Exercices sur les Conditionnelles

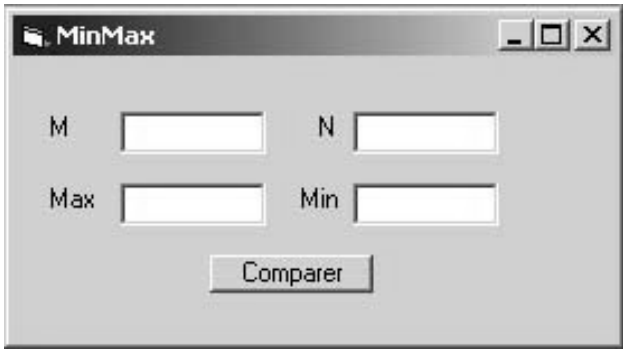
E. Thirion - 03/07/2015

Les exercices suivants sont en majorité des projets à complèter. L'interface graphique de ces projets 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 procédurale, programmation objet, programmation web, bases de données) auxquels vous pouvez accédez en cliquant ici.

Exercice 1: Min et Max

Ouvrir le projet: Exo-StructControle/Pascal/MinMax/ProjetMinMax.lpi

Formulaire	Nom des contrôles
	• ZT_M, ZT_N • ZT_Max, ZT_Min

Lorsque l'utilisateur clique sur le bouton **Comparer**, le plus grand des deux nombres **M** et **N** doit s'afficher dans le champ de texte libellé **Max** et le plus petit, dans le champ de texte libellé **Min**.

Organisez votre code selon le schéma :

1. Lecture des données dans des variables locales,
2. Traitement : détermination de la plus petite et de la plus grande valeur. On affectera la plus petite valeurs à une variable locale et la plus grande à une autre. Ce sont les résultats du traitement.
3. Affichage des résultats.

Exercice 2: Pile ou Face

Ouvrir le projet: Exo-StructControle/Pascal/PileOuFace/ProjetPileOuFace.lpi

Objectif: simuler un jeu de pile ou face. L'ordinateur "lance" une pièce de monnaie virtuelle que l'utilisateur ne voit pas. L'utilisateur indique si la pièce est tombée du coté pile ou du coté face en cliquant sur le bouton correspondant.

Formulaire	Noms des contrôles
	• ZT_Resultat • ZT_NbrLancer • ZT_Score • ZT_Test

Question1: Simulation de la pièce de monnaie

On simulera la pièce de monnaie par une fonction **Piece** (sans paramètres) retournant aléatoirement une des deux chaines de caractères 'Pile' ou 'Face'.

On utilisera pour cela la fonction **Random (n)**. Rappelez vous que cette fonction retourne un nombre au hasard entre 0 (compris) et n (exclu). L'appel de fonction **Random (2)** retournera donc 0 ou 1.

Dès que votre fonction **Piece** est écrite, vous pourrez la tester à l'aide du bouton **Test**. Le code de la procédure évènementielle associée à ce bouton est le suivant:

```
procedure TForm1.BT_TestClick(Sender: TObject);
begin
  Afficher (Piece(), ZT_Test);
end;
```

Si votre fonction est bien écrite, chaque clic sur le bouton **Test** provoquera donc un affichage aléatoire de 'Pile' ou 'Face'.

Question2: gagné ou perdu ?

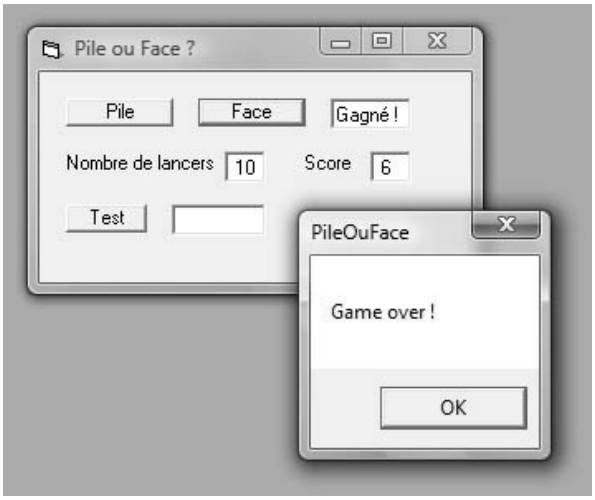
Ecrire les codes associés aux boutons **Pile** et **Face** de manière à afficher **Gagné** ou **Perdu** (dans la zone de texte située à droite de ces deux boutons) selon le résultat du lancer de la pièce virtuelle.

Question3: score et nombre de lancers

Trouvez un moyen d'afficher le score (nombre de succès) et le nombre de lancers dans les zones de texte prévues à cet effet.

Question4: limitation du nombre de lancers

Trouvez un moyen de limiter le jeu à un certain nombre de lancers. Représentez ce nombre maximum de lancer par une constante. Si l'utilisateur tente de dépasser cette limite, l'ordinateur affiche le message "Game over" par l'intermédiaire d'une boite de dialogue (dans cet exemple, la constante est fixé à 10) :

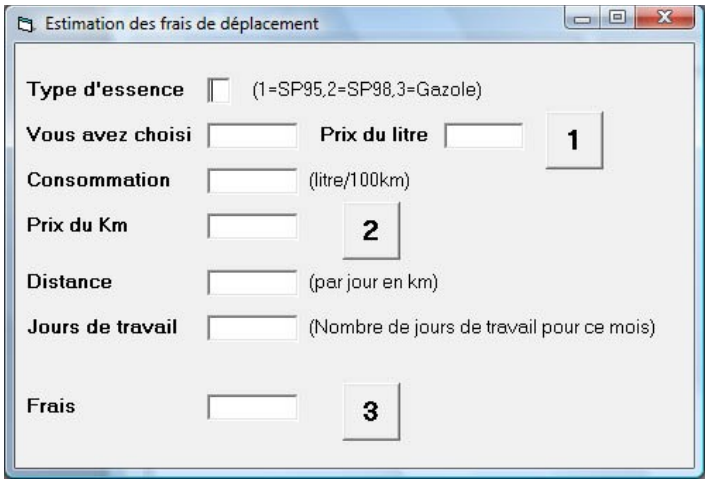


Exercice 3: Frais mensuels

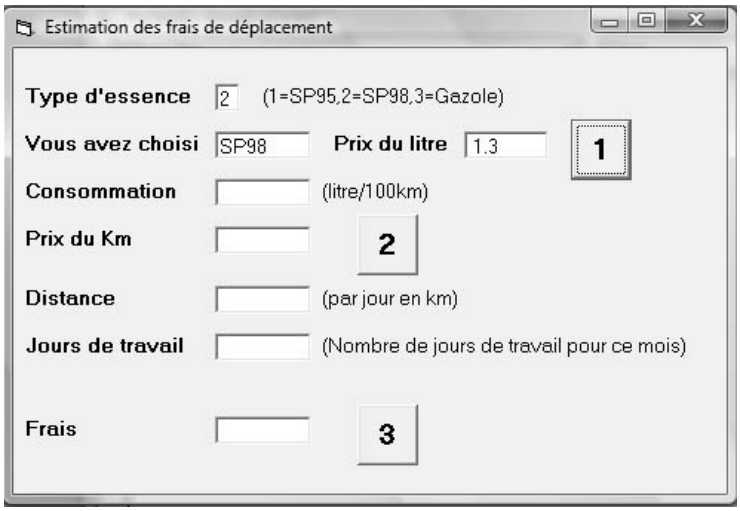
Objectif: Estimer les frais mensuels de déplacements d'un salarié en fonction de différents paramètres.

Règles du jeu: les variables globales déclarées sont imposées. Vous devez nécessairement les utiliser sans changer ni leur noms, ni leur types.

Ouvrir le projet: Exo-StructControle/Pascal/FraisMensuel/ProjetFraisMensuel.lpi

Formulaire	Noms des zones de texte
	• ZT_Code • ZT_NomEssence, ZT_PrixLitre • ZT_Conso • ZT_PrixKm • ZT_Dist • ZT_Jour • ZT_Frais

Question 1: Prix du litre selon le type d'essence



L'utilisateur saisi tout d'abord un code pour définir le type d'essence qu'il utilise pour sa voiture: 1 pour du super 95, 2 pour du super 98 et 3 pour du gazole. Les prix des carburants sont de :

- 1.25 € / litre pour du super 95
- 1.30 € / litre pour du super 98
- 1.05 € / litre pour du gazole.

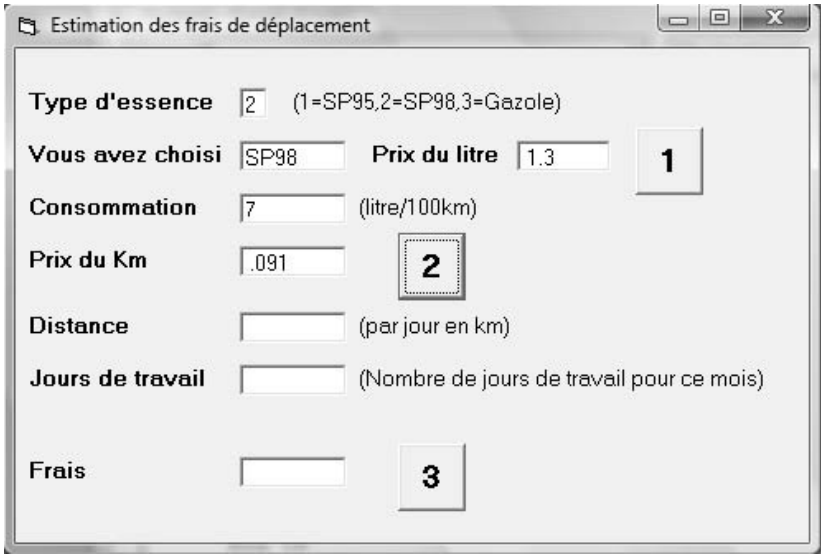
Après avoir entré le code, il clique sur le bouton 1 et le logiciel affiche le type d'essence (SP95 pour du super 95, SP98 pour du super 95 et Gazole pour du gazole) ainsi que le prix du litre pour ce type d'essence.

Dans cette première question, on vous demande d'écrire la procédure évenementielle associée au bouton 1, en utilisant toutes les variables suivantes et aucune autre:

- **CodeEssence** de type **Integer**. Code essence donnée par l'utilisateur.
- **NomEssence** de type **String**. Nom du type d'essence affiché par le logiciel.
- **PrixLitre** de type **Double**. Prix du litre affiché par le logiciel.

Question 2 : Prix du kilomètre

Après avoir saisi le type d'essence, puis appuyé sur le bouton 1, l'utilisateur donne la consommation de sa voiture en nombre de litres pour 100 kilomètres. Lorsqu'il clique sur le bouton 2, le logiciel affiche le prix du kilomètre en euros :



2-A Fonction de calcul

Pour cette question, vous écrirez tout d'abord une fonction pour calculer le prix du kilomètre à partir des paramètres suivants (attention: **n'utilisez pas les variables globales !**) :

- prix du litre.
- consommation de la voiture.

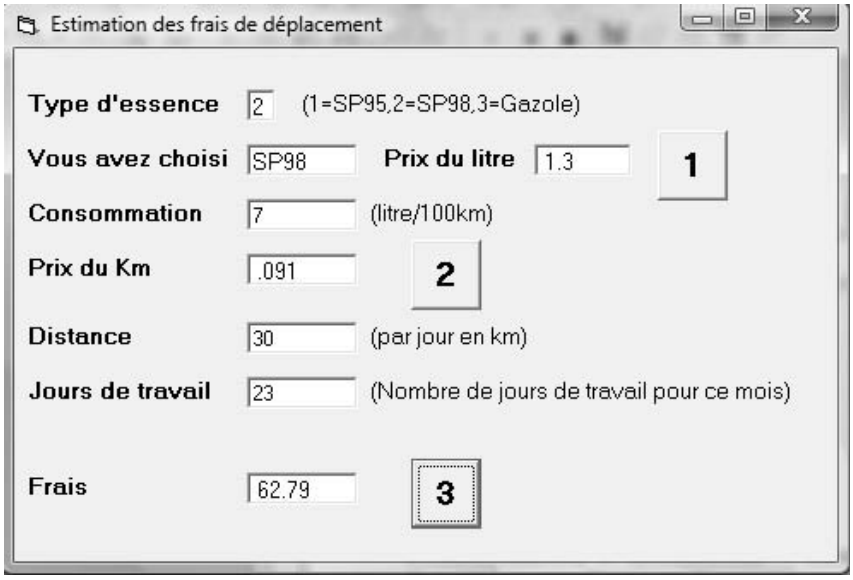
2-B Code du bouton 2

Ecrivez la procédure évènementielle associée au bouton 2, en utilisant la fonction précédente ainsi que les variables suivantes et aucune autre:

- **Consommation** de type **Double**. Consommation de la voiture exprimée en nombre de litres pour 100 kilomètres.
- **PrixKilomètre** de type **Double**. Prix du kilomètre exprimé en euros.
- **PrixLitre** de type **Double**. Prix du litre affiché par le logiciel.

Question 3 : Frais mensuels

Finalement, l'utilisateur donne la distance qu'il parcourt tous les jours (aller-retour) pour se rendre à son lieu de travail, ainsi que le nombre de jours de travail pour le mois. Il clique sur le bouton 3. Le logiciel affiche alors une estimation de ses frais d'essence pour ce mois:



3-A Fonction de calcul

Ecrirez tout d'abord une fonction pour calculer les frais mensuel de déplacement avec les paramètres suivants:

- distance en kilomètre.
- nombre de jours de travail.
- prix du kilomètre

3-B Code du bouton 3

Ecrivez ensuite le code associé au bouton 3 en utilisant la fonction précédente ainsi que les variables globales suivantes:

- **PrixKilometre**
- **Distance**
- **NombreJour**
- **Frais**

Question 4: Protection contre les erreurs de saisie

Des erreurs de saisie peuvent se produire et on souhaiterait protéger le logiciel contre ces erreurs.

Lorsqu'une telle erreur se produit, le programme affichera un message d'erreur et n'effectuera aucun traitement.

On traitera trois types d'erreur (pensez à utiliser l'instruction **exit** !):

1. L'utilisateur entre une valeur numérique abérrante. Par exemple une consommation de 50 litres pour 100 km.

2. Ordre des opérations non respecté (par exemple: clic sur le bouton deux avant d'avoir cliqué sur le bouton 1). Pour traiter ce type d'erreur, vous utiliserez les deux booléens déclarés en variable globale:

    - **PrixLitreConnu**: indique que le prix du litre a été correctement déterminé.
    - **PrixKilometreConnu**: idem pour le prix du kilomètre.

3. L'utilisateur n'a pas saisi toutes les données nécessaires à un traitement. Pour cela, on pourra utiliser la fonction **ZoneDeTexteVide** (zt) de la librairie **ETBib**. Cette fonction permet de tester si une zone de texte passée en paramètre est vide. Si c'est le cas, elle retourne la valeur **True** et sinon **False**.

Exercice 4: Min-Max-Med

Notion de médiane

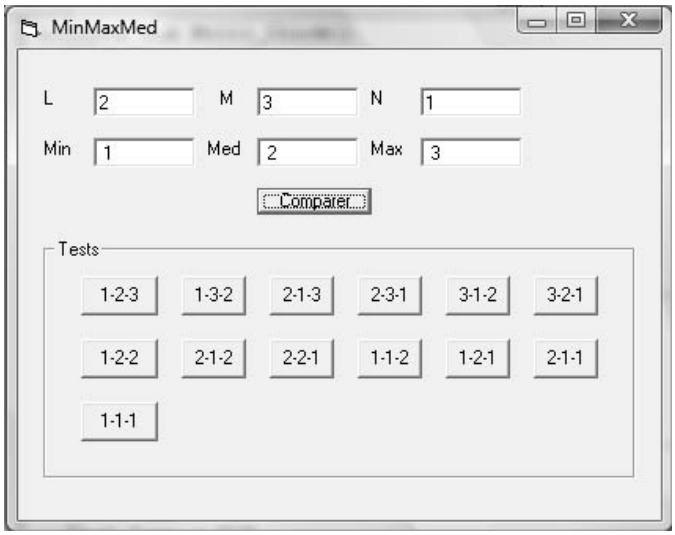
La médiane de trois nombres **L**, **M**, **N** est le nombre qui se trouve au milieu lorsqu'on les range dans l'ordre croissant.

Premier exemple : **L** = **22**, **M** = **1**, **N** = **10**. Si on range ces trois nombres dans l'ordre croissant on obtient 1,10, 22. La médiane est donc 10.

Deuxième exemple : **L = 6**, **M = 61**, **N =** 6. Si on range ces trois nombres dans l'ordre croissant on obtient: 6, 6, 61. La médiane est donc 6.

Objectif: réaliser un programme permettant de calculer le minimum, le maximum et la médiane de trois nombres.

Ouvrir le projet: Exo-StructControle/Pascal/MinMaxMed/ProjetMinMaxMed.lpi

Formulaire	Noms des zones de texte
	• ZT_L • ZT_M • ZT_N • ZT_Min • ZT_Med • ZT_Max

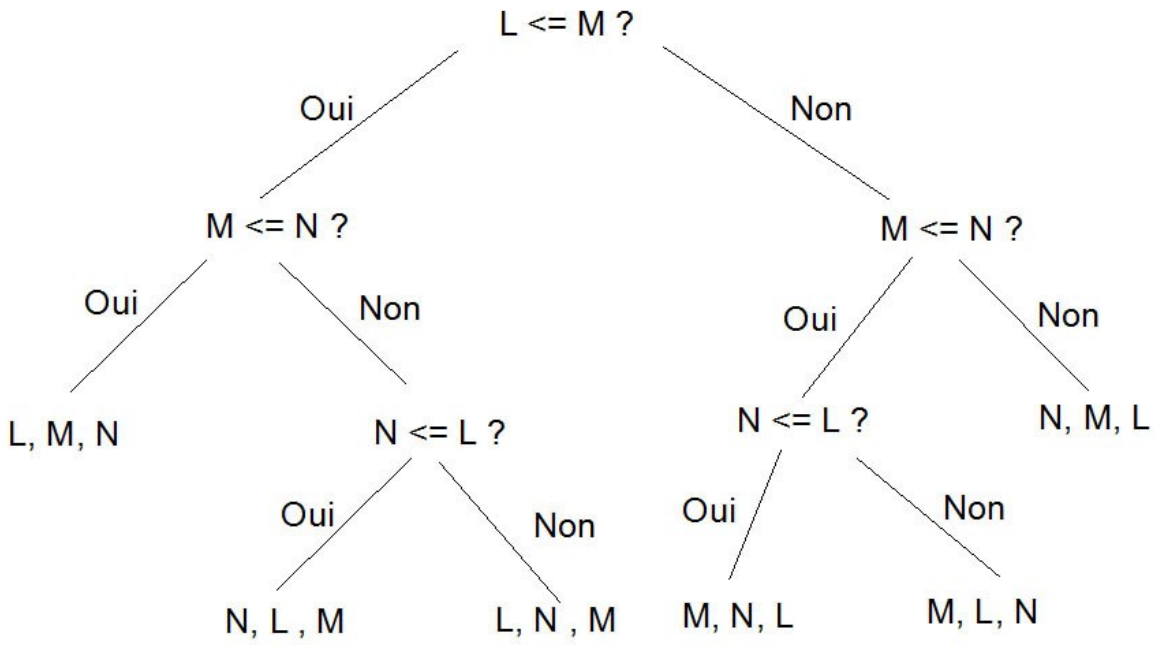
Lorsque l'utilisateur clique sur le bouton **Comparer** après avoir saisi les valeurs de **L**, **M** et **N**, le programme affiche le minimum, la médiane et le maximum de ces trois nombres dans les trois zones de texte prévues à cet effet.

Boutons de test

Pour vous aider à tester rapidement le fonctionnement de votre programme, les boutons 1-2-3, 1-3-2, etc vous permettent de remplir les zones de textes associées aux valeurs de **L**, **M** et **N**. Ces 13 boutons de test représentent tous les cas de figure possibles.

Votre Travail

Ecrire le code de la procédure évènementielle associée au bouton **Comparer** . Le résultat doit être obtenu en effectuant au maximum trois comparaisons. Pour cela vous pouvez vous aider de la figure ci-dessous :



Exercice 5: Machine à café

Objectif: réalisation d'un programme simulant un distributeur automatique de boissons.

Ouvrir le projet: Exo-StructControle/Pascal/MachineACafe/ProjetMachineACafe.lpi

Variables globales

L'état de la machine à un instant donné est représenté par 12 variables globales de type entier qui définissent le nombre de pièces contenues dans la machine, ajoutées par l'utilisateur ou rendues à l'utilisateur:

- **N10M**, **N10A**, **N10R** : nombre de pièces de 10 cts respectivement dans la machine, ajoutées, rendues.
- **N20M**, **N20A**, **N20R** : nombre de pièces de 20 cts respectivement dans la machine, ajoutées, rendues.
- **N50M**, **N50A**, **N50R** : nombre de pièces de 50 cts respectivement dans la machine, ajoutées, rendues.
- **N1M**, **N1A**, **N1R** : nombre de pièces de 1 euro respectivement dans la machine, ajoutées, rendues.

I - Démarrage de la machine

Le démarrage de la machine est simulé par le lancement du programme. Le formulaire apparait alors pour la première fois (voir page 10). Il affiche les prix des boissons, les pièces contenues dans la machine et demande à l'utilisateur d'introduire des pièces, puis de sélectionner la boisson.

Ceci est réalisé par l'appel automatique de la procédure **FormCreate** dont voici le code:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
RemplirLaMachine (3, 2, 1, 0);
AfficherLesPiecesDansLaMachine;
Afficher ('Introduisez des pièces puis sélectionnez la boisson', ZT_Message);
AfficherLesPrix;
end;
```

La procédure **RemplirLaMachine** permet de mettre des pièces dans la machine afin qu'elle puisse rendre la monnaie. Voici son entête:

```
procedure RemplirLaMachine(p10, p20, p50, p1 : Integer);
```

p10 (respectivement **p20**, **p50**, **p1**) représente le nombre de pièces de 10 cts (respectivement 20 cts, 50 cts et un euro) à mettre dans la machine.

AfficherLesPiecesDansLaMachine affiche le nombre de pièces de chaque sorte contenues dans la machine dans les quatre zones de texte prévues à cet effet (**ZT_10M**, **ZT_20M**, **ZT_50M** et **ZT_1M**).

AfficherLesPrix affiche les prix de chaque boisson dans les quatre zones de texte prévues à cet effet (**ZT_Prix1**, **ZT_Prix2**, **ZT_Prix3** et **ZT_Prix4**). Voici son code :

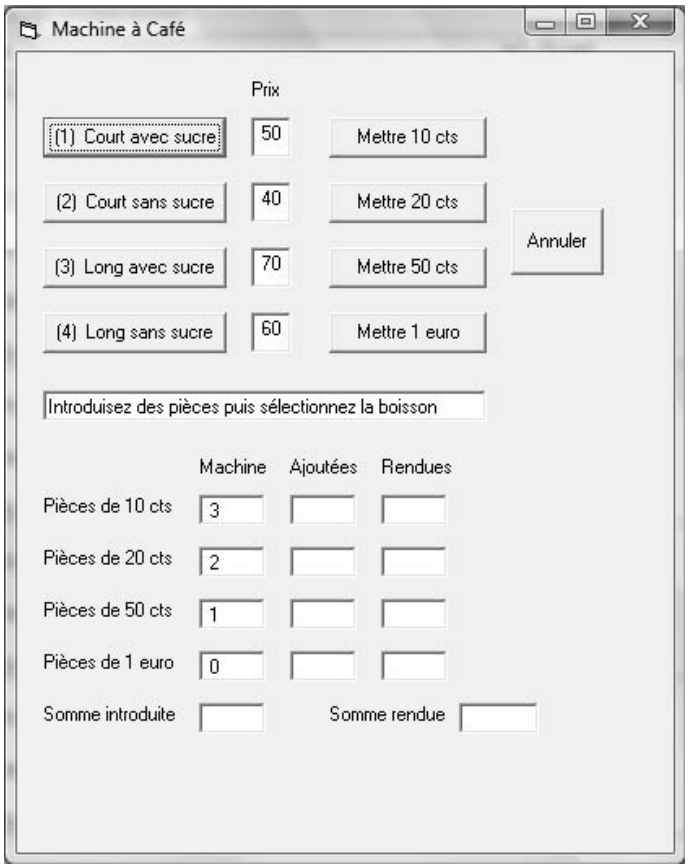
```
Procedure AfficherLesPrix();
begin
  AfficherEntier (PrixDeLaBoisson(1), Form1.ZT_Prix1);
  AfficherEntier (PrixDeLaBoisson(2), Form1.ZT_Prix2);
  AfficherEntier (PrixDeLaBoisson(3), Form1.ZT_Prix3);
  AfficherEntier (PrixDeLaBoisson(4), Form1.ZT_Prix4);
End;
```

elle utilise donc une fonction nommée **PrixDeLaBoisson,** qui retourne le prix d'une boisson de numéro donnée (numéro du bouton de sélection): 50 cts pour la boisson 1, 40 cts pour la boisson 2, 70 cts pour la boisson 3 et 60 cts pour la boisson 4.

Travail à faire:

1. Ecrire le code de la procédure **RemplirLaMachine** en respectant exactement l'entête donnée.
2. Ecrire le code de la procédure **AfficherLesPiècesDansLaMachine.**
3. Ecrire le code de la fonction **PrixDeLaBoisson.**

Le Formulaire



Nom des contrôles

	Prix		
(1) Court avec sucre	**ZT_Prix1**	*Mettre 10 cts*	
(2) Court sans sucre	**ZT_Prix2**	*Mettre 20 cts*	*Annuler*
(3) Long avec sucre	**ZT_Prix3**	*Mettre 50 cts*	
(4) Long sans sucre	**ZT_Prix4**	*Mettre 1 euro*	

ZT_Message

	Machine	*Ajoutées*	*Rendues*	
Pièces de 10 cts	**ZT_10M**	**ZT_10A**	**ZT_10R**	
Pièces de 20 cts	**ZT_20M**	**ZT_20A**	**ZT_20R**	
Pièces de 50 cts	**ZT_50M**	**ZT_50A**	**ZT_50R**	
Pièces de 1 euro	**ZT_1M**	**ZT_1A**	**ZT_1R**	
Somme introduite	**ZT_SommeIntroduite**		*Somme rendue*	**ZT_SommeRendue**

II - Introduction des pièces

II-A Calcul de la somme introduite

La somme introduite par un client peut se calculer en fonction du nombre de pièces 10cts, 20cts, 50 cts, 1 euro qu'il a ajouté.

Travail à faire: complètez la fonction nommée **SommeIntroduite** de manière à ce qu'elle retourne la somme introduite (en cents).

II-B Introduction d'une pièce

Voici le code des procédures évenementielles permettant d'introduire des pièces.

```
procedure TForm1.BT_10CtsClick(Sender: TObject);
begin
  IntroduirePiece (10);
end;
```

```
procedure TForm1.BT_20CtsClick(Sender: TObject);
begin
  IntroduirePiece (20);
end;
```

```
procedure TForm1.BT_50CtsClick(Sender: TObject);
begin
  IntroduirePiece (50);
end;
```

```
procedure TForm1.BT_1euroClick(Sender: TObject);
begin
  IntroduirePiece (100);
end;
```

La procédure **IntroduirePiece** est paramétrisée par la valeur **V** de la pièce introduite dans la machine exprimée en cents. Elle effectue les opérations suivantes:

Si la somme déjà introduite augmentée de la valeur de la pièce est strictement supérieure à un euro, elle affiche un message d'erreur: ***"Inutile d'ajouter des pièces !"*** par une boite de dialogue. Autrement dit la machine refuse une pièce si la nouvelle somme introduite avec cette pièce serait supérieure à un euro.

Dans le cas contraire, elle effectue les opérations suivantes:

- le nombre de pièces de valeur **V** ajoutées par le client est augmenté de 1 et est affiché.
- le nombre de pièces de valeur **V** contenues dans la machine est augmenté de 1 et est affiché.
- la somme introduite est affichée

Travail à faire: complètez le code de la procédure **IntroduirePiece**, en utilisant la fonction **SommeIntroduite**.

III - Annulation

III-A Fonction SommeRendue

Complètez la fonction **SommeRendue** de manière à ce qu'elle retourne la somme rendue par la machine (en cents).

III-B Procédure RendrePiece

Voici le code du bouton d'annulation:

```
procedure TForm1.BT_AnnulerClick(Sender: TObject);
begin
  RejeterLesPieces;
  ShowMessage ('Opération annulée !');
  AnnulerPiecesAjoutees;
  AnnulerPiecesRendues;
end;
```

La procédure **RejeterLesPièces** retourne toutes les pièces ajoutées de la manière suivante:

```
Procedure RejeterLesPieces();
begin
   RendrePiece (N10A, N20A, N50A, N1A);
End;
```

Voici l'entête de la procédure **RendrePiece**:

```
Procedure RendrePiece(p10, p20,p50, p1 :Integer);
```

p10 (respectivement **p20**, **p50**, **p1**) représentent le nombre de pièces de 10 cts (respectivement 20 cts, 50 cts et 1 euro) à rendre.

Les opérations effectuées par cette procédure sont les suivantes:

- le nombre de pièces contenues dans la machine est mis à jour et affiché.
- le nombre de pièces rendues est mis à jour.
- les pièces rendues sont affichées.
- la somme rendue est affichée (on utilisera ici la fonction **SommeRendue**)

Travail à faire: écrire le code de la procédure **RendrePiece**.

IV - Sélection de la boisson et retour de la monnaie

Voici le code des procédures évenementielles associées aux boutons de sélection de boisson:

```
procedure TForm1.BT_CourtAvecSucreClick(Sender: TObject);
begin
  SelectionBoisson (1);
end;
```

```
procedure TForm1.BT_CourtSansSucreClick(Sender: TObject);
begin
   SelectionBoisson (2);
end;
```

```
procedure TForm1.BT_LongAvecSucreClick(Sender: TObject);
begin
   SelectionBoisson (3);
end;
```

```
procedure TForm1.BT_LongSansSucreClick(Sender: TObject);
begin
   SelectionBoisson (4);
end;
```

Elles appellent toutes la procédure **SelectionBoisson** qui prend en paramètre le numéro de la boisson sélectionnée. Voici son code :

```
Procedure SelectionBoisson(n : Integer);
begin

  Afficher ('Vous avez choisi: ' + NomDeLaBoisson(n), Form1.ZT_Message);

  If SommeIntroduite < PrixDeLaBoisson(n) Then
  begin
     ShowMessage ('Ajoutez des pièces ou sélectionnez une autre boisson!');
     Effacer (Form1.ZT_Message)
  end
  Else
  begin
     If SommeIntroduite = PrixDeLaBoisson(n) Then
         ShowMessage ('Café en cours ...')
     Else
     begin
          If PeutRendre(SommeIntroduite - PrixDeLaBoisson(n)) Then
          begin
           Rendre (SommeIntroduite - PrixDeLaBoisson(n));
           ShowMessage ('Café en cours ...');
          end
          Else
          begin
           RejeterLesPieces;
           ShowMessage ('Impossible de rendre la monnaie !')
          End;
        AnnulerPiecesRendues
     end;
     AnnulerPiecesAjoutees;
   End;
End;
```

Les fonctions appelées sont:

Fonction	Code
PrixDeLaBoisson	Déjà écrit
SommeIntroduite	Déja écrit
PeutRendre	Donné, mais utilise des fonctions à complèter
Rendre	Donné, mais utilise des fonctions à complèter
RejeterLesPieces	Donné
AnnulerPiecesRendues	Donné
AnnulerPiecesAjoutees	Donné

IV-A Possibilité de rendre la monnaie

Dans cette question, nous nous interesserons uniquement au cas où le client introduit une somme strictement supérieure au prix de la boisson. Dans ce cas, la machine doit lui rendre la monnaie et cela n'est pas toujours possible ....

La fonction **PeutRendre** permet de déterminer si la machine peut rendre une certaine somme. Comme le client ne peut introduire que un euro au maximum et que la boisson la moins chère est à 40 cents, la somme à rendre varie entre 10 et 60 cents. Voici le code de la fonction **PeutRendre**:

```
Function PeutRendre(SommeDue : Integer): Boolean;
begin
 Case SommeDue of
   10:  PeutRendre := PeutRendre10Cts;
   20:  PeutRendre := PeutRendre20Cts;
   30:  PeutRendre := PeutRendre30Cts;
   40:  PeutRendre := PeutRendre40Cts;
   50:  PeutRendre := PeutRendre50Cts;
   60:  PeutRendre := PeutRendre60Cts;
 Else PeutRendre := False;
 End;
End;
```

Les six fonctions **PeutRendre10Cts**, ..., **PeutRendre60Cts**, permettent de savoir si la machine peut rendre respectivement 10 cents, ..., 60 cents.

Votre travail:

Complètez ces fonctions en utilisant impérativement la fonction **PeutRendrePiece** (à complèter également) dont voici l'entête:

```
Function PeutRendrePiece(p10,  p20 , p50 :integer) : Boolean;
```

Cette fonction retourne la valeur **True** si et seulement si la machine peut rendre **p10** pièces de 10 cents, **p20** pièces de 20 cents et **p50** pièces de 50 cents.

IV-B Retour effectif de la monnaie

La procédure **Rendre** permet de rendre une certaine somme au client. Voici son code:

```
procedure  Rendre(Somme : Integer);
begin
Case Somme of
  10: Rendre10Cts;
  20: Rendre20Cts;
  30: Rendre30Cts;
  40: Rendre40Cts;
  50: Rendre50Cts;
  60: Rendre60Cts;
End;
End;
```

Les procédures **Rendre10Cts**, ..., **Rendre60Cts** permettent de rendre respectivement 10 cents, ..., 60 cents.

Afin de pouvoir rendre la monnaie aussi longtemps que possible, la machine essaie toujours de rendre le moins de pièces possible à l'utilisateur.

Travail à faire: écrire les procédures **Rendre10Cts**, ..., **Rendre60cts** en utilisant la fonction **PeutRendrePiece** et la procédure **RendrePiece** de manière à économiser au maximum le nombre de pièces rendues.