École de Commerce et d'Informatique

Principes et méthodes de programmation

Test de programmation (novembre 2010) - Cours de Jacques THOORENS

Solutions

1 Vocabulaire (*20 points*)

Ajouter les mots manquants.

.

Le cours de PMP nous a appris que tout programme commence par la déclaration des *variables* qui permettent d'accéder à des zones de *mémoire* où ranger des valeurs. Le corps même du programme comporte des *instructions* parfois regroupées dans des structures *alternatives* ou des *boucles*. L'écriture de ces dernières est l'aspect le plus délicat : avec un peu de maladresse, on écrit facilement une boucle qui engendre une infinité d'*itérations*. Nous avons vu que le *moteur* était une instruction particulière qui nous assurait que la boucle allait se terminer : cette instruction nous donne l'espoir que la *condition* finisse par devenir fausse. Nous savons aussi malheureusement que l'*exécution* d'un programme ne permet pas d'en démontrer la justesse. L'écriture d'un *algorithme* nécessite de l'astuce et du talent. Ce n'est pas demain que les ordinateurs en seront capables.

Des exemples d'exécution des programmes demandés figurent à la fin des questions.

2 PGCD (*30 points*)

Nous avons vu dans les exercices du laboratoire une méthode astucieuse pour calculer le PGCD de deux nombres par de simples soustractions. Il en existe une autre que je vous demande de programmer. Vous devez donc déterminer le Plus Grand Commun Diviseur de deux nombres entiers entrés au clavier en utilisant cette nouvelle méthode.

Soit deux nombres N1 et N2, pris dans un ordre quelconque. On remplace (**prudemment**) N1 par N2 et N2 par N1 modulo N2. Lorsque le reste de la division est nul, on a trouvé la valeur du PGCD.

Exemple

N1	N2	N1	N2	N1	N2
32	14	36	28	17	15
14	32 mod 14 -> 4	28	36 mod 28 -> 8	15	17 mod 15 -> 2
4	14 mod 4 -> 2	8	28 mod 8 -> 4	2	15 mod 2 -> 1
2	**4 mod 2 -> 0**	**4**	**8 mod 4 -> 0**	**1**	**2 mod 1 -> 0**

Solution

```
Variables entières iNombre1, iNombre2
Variables entières iN1, iN2, iTampon

Début du traitement

   Afficher "Entrez un premier nombre "
```

```
   Lire iNombre1
   Afficher "Entrez un second nombre "
   Lire iNombre2

   /* Boucle : */
   Initialisation
      iN1 <- iNombre1
      iN2 <- iNombre2
   Tant que
      iN2 > 0
   Répéter
      iTampon <- iN1
      iN1 <- iN2
      Moteur iN2 <- (iTampon % iN2)
   Fin de boucle

   /* Alternative : */
   Si iN1>0 Alors
      Afficher "Le PGCD de " & iNombre1 & " et "
      & iNombre2 & " vaut " & iN1
      /* */
   Sinon
      Afficher "Le PGCD de deux nombres nuls n'existe pas"
   Fin de si
   Attente

Fin du traitement
```

3 Programme au choix (*40 points*)

Réaliser **un** des programmes suivants au choix.

3.1 Changement de mot de passe

Lorsqu'un utilisateur de notre système décide de changer de mots de passe, les choses se passent de la manière suivante :

- il propose un nouveau mot de passe
- le programme vérifie que ce mot de passe n'a pas été employé récemment, pour l'obliger à trouver un mot de passe original
- si le mot de passe est nouveau, on lui demande de le confirmer pour éviter les erreurs

Notons que, tant que l'utilisateur n'a pas satisfait aux deux conditions, on lui demande de recommencer. Il est évidemment inutile de demander confirmation pour un mot de passe qui va être refusé. Dans un programme réel, on demanderait d'abord le mot de passe actuel avant de réaliser le changement, mais cette phase peut être omise ici.

Conseils :

- on imagine que vous avez la possibilité de récupérer les anciens mots de passe par une méthode magique que vous pouvez placer dans votre programme là où vous l'estimez

utile. Ces anciens mots de passe (supposons qu'on en a mémorisé 10) doivent évidemment être mémorisés par le moyen que vous voudrez (on suppose que la méthode magique est au courant de votre technique).

– un programme réel devrait évidemment mémoriser ce nouveau mot de passe, mais vous pouvez vous arrêter juste avant.

Solution

```
/* Je me limite à trois mots de passe mémorisés */
Constante MAX = 3
Tableau chaines sTAnciensMots taille MAX
<* = {"obelix","asterix","panoramix"} *>
Variables chaines sMotdePasse, sConfirmation
Variable indice iI

Début du traitement

   /* La méthode magique employée est ici une initialisation
    lors de la déclaration */

   /* Boucle : */
   Initialisation

   Répéter
      Afficher "Entrez le nouveau mot de passe "
      Moteur lire sMotdePasse
      /* Boucle : */
      Initialisation
         iI <- 0
      Tant que
         iI < MAX et sMotdePasse <> sTAnciensMots[iI]
      Répéter

         Moteur incrémenter iI

      Fin de boucle
      si iI < MAX alors
         Afficher "Désolé, ce mot de passe a été récemment utilisé"
         sConfirmation <- ""
      sinon
         Afficher "Veuillez confirmer le mot de passe: "
         Lire sConfirmation
         si sConfirmation <> sMotdePasse alors
            afficher "Vous avez entré deux mots de passe différents"
         fin de si
      fin de si
      fin de ligne

   Boucler tant que sConfirmation <> sMotdePasse
   Afficher "Mémorisation du mot de passe"
   Attente
```

3.2 Sommes et produits

Il s'agit de calculer et de mémoriser la somme et le produit des N premiers nombres entiers, pour des valeurs de N allant de 1 à 10. Le programme se réalisera en deux parties :

1. calcul des sommes et des produits
2. affichage des résultats

Remarque :

ce programme va bien entendu utiliser des tableaux. Il est possible, mais non nécessaire, d'utiliser des tableaux à deux dimensions.

Solution

Ma solution travaille ici avec un tableau à deux dimensions. Pour une solution avec des tableaux à une seule dimension voir ci-après.

```
Constante MAX = 10
Constante SOMME = 0
Constante PRODUIT = 1
Tableau entiers iTableau taille 2:MAX
Variable indice iI

Début du traitement

   /* Remplissage */

   /* Boucle : */
   Initialisation
      iTableau[SOMME,0] <- 1
      iTableau[PRODUIT,0] <- 1
   Compter avec iI de 1 à MAX-1
      iTableau[SOMME,iI] <- iTableau[SOMME,iI-1]+iI+1
      iTableau[PRODUIT,iI] <- iTableau[PRODUIT,iI-1]*(iI+1)
   Fin de compter

   /* Boucle : */
   Afficher "Sommes : \n"
   Initialisation
   Compter avec iI de 0 à MAX-1
      afficher iTableau[SOMME,iI] & '-'
   Fin de compter
   fin de ligne

   /* Boucle : */
   Afficher "Produits : \n"
   Initialisation
   Compter avec iI de 0 à MAX-1
      afficher iTableau[PRODUIT,iI] & '-'
```

```
   Fin de compter
   fin de ligne

   attente


   /* Affichage */
Fin du traitement
```

Version avec tableau à une seule dimension.

```
Constante MAX = 10
Tableau entiers iTProduits taille MAX
Tableau entiers iTSommes taille MAX
Variable indice iI
```

Il suffit de remplacer `iTableau[SOMME,??]` par `iTSommes[??]` et `iTableau[PRODUIT,???]` par `iTProduits[??]`.

4 Justification (*20 points*)