minute. Le résultat est la partie heure de cette durée. Il s'agit donc également d'un entier. Pour l'obtenir, on calculera la partie entière (voir fonction **Floor**) de la durée divisée par 60.
 - **PartieMinutes** donne la partie minutes. Elle possède le même paramètre que la fonction **PartieHeures** et retourne également un résultat de type **Integer**, qui représente cette fois-ci le nombre de minutes. A vous de trouver la formule mathématique permettant de calculer ce résultat.

Question 2: Calcul de la durée

La procédure événementielle associée au bouton **Durée** est déjà écrite. Voici son code:

```

procedure TForm1.BT_DureeClick(Sender: TObject);
var depart, arrivee, duree : integer;
begin

  Depart := LireDepart();
  Arrivee := LireArrivee();
  Duree := Arrivee - Depart;
  AfficherDuree (Duree);

end;

```

Ce code contient des appels de sous-programmes qui ne sont pas encore complètement écrits:

- **LireDepart**: fonction sans paramètres qui lit l'heure de départ depuis les deux zones de textes prévues à cet effet. Cette fonction retourne l'heure de départ en minutes.
- **LireArrivee**: idem pour l'heure d'arrivée.
- **AfficherDuree**: procédure affichant la durée en Heure/Minutes dans les deux zones de textes prévues à cet effet. Elle possède un paramètre: la durée en minutes.

Il vous reste donc à compléter ces trois sous-programmes de manière à faire fonctionner le bouton **Durée**.

Question 3: Calcul de l'heure de départ et d'arrivée

En vous inspirant de la procédure **BoutonDuree_Click**, écrivez les procédures événementielles associées aux boutons **Départ** et **Arrivée** sachant que:

- Le bouton **Départ** donne l'heure de départ à partir de l'heure d'arrivée et de la durée.
- Le bouton **Arrivée** donne l'heure d'arrivée à partir de l'heure de départ et de la durée.

On vous impose d'écrire ce code en utilisant les sous-programmes suivants:

- **LireDuree**, pour lire la durée.
- **AfficherDepart**, pour afficher l'heure de départ.
- **AfficherArrivee**, pour afficher l'heure d'arrivée.

Question 4: Calcul de la vitesse

Le bouton **Vitesse** permet de calculer la vitesse en km/h à partir de la durée (donnée en Heures/Minutes) et de la distance (donnée en km). Le code de la procédure événementielle associée à ce bouton est déjà écrit. Le voici:

```

procedure TForm1.BT_VitesseClick(Sender: TObject);
var duree: integer; vitesse, distance : double;
begin

    Distance := LireDistance();
    Duree := LireDuree();
    Vitesse := CalculVitesse(Distance, Duree) ;
    AfficherVitesse (Vitesse );

end;

```

Ce code contient des appels de sous-programmes qui ne sont pas encore totalement écrits et que l'on vous demande de compléter:

- **LireDistance**: lit la distance donnée par l'utilisateur.
- **CalculVitesse**: fonction à deux paramètres. Le premier est la distance en km. Le deuxième est la durée en minutes. Le résultat retourné est la vitesse en km/h.
- **AfficherVitesse**: procédure affichant la vitesse passée en paramètre dans la zone de texte prévue à cet effet. On arrondira le résultat au dixième en utilisant la fonction mathématique **RoundTo**.

Question5: Calcul de la distance

Le bouton **Distance** permet de calculer la distance parcourue (en km) à partir de la vitesse (en km/h) et de la durée (donnée en Heures/Minutes).

Ecrivez le code de la procédure événementielle associée à ce bouton, en utilisant impérativement les sous-programmes suivants :

- **LireDuree**, pour lire la durée.
 - **LireVitesse**: fonction permettant de lire la vitesse depuis la zone de texte associée.
 - **CalculDistance**: fonction à deux paramètres. Le premier est la durée en minutes. Le deuxième est la vitesse en km/h. Le résultat retourné est la distance en km.
 - **AfficherDistance**: procédure (à compléter) affichant la distance dans la zone de texte associée.
-