Algorithmique

CHAPITRE 5 : LES TESTS

Sommaire

1. Définitions

- Supposons qu'on veuille afficher un message « Mot de passe correct » ou « Mot de passe incorrect » selon la saisie de l'utilisateur. On aura comme raisonnement :
  - Demander à l'utilisateur de saisir son mot de passe
  - Si le mot de passe saisi est identique au mot de passe enregistré alors on affiche « Mot de passe correct »
  - Sinon on affiche « Mot de passe incorrect »
- La sortie du message dépend ici de la saisie de l'utilisateur, pour afficher le message correct, on effectue un **test** sur le mot de passe que l'utilisateur aura saisi.
- On affiche le message « Mot de passe incorrect », quand le mot de passe saisi par l'utilisateur ne correspond pas au mot de passe enregistré. Cette action est appelée **structure alternative**.
- Un test donne comme résultat une valeur booléenne **VRAI** ou **FAUX.**
- Il peut arriver par exemple d'avoir des tests à l'intérieur d'un test exemple:
  - Si le mot de passe de l'utilisateur est correct alors, si l'utilisateur est administrateur, afficher « Bonjour Administrateur » sinon afficher « Bonjour utilisateur »
  - Sinon, si le mot de passe est inférieur à 8 caractères, afficher « Mot de passe court » sinon afficher « Mot de passe incorrect »

2. Structure et condition d'un test

- Un test n'a que deux structures possibles la première qui est une structure **sans alternative** et l'autre **avec alternative**.

- Pour le **test sans alternative,** on exécute juste une instruction lorsque le test est vrai. Par exemple dans le cas du mot de passe on aurait comme test sans alternative :
  - Demander à l'utilisateur de saisir son mot de passe
  - Si le mot de passe saisi est identique au mot de passe enregistré, afficher « Mot de passe correct »

- Dans l'exemple précédent on a aucune alternative lorsque l'utilisateur saisit un mot de passe incorrect, on a aucune instruction à exécuter.

- Pour le **test avec alternative,** on exécute une instruction lorsque le test est vrai ou lorsqu'il est faux. Pour le mot de passe on aura :
  - Demander à l'utilisateur de saisir son mot de passe
  - Si le mot de passe saisi est identique au mot de passe enregistré afficher « Mot de passe correct »
  - Sinon afficher « Mot de passe incorrect »

- Avec le test avec alternative, on affiche un message quelque soit la saisie de l'utilisateur contrairement au test sans alternative où un message est affiché uniquement en cas de mot de passe correct.

- Il peut arriver des fois d'avoir dans un test un autre test, on parle de **test imbriqué**.

2. Structure et condition d'un test

- Un test permet de vérifier si une condition est oui ou non réalisée, une condition est en fait une comparaison.
- Prenons l'exemple du mot de passe, l'ordinateur est déverrouillé si et seulement si la condition mot de passe saisie est identique au mot de passe enregistré.
- Une condition est une comparaison renvoyant un résultat booléen **VRAI** ou **FAUX.**
- En pseudo code pour faire un **test sans alternative,** nous utiliserons l'instruction suivante :

```
SI condition ALORS
    ....
FINSI

Ou

Si condition ALORS
    ...
FinSi
```

- Pour un **test avec alternative,** nous utiliserons l'instruction suivante :

```
SI condition ALORS
    ...
SINON
    ...
FINSI
```

3. Exemple de test avec condition

Soit un ordinateur dont le mot de passe est « ORODeIRIestROIdeRIO », écrire un algorithme qui demande le mot de passe à l'utilisateur. Si le mot de passe saisi est différent du mot de passe enregistré, afficher « Mot de passe incorrect » sinon afficher « Mot de passe correct »

```
Début
    Variables secret, motDePasse en Chaînes
    secret <- "ORODeIRIestROIdeRIO"
    Ecrire "Entrer le mot de passe:"
    Lire motDePasse
    Si motDePasse = secret Alors
        Ecrire "Mot de passe correct"
    Sinon
        Ecrire "Mot de passe incorrect"
    FinSi
Fin
```

```
Début
    Variables secret, motDePasse en Chaînes
    secret <- "ORODeIRIestROIdeRIO"
    Ecrire "Entrer le mot de passe:"
    Lire motDePasse
    Si Non (motDePasse = secret) Alors
        Ecrire "Mot de passe incorrect"
    Sinon
        Ecrire "Mot de passe correct"
    FinSi
Fin
```

4. Conditions composées

- La condition d'un test comme vue précédemment peut être constituée d'opérateurs logiques.
- Une condition étant un expression qui renvoie soit **VRAI** ou **FAUX**, on peut retrouver des conditions comme :
  - (A = B) OU (B = C)
  - (A > B) ET (B < C)
  - (A < B) XOR (B < C)
- Il peut arriver dans un algorithme qu'on ait besoin de faire un test booléen **différent de**, dans ce cas on utilisera pour ce cours le symbole <> pour vérifier si une valeur est différente d'une autre.
  - Par exemple **Non(2 = 5)** peut s'écrire **2 <> 5** et cela revient au même résultat **VRAI.**
  - Dans le cas du mot de passe, on pouvait utiliser la condition **secret <> motDePasse**, cela aurait donné l'algorithme suivant :

```
Début
    Variables secret, motDePasse en Chaînes
    secret <- "ORODeIRIestROIdeRIO"
    Ecrire "Entrer le mot de passe:"
    Lire motDePasse
    Si motDePasse <> secret Alors
        Ecrire "Mot de passe incorrect"
    Sinon
        Ecrire "Mot de passe correct"
    FinSi
Fin
```

5. Exemple de test avec conditions composées

Soit un ordinateur dont le mot de passe est « doé » et le nom d'utilisateur est « john », un utilisateur peut se connecter s'il entre le bon mot de passe et le bon nom d'utilisateur. Si les informations entrées sont correctes, afficher « Autorisé à se connecter » sinon afficher « Interdit de se connecter ».

```
Début
    Variables motDePasseSaisi, motDePasse, pseudo, pseudoSaisi en Chaînes
    pseudo <- "john"
    motDePasse <- "doé"
    Ecrire "Entrer votre nom d'utilisateur:"
    Lire pseudoSaisi
    Ecrire "Entrer votre mot de passe:"
    Lire motDePasseSaisi
    Si (pseudoSaisi <> pseudo) Ou (motDePasse <> motDePasseSaisi) ALORS
        Ecrire "Interdit de se connecter"
    Sinon
        Ecrire "Autorisé à se connecter"
    FinSi
Fin
```

6. Tests imbriqués

- Un test imbriqué est un test qui en contient un autre, c'est une suite de tests.

- Nous avons deux méthodes pour effectuer des tests imbriqués. En pseudo code nous utiliserons cette notation :

```
Si condition Alors
    Si condition Alors
        ...
    Sinon
     ...
    FinSi
Sinon
    Si condition Alors
        ...
    Sinon
        ...
    FinSi
FinSi
```

```
Si condition Alors
    ...
SinonSi condition Alors
    ...
SinonSi condition Alors
    ...
Sinon
    ...
FinSi
```

Avec les imbrications, il faut être très rigoureux dans l'indentation du code pour qu'il soit lisible. Une condition « Si » doit toujours se terminer par un « FinSi ». Il doit y avoir autant de « FinSi » que de « Si ».

7. Exemple de test imbriqué

Soit un ordinateur dont le mot de passe est « doé » et le nom d'utilisateur est « john », demander à l'utilisateur son nom d'utilisateur et après son mot de passe. Si le nom d'utilisateur et le mot de passe sont incorrects, afficher « Nom d'utilisateur et mot de passe incorrects », sinon si c'est uniquement le mot de passe, afficher « Mot de passe incorrect », sinon si c'est uniquement le nom d'utilisateur afficher « Nom d'utilisateur incorrect ». Sinon si les identifiants sont corrects afficher « Utilisateur autorisé ».

```
Début
    Variables motDePasseSaisi, motDePasse, pseudo, pseudoSaisi en Chaînes
    pseudo <- "john"
    motDePasse <- "doé"
    Ecrire "Entrer votre nom d'utilisateur:"
    Lire pseudoSaisi
    Ecrire "Entrer votre mot de passe:"
    Lire motDePasseSaisi
    Si Non (pseudoSaisi = pseudo ) ET Non (motDePasseSaisi = motDePasse) Alors
        Ecrire "Nom d'utilisateur et mot de passe incorrects"
    SinonSi (pseudoSaisi <> pseudo) OU (motDePasseSaisi <> motDePasse) Alors
            Si pseudo <> pseudo Alors
                Ecrire "Nom d'utilisateur incorrect"
            Sinon
                Ecrire "Mot de passe incorrect"
            FinSi
    Sinon
        Ecrire "Utilisateur autorisé"
    FinSi
Fin
```

8. Exercices

Exercice 1

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif. On prendra en compte le cas où le nombre est nul.

Exercice 2

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif ou nul.

Exercice 3

Ecrire un algorithme qui demande trois nombres à l'utilisateur et l'informe ensuite s'ils sont rangés ou non en ordre croissant.

Exercice 4

Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie: « Poussin » de 6 à 7 ans, « Pupille » de 8 à 9 ans, « Minime » de 10 à 11 ans et « Cadet » après 12 ans.

9. Corrections

Exercice 1

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif. On prendra en compte le cas où le nombre est nul.

```
Début
    Variable nbre en Réel
    Ecrire "Entrer un nombre:"
    Lire nbre
    Si nbre > 0 Alors
        Ecrire nbre + " est un nombre positif"
    SinonSi nbre < 0 Alors
        Ecrire nbre + " est un nombre négatif"
    Sinon
        Ecrire nbre + " est nul"
    FinSi
Fin
```

9. Corrections

Exercice 2

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif ou nul.

```
Début
    Variables nbre1, nbre2 en Réels
    Ecrire "Entrer nombre 1:"
    Lire nbre1
    Ecrire "Entrer nombre 2:"
    Lire nbre2
    Si (nbre1 * nbre 2) < 0 Alors
        Ecrire "Le produit de " + nbre1 + " par " + nbre2 + " est négatif."
    SinonSi (nbre1 * nbre2) > 0 Alors
        Ecrire "Le produit de " + nbre1 + " par " + nbre2 + " est positif."
    Sinon
        Ecrire "Le produit de " + nbre1 + " par " + nbre2 + " est nul."
    FinSi
Fin
```

9. Corrections

Exercice 3

Ecrire un algorithme qui demande trois nombres à l'utilisateur et l'informe ensuite s'ils sont rangés ou non en ordre croissant.

```
DEBUT
    VARIABLES nbre1, nbre2, nbre3 en REELS
    ECRIRE "Entrer nombre 1:"
    LIRE nbre1
    ECRIRE "Entrer nombre 2:"
    LIRE nbre2
    ECRIRE "Entrer nombre 3:"
    LIRE nbre3
    SI nbre < nbre2 ET nbre2 < nbre3 ALORS
        ECRIRE "Nombres rangés en ordre croissant"
    SINON
        ECRIRE "Nombres non rangés en ordre croissant"
    FINSI
FIN
```

9. Corrections

Exercice 4

Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie: « Poussin » de 6 à 7 ans, « Pupille » de 8 à 9 ans, « Minime » de 10 à 11 ans et « Cadet » après 12 ans.

```
Début
    Variables age en Entier
    Ecrire "Entrer votre âge svp:"
    Lire age
    Si (age = 6 ) Ou (age = 7) Alors
        Ecrire "Vous êtes Poussin"
    SinonSi (age = 8) Ou (age = 9) Alors
        Ecrire "Vous êtes Pupille"
    SinonSi (age = 10) Ou (age = 11) Alors
        Ecrire "Vous êtes Minime"
    SinonSi age > 12 Alors
        Ecrire "Vous êtes Cadet"
    Sinon
        Ecrire "Catégorie inconnue"
    FinSi
Fin
```

Citations

« Il est assez difficile de trouver une erreur dans son code quand on la cherche. C'est encore bien plus dur quand on est convaincu que le code est juste. » Steve McConnell

« Il n'existe pas, et il n'existera jamais, de langage dans lequel il soit un tant soit peu difficile d'écrire de mauvais programmes. » Anonyme

« Si le débogage est l'art d'enlever les bogues, alors la programmation doit être l'art de les créer. » Anonyme