Exercices sur les Boucles

E. Thirion - 04/07/2015

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 et objet, programmation web, bases de données) auxquels vous pouvez accédez en cliquant ici.

Exercice 1: Relevé de compte

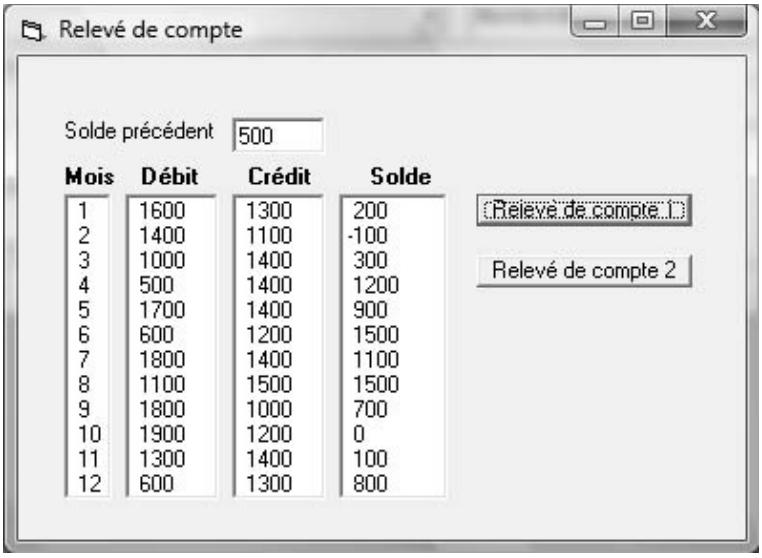
Objectif: Simuler un relevé de compte annuel.

Ouvrir le projet: Exo-StructControle/Pascal/ releveDeCompte/ProjetReleveDeCompte.lpi

Formulaire	Nom des contrôles
	Zone de texte: ZT_SoldePrec Zones de liste de gauche à droite : • ZL_Mois • ZL_Debit • ZLCredit • ZL_Solde

Question1: relevé de compte 1

Le bouton **Relevé de Compte 1** affiche les débits, crédits et solde pour chaque mois de l'année comme suit:



Mois	Débit	Crédit	Solde
1	1600	1300	200
2	1400	1100	-100
3	1000	1400	300
4	500	1400	1200
5	1700	1400	900
6	600	1200	1500
7	1800	1400	1100
8	1100	1500	1500
9	1800	1000	700
10	1900	1200	0
11	1300	1400	100
12	600	1300	800

Le débit de chaque mois est un nombre au hasard parmis les multiples de 100 compris entre 500 et 2000. On l'obtient par l'instruction suivante:

```
Debit := 500 + random (16)* 100;
```

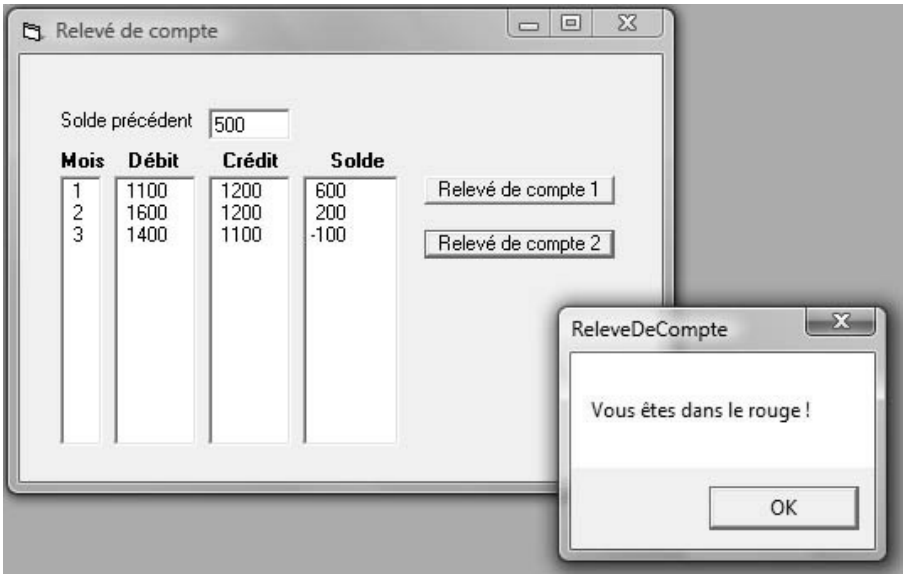
Le crédit de chaque mois qui est un multiple de 100 compris entre 1000 et 1500, que l'on obtient par l'instruction suivante:

```
Credit := 1000 + random (6)* 100;
```

A vous de choisir le type de boucle qui conviendra le mieux pour cette opération.

Question2: relevé de compte 2

Avec le bouton **Relevé de Compte 2** l'affichage s'arrête dès que le solde est négatif et dans ce cas le logiciel affiche un message d'alerte:

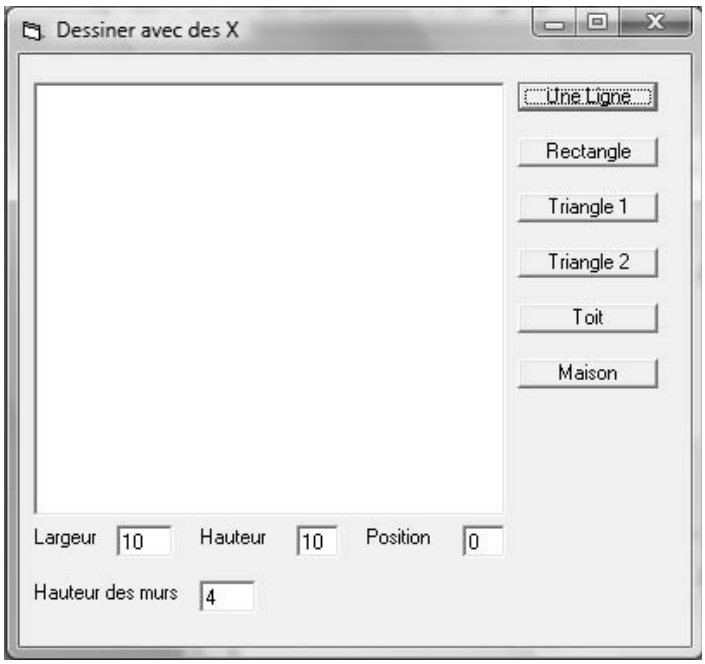


Si le solde n'est jamais négatif, tout se passe comme avec le bouton **Relevé de Compte 1**.

Exercice 2 : Avec des X

Objectif: Dessiner différentes formes à l'aide du caractère X en utilisant des boucles.

Ouvrir le projet: Exo-StructControle/Pascal/AvecDesX/ProjetAvecDesX .lpi

Formulaire	Nom des contrôles
	Zone de liste destinée à afficher les dessins: ZL
	Les zones de texte: ● ZT_Largeur ● ZT_Hauteur ● ZT_Position ● ZT_HauteurMur

Question 1: Une ligne de X (bouton UneLigne)

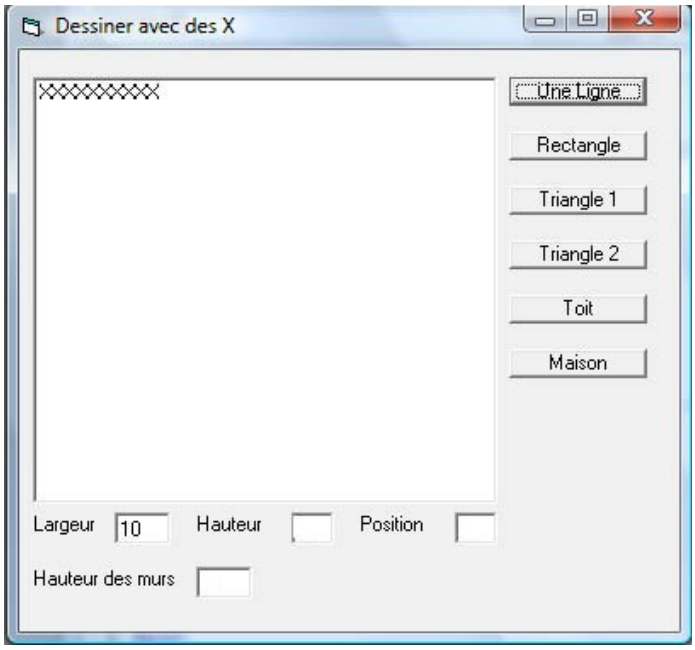
Complètez la fonction RepeterX dont l'entête est la suivante:

```
Function RepeterX( n : Integer): String;
```

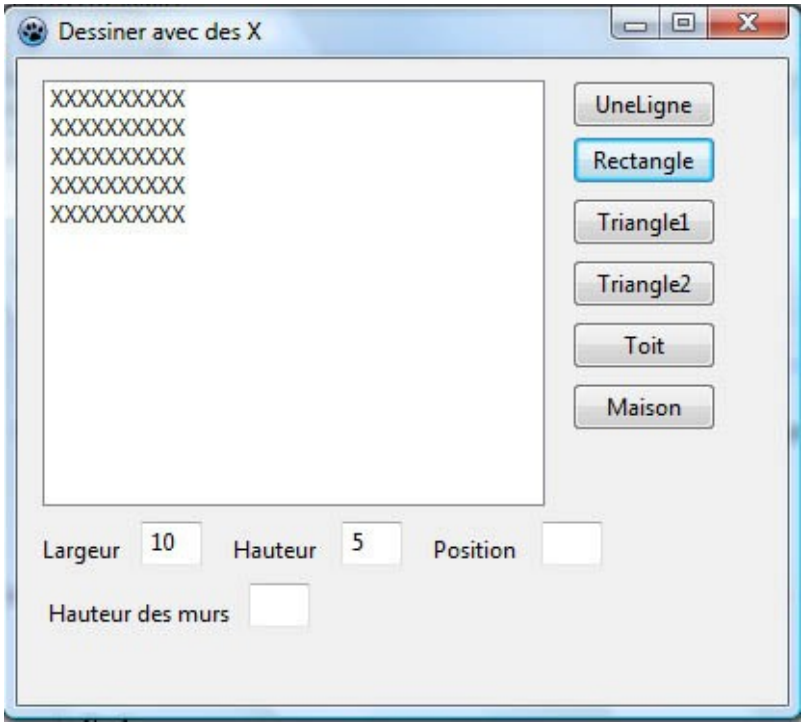
Le résultat retourné est une chaine de caractères constitué de **n** "X".

Indication: utiliser une boucle **for**, en concaténant **n** fois la chaine 'X' à la chaine vide.

Si cette fonction est correctement écrite, le bouton **UneLigne** devrait fonctionner: il affiche dans la zone de liste une ligne de "X" dont la largeur (le nombre de X) est celle-donnée dans la zone de texte libellée **Largeur**:



Question 2: Afficher un rectangle

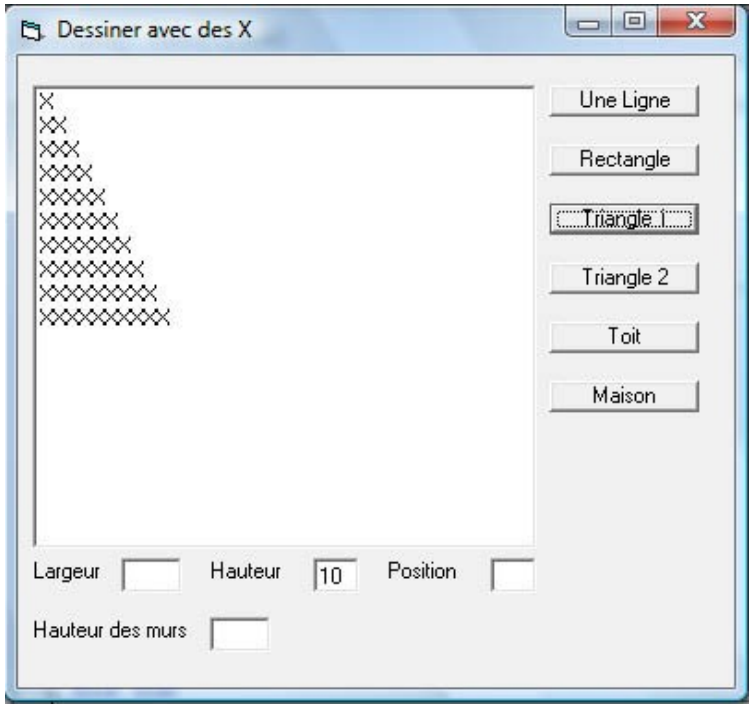


Lorsque l'utilisateur clique sur le bouton **Rectangle**, le programme affiche un rectangle constitué de X, dont les dimensions sont celles données dans les zones de textes étiquetée **Largeur** (nombre de X par ligne) et **Hauteur** (nombre de lignes).

Question 3: Triangle 1

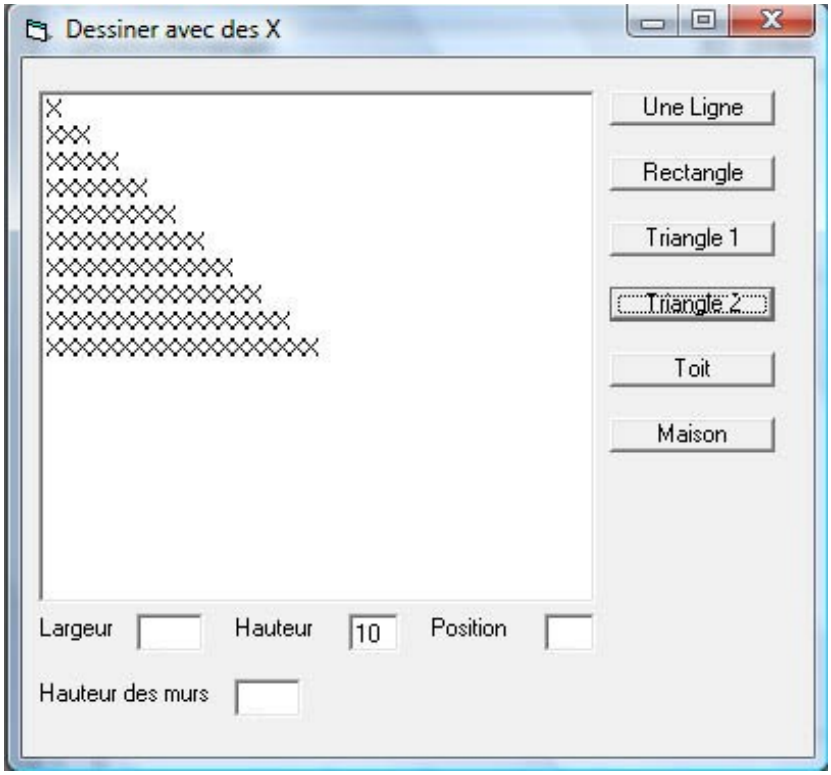
Il s'agit d'afficher un triangle dont la première ligne contient un **X**, la deuxième deux **X**, etc...

Le nombre de lignes du triangle est la valeur de la zone texte étiquetée **Hauteur**:



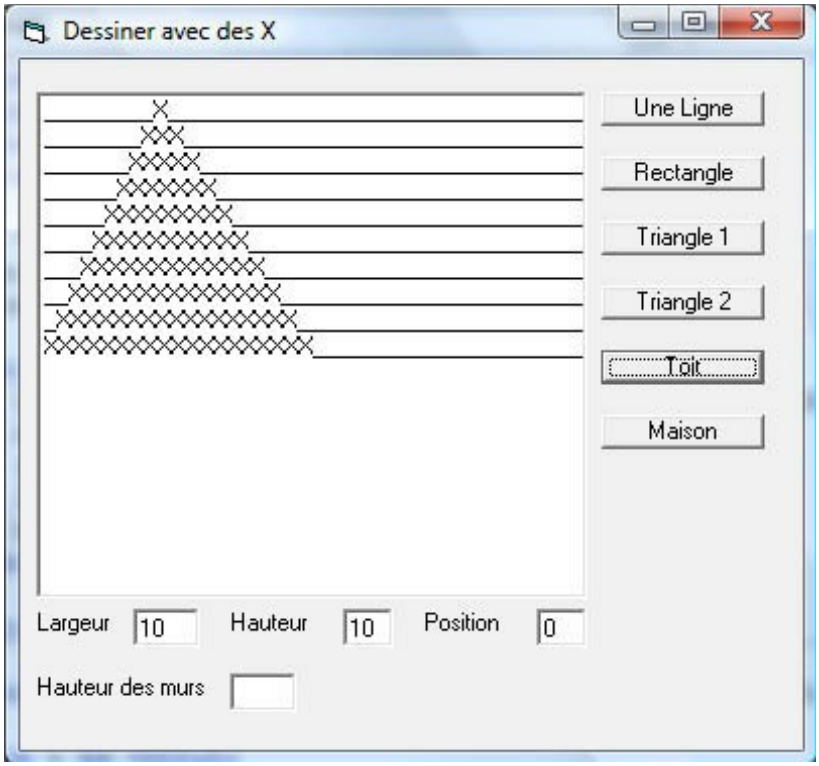
Question 4: Triangle 2

Comme dans la question précédente, sauf que le nombre de **X** augmente de deux d'une ligne à l'autre:



Question 5: Toit de maison

Il s'agit d'afficher une sorte de "toit" de maison constitué de **X** et entourés de traits, comme ceci:



Question 5-A: Affichage des traits

Ecrivez une fonction retournant une chaine de caractères constituée de **n** "_". Cette fonction vous servira à afficher les traits.

Question 5-B: Affichage du toit

Le code du bouton **Toit** est déjà écrit, mais il appelle une procédure qui n'est pas encore écrite. Voici son entête:

```
procedure Toit(Hauteur,  Position : Integer);
```

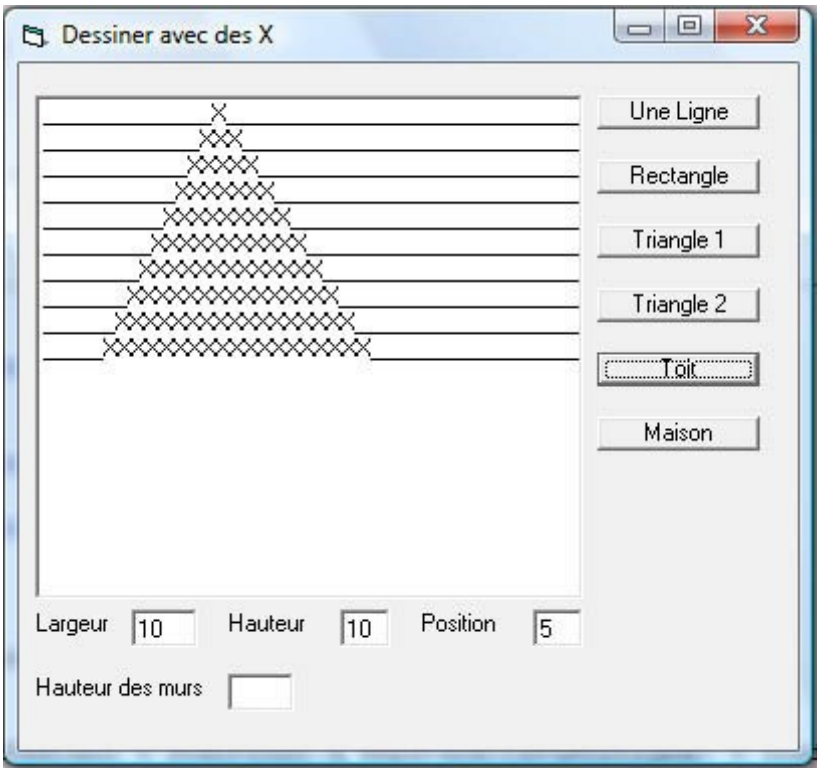
Cette procédure affiche un toit de hauteur donnée (nombre de lignes).

Le paramètre **Position**, permet de règler la position verticale du toit.

Plus précisement **Position** contient le nombre de "_" avant le premier **X** de la dernière ligne.

Donc avec **Position** = 0, le toit est collé à gauche de la zone de liste comme dans l'exemple précédent.

Voici une autre exemple où la position est égale à 5:



Indications :

- analysez comment varient le nombre de X et le nombre de trait (à gauche du toit) lorsque l'on va de la première (haut du toit) à la dernière ligne.
- le nombre de traits à droite du toit n'a pas d'importance. Mettez en suffisament pour remplir la zone de liste.

Question 6: Maison

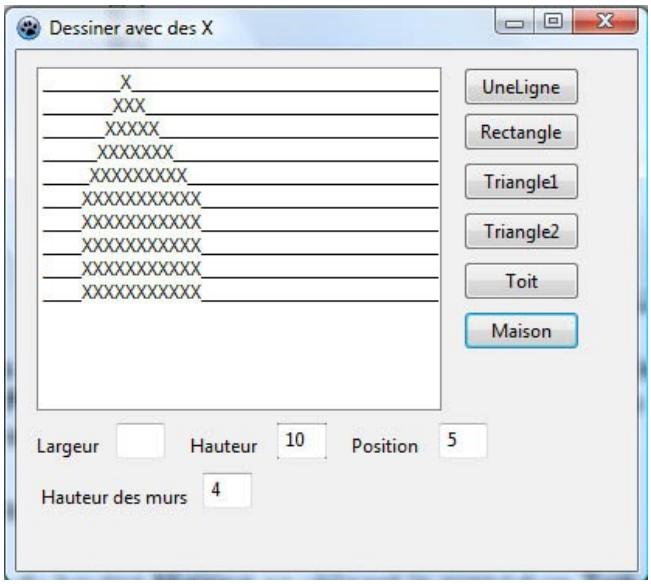
Question 6-A: Murs de la maison

Ecrivez la procédure suivante:

```
procedure Mur(Largeur,Hauteur,Position : Integer);

Affiche un rectangle de largeur et hauteur (nombre de ligne) constitué de X.

Comme dans la question précédente, le mur doit être précédé et suivi de
traits.

La paramètre Position détermine l'écart entre le bord gauche de la zone de
liste et le bord gauche du mur. C'est donc également le nombre de "_" des
traits situé à gauche du mur.
```

Question 6-B: Code du bouton Maison

Le bouton **Maison** permet d'afficher une maison comme suit:



Les paramètres pris en compte ici sont les valeurs des zones de texte étiquetées **Hauteur** (hauteur totale de la maison), **Position** (écart en nombre de caractères entre le bord gauche de la zone de liste et le bord gauche de la maison) et **Hauteur des murs**.

La zone de texte étiquetée **Largeur** n'est donc pas utilisée.

Ecrivez le code du bouton **Maison** en utilisant la procédure **Toit** et la procédure **Mur**.

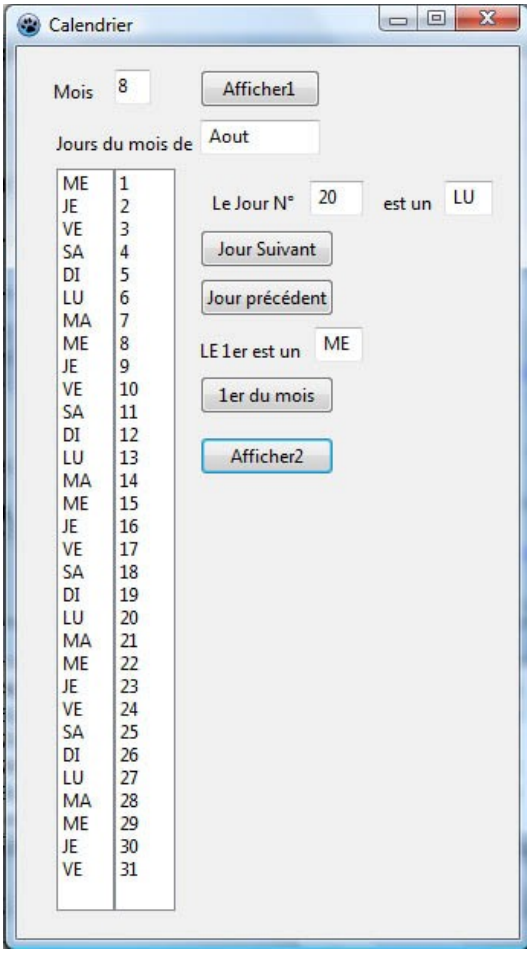
Indications: calculez la hauteur du toit en fonction de la hauteur de la maison et de celle des murs, puis la largeur du toit en fonction de la hauteur du toit.

Attention: n'utilisez pas de variable locale nommée position car cela provoque un problème de conflit de nom (ce nom de variable est déjà utiisé par Lazarus).

Exercice 3: Calendrier

Objectif: Afficher les jours d'un mois quelconque de l'année avec les noms des jours associés.

Ouvrir le projet: Exo-StructControle/Pascal/Calendrier/ProjetCalendrier.lpi

Formulaire	Nom des contrôles
	Zones de texte dans le sens de la lecture: ● ZT_NumMois ● ZT_NomMois ● ZT_NumJour ● ZT_NomJour ● ZT_NomDu1er
	Zones de liste de gauche à droite : ● ZL_NomJour ● ZL_NumJour

Question1: Numéros des jours

Il s'agit ici de faire fonctionner le bouton **Afficher1**. Celui-ci permet d'afficher les numéros des jours d'un mois dans la zone de liste **ZL_NumJour**. Le mois est donné par son numéro (zone de texte **ZT_NumMois**). Le nom du mois correspondant doit s'afficher dans la zone de texte **ZT_NomMois**.

Vous écrirez le code de ce bouton en utilisant deux fonctions que vous devrez tout d'abord complèter:

- **function** NomDuMois (n: **integer**) : **string**;
  Retourne le nom du mois de numéro **n**.

- **function** NombreDeJoursDuMois (n: **integer**): **integer**;
  Retourne le nombre de jours du mois de numéro **n**. Rappel: Janvier, Mars, Mai, Juillet, Aout, Octobre et Décembre ont 31 jours. Les autres mois ont 30 jours sauf Février qui en a 28 (pour simplifier on ignore l'exception des années bissextiles).

Question2: Nom du jour suivant

Le bouton **Jour Suivant** permet d'afficher le numéro et le nom du jour qui suit un jour de numéro et de nom donné dans les zones de texte **ZT_NumJour** et **ZT_NomJour**.

Le nom du jour est représenté par ses deux premières lettres en majuscules : 'LU', 'MA', 'ME', ..., 'DI'.

Exemple: **ZT_NumJour** contient 20 et **ZT_NomJour** contient 'LU'. Après avoir cliqué sur le bouton **Jour Suivant**, **ZT_NumJour** devra contenir 21 et **ZT_NomJour** contient 'MA'.

Pour coder ce bouton vous utiliserez obligatoirement la fonction **NomDuJourSuivant** dont voici l'entête:

```
function NomDuJourSuivant (nj: string) : string;
```

Cette fonction retourne le nom du jour qui suit immédiatement un jour de nom donné.

Attention: la conditionnelle **Case** ne fonctionne pas avec des chaines de caractères !

Question3: Nom du jour précédent

Même chose qu'avant, mais avec le bouton **Jour précédent** et la fonction **NomDuJourPrecedent**.

Question4: Nom du premier jour du mois

Il s'agit ici de faire fonctionner le bouton **1er du mois** qui permet de trouver le nom du premier jour du mois donné, connaissant le nom du jour correspondant à un numéro de jour de ce mois. Ce nom sera affiché dans la zone de texte **ZT_NomDu1er**.

Exemple: **ZT_NumJour** contient 3 et **ZT_NomJour** contient 'VE'. Lorsque l'utilisateur clique sur le bouton **1er du mois**, le programme affichera 'ME'.

Pour coder ce bouton vous utiliserez obligatoirement la fonction **PremierDuMois** que vous devrez tout d'abord complèter. Voici son entête:

```
function PremierDuMois (numj: integer; nomj: string): string;
```

Cette fonction retourne le nom du premier jour du mois, sachant que le jour numéro **numj** se nomme **nomj**.

Exemple: **PremierDuMois (**3, 'VE'**)** retourne 'ME'.

Indication: utilisez une boucle faisant appel à la fonction **NomDuJourPrecedent**.

Question5: Affichage des numéros et des noms des jours du mois

Il s'agit ici de faire fonctionner le bouton **Afficher2** qui permet d'afficher les numéros et les noms des jours d'un mois de numéro donné ainsi que le nom de ce mois.

Les données utilisées ici sont:

- le numéro du mois (zone de texte **ZT_NumMois**).
- le numéro d'un jour de ce mois (zone de texte **ZT_NumJour**).
- le nom de ce jour (zone de texte **ZT_NomJour**).

Indication: utilisez la fonction **PremierDuMois** et une boucle faisant appel à la fonction **NomDuJourSuivant**.

Question6: protection contre les erreurs

Pour protéger votre programme contre les erreurs de l'utilisateur, complètez les fonctions suivantes:

- `function MoisNonValide ( ) : boolean;`

  Si la zone de texte **ZT_NumMois** est vide, elle affiche le message **'***Donnez le numéro du mois !'* et retourne **true**.

  Si le nombre entier contenu dans la zone de texte **ZT_NumMois** n'est pas compris entre 1 et 12, elle affiche le message *'Numéro de mois incorrect !'* et retourne **true**.

  Sinon elle retourne **false**.

- `function NumeroDuJourNonValide (M: integer): boolean;`

  Si la zone de texte **ZT_NumJour** est vide, elle affiche le message *'Donnez le numéro du jour !'* et retourne **true**.

  Si le nombre entier contenu dans la zone de texte **ZT_NumJour** n'est pas compris entre 1 et le nombre de jours du mois M, elle affiche le message *'Numéro de jour incorrect !'* et retourne **true**.

  Sinon elle retourne **false**.

- `function NomDuJourNonValide ( ): boolean;`

  Si la zone de texte **ZT_NomJour** est vide, elle affiche le message *'Donnez le nom du jour !'* et retourne **true**.

  Si la chaine de caractères contenu dans la zone de texte **ZT_NomJour** n'est ni 'LU', ni 'MA', ..., ni 'DI', elle affiche le message *'Nom de jour incorrect !'* et retourne **true**.

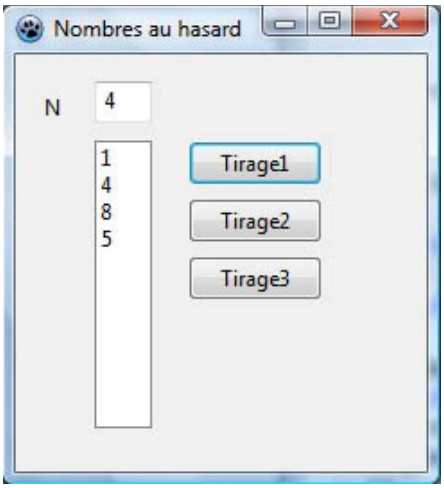
Enfin pour terminer, affichez un message d'erreur lorsque l'utilisateur :

- clique sur le bouton **Jour Suivant** alors que le jour donné est le dernier du mois.
- clique sur le bouton **Jour Précédent** alors que le jour donné est le premier du mois.

Exercice 4: Nombres au hasard

Objectif: Afficher des suites de nombres au hasard vérifiant certaines propriétés imposées.

Ouvrir le projet: Exo-StructControle/Pascal/NombresAuHasard/ProjetNombresAuHasard.lpi

Formulaire	Nom des contrôles
	Zone de texte : ● ZT_N Zone de liste : ● ZL_Nombre

Question1: Tirage1 - Suite de longueur fixe

Lorsque l'utilisateur entre un nombre **n** dans la zone de texte puis clique sur le bouton **Tirage1**, le programme affiche une suite de **n** nombres au hasard compris entre 0 et 9.

Complètez la procédure évènementielle associée au bouton **Tirage1** de manière à obtenir ce résultat.

Rappel: pour un nombre entier **M**, **random (M)** retourne un nombre entier au hasard entre 0 (inclu) et **M** (exclu).

Question2: Tirage2 - Suite se terminant par un 7

Lorsque l'utilisateur clique sur le bouton **Tirage2**, le programme affiche une suite de nombres au hasard compris entre 0 et 9 se terminant par un 7. De plus, la longueur de la suite est affichée dans la zone de texte **ZT_N**.

Complètez la procédure évènementielle associée au bouton **Tirage2** de manière à obtenir ce résultat.

Vous constaterez que la boucle **repeat** est mieux adaptée à ce traitement que la boucle **while**.

Question3: Tirage3 - Suite croissante

Lorsque l'utilisateur clique sur le bouton **Tirage3**, le programme affiche une suite croissante de nombres au hasard compris entre 0 et 9. Cela signifie que tout nombre de la suite est forcément supérieur ou égal au nombre précédent. Comme précédemment, le nombre de terme de la suite est affiché dans la zone de texte **ZT_N**.

Complètez la procédure évènementielle associée au bouton **Tirage3** de manière à obtenir ce résultat.