Chapitre 1 : Instructions algorithmiques de base

Laurent Debize

BTS SIO 1A - Algorithmique appliquée

Quatrième partie IV

Notion de variable

Différents types de variables

- Numérique : nombres entiers, réels...
- Chaîne / caractère
- Booléen (vrai ou faux)
- Date

Manipulation des chaînes

```
programme conversion
   choix : caractère
debut
   //Boucle sur le menu
   choix <- "Z"

   // afficher le menu
   tantque choix <> "Q" et choix <> "q"
      afficher "conversion entier vers binaire ..... 1"
      afficher "conversion binaire vers entier ..... 2"
      afficher "quitter ............................. Q"
      saisir choix

     // conversion entier vers binaire
      si choix = "1" alors
         ...
      sinon

         // conversion binaire vers entier
         si choix = "2" alors
            ...
         finsi
      finsi
   fintantque
fin
```

Conversion entier vers binaire

```
binaire <- ""
afficher "entrer un entier = "
saisir nb
tantque nb <> 0
   binaire <- str(nb mod 2) + binaire
   nb <- nb div 2
fintantque
afficher "conversion en binaire = " + binaire
```

Conversion binaire vers entier

```
nb <- 0
k <- 0
afficher "entrer un nombre binaire = "
saisir binaire
tantque taille(binaire) > 0
   nb <- nb + val(droite(binaire, 1)) * 2^k
   binaire <- gauche(binaire, taille(binaire)-1)
   k <- k + 1
fintantque
afficher "conversion en base 10 = " + nb
```

L'algorithme complet

```
programme conversion
   choix : caractère
    binaire : chaîne
   nb : nombre réel
   k : entier
debut
   //Boucle sur le menu
   choix <- "Z"
   tantque choix <> "Q" et choix <> "q"

      // afficher le menu
      afficher "conversion entier vers binaire ..... 1"
      afficher "conversion binaire vers entier ..... 2"
      afficher "quitter ............................. Q"
      saisir choix

      // conversion entier vers binaire
      si choix = "1" alors
         binaire <- ""
         afficher "entrer un entier = "
         saisir nb
         tantque nb <> 0
            binaire <- str(nb mod 2) + binaire
            nb <- nb div 2
         fintantque
         afficher "conversion en binaire = " + binaire
      sinon

         // conversion binaire vers entier
         si choix = "2" alors
            nb <- 0
            k <- 0
            afficher "entrer un nombre binaire = "
            saisir binaire
            tantque taille(binaire) > 0
               nb <- nb + val(droite(binaire, 1)) * 2^k
               binaire <- gauche(binaire, taille(binaire)-1)
               k <- k + 1
            fintantque
            afficher "conversion en base 10 = " + nb
         finsi
      finsi
   fintantque
fin
```

Exercice 1

Avec l'aide du mémento, transcrire cet algorithme en C++ pour le tester.

Transtypage

- **str(uneValeurNumerique)** : retourne une chaîne correspondant à la valeur numérique, ou le contenu de la variable numérique, mise en paramètre.
  Exemple :

  ```
  uneVariableChaine <- str(3)
  ```

- **val(uneChaine)** : retourne une valeur numérique correspondant à la chaîne, ou le contenu de la variable de type chaîne, mise en paramètre..
  Exemple :

  ```
  uneVariableNumerique <- val("3")
  ```

Opérations sur les chaînes

Concaténation

```
ch1 <- "hello"
ch2 <- "world"
ch3 <- ch1 + " " + ch2 + " !"
afficher ch3 // on obtient "hello world !"
```

Opérations sur les chaînes

Taille

```
ch1 <- "hello world !"
afficher taille(ch1) // on obtient 13
```

Opérations sur les chaînes

Extraction

- **extraire(uneChaine, depart, longueur)** : retourne la chaîne extraite de une-Chaine, à partir de la position depart et d'un nombre de caractères correspondant à longueur. On part du principe que l'indice du premier caractère est 0 (le démarrage peut se faire à 1 suivant les algos : cette information est normalement précisée).

```
ch1 <- "hello world !"
ch2 <- extraire(ch1, 6, 5)
afficher ch2 // on obtient "world"
```

- **extraire(uneChaine, depart)** : retourne la chaîne extraite de uneChaine, à partir de la position depart et jusqu'à la fin.

```
ch1 <- "hello world !"
ch2 <- extraire(ch1, 6)
afficher ch2 // on obtient "world !"
```

- **gauche(uneChaine, longueur)** : idem extraire(uneChaine, 1, longueur)
- **droite(uneChaine, longueur)** : idem extraire(uneChaine, taille(uneChaine)-longueur)

Opérations sur les chaînes

Recherche de motif

- position(chaineAchercher, uneChaine) : retourne la position où se trouve le contenu de chaineAchercher dans uneChaine.

  ```
  ch1 <- "hello world !"
  afficher position("r", ch1) // on obtient 8
  ```

  Si la chaîne n'est pas trouvée, la fonction retourne 0. Si la chaîne est présente plusieurs fois, elle retourne la première position.

Opérations sur les chaînes

Autres possibilités

- découper une chaîne par rapport à un caractère
- remplacer des caractères ou des sous-chaînes par d'autres
- transformer une chaîne en majuscule
- etc.

Opérations binaires

et, ou, ou exclusif

Ces opérateurs existent au niveau binaire. Il est rare de les voir au niveau algorithmique. Pour simplifier, ou utilisera la même syntaxe que les opérateurs classiques. Il n'y aura pas de confusion car les opérateurs booléens classiques s'appliquent entre valeurs booléennes, alors que les opérateurs binaires s'appliquent sur des nombres.
Exemple :

```
val1 <- 78 // correspond à 1001110
val2 <- 45 // correspond à 101101
val3 <- val1 et val2 // val3 contient 0001100 donc 1210
val4 <- val1 ou val2 // val4 contient 1101111 donc 11110
val5 <- val1 oux val2 // val5 contient 1100011 donc 9910
```

- **et :** les deux bits doivent être à 1 pour que le résultat soit à 1, sinon le résultat est à 0.
- **ou :** il suffit qu'un bit soit à 1 pour que le résultat soit à 1.
- **oux :** un seul des deux bits doit être à 1 pour que le résultat soit à 1.

Opérations binaires

Changement d'état

Il existe aussi un opérateur qui permet de changer l'état d'un bit : s'il est à 1, il passe à 0, et vice-versa. Pour cet opérateur, on va utiliser le même signe qu'en C++.
Exemple :

```
val1 <- 78 // correspond à 1001110
val2 <- ~val1 // val2 contient 0110001 donc 4910
```

Opérations binaires

Masque

En informatique, l'utilisation de masques est classique : vous le verrez plus en détail lorsque vous étudierez les adressages au niveau des ordinateurs. Appliquer un masque à une adresse de machine, c'est récupérer l'adresse du réseau dans lequel se trouve la machine.
Comment cela fonctionne-t-il ? Appliquer un masque à une valeur, c'est récupérer uniquement les bits qui correspondent aux bits à 1 du masque, tous les autres bits étant passés à 0. L'opérateur ET est utilisé pour appliquer un masque.
Exemple :

```
vadresse <- 78 // correspond à 1001110
masque <- 120 // correspond à 1111000
reseau <- adresse et masque // reseau contient 1001000 donc 7210
```

Exercice 2

Ecrire l'algorithme qui permet de saisir 2 valeurs binaires. On affichera pour chaque valeur sa conversion en base 10 puis le résultat du et entre ces 2 valeurs, affiché en base 10 et en binaire.
Traduire l'algo en C++ pour le tester. Pour les manipulations de valeurs numériques qui doivent subir des opérations binaires, il est conseillé de travailler avec des variables non signées, donc positives (unsigned int en C++). Attention, le et binaire ne se note pas && mais & en C++.