Numérique et sciences de l'information

Tous niveaux

Module No 06

Combinaison des structures de base dans un programme

© Trigger

Programme

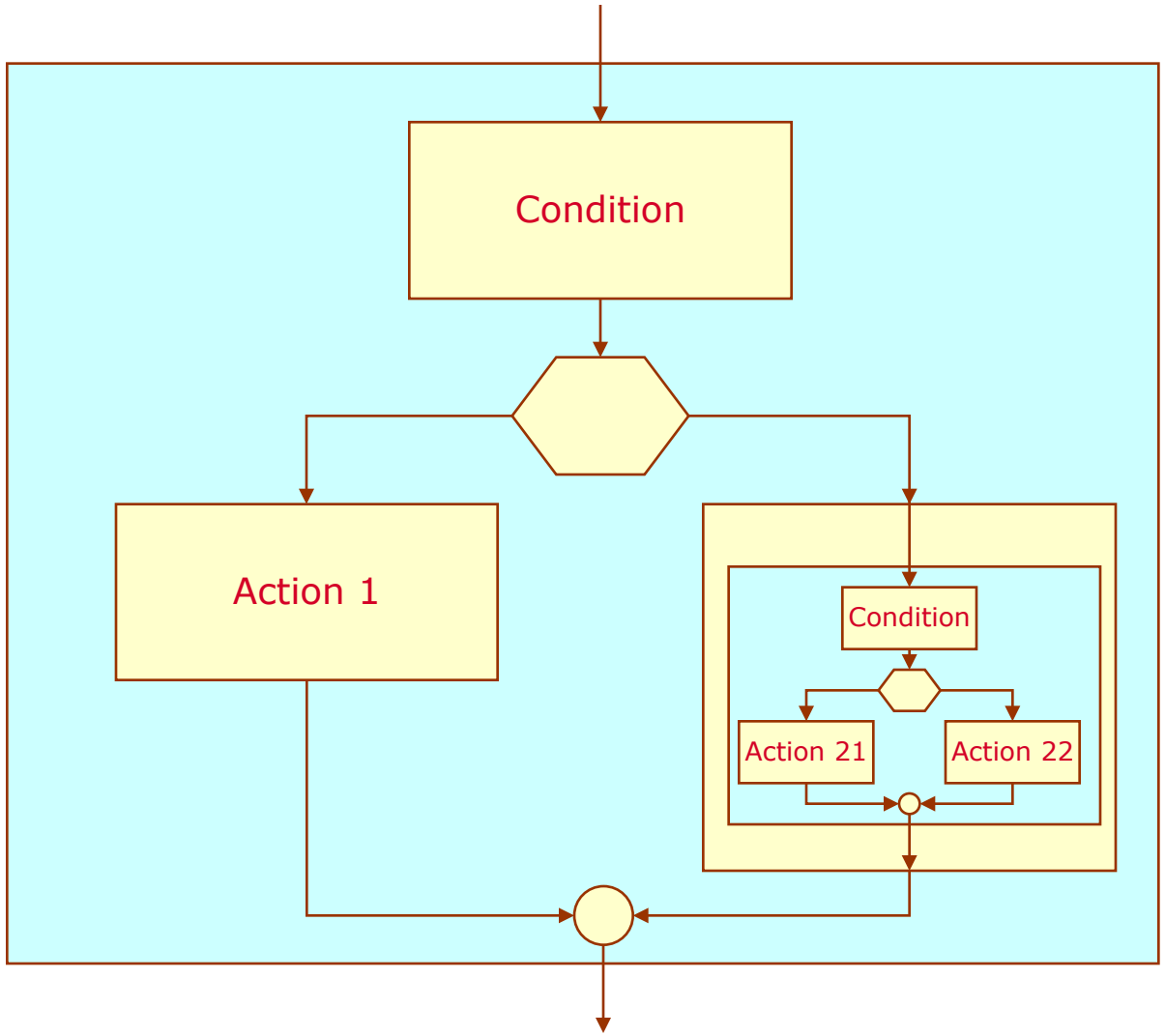
- **CM#6 Combinaisons des structures de base et langage de programmation**
- Concept d'imbrication.
- Combinaison de structures de base.
- Sous-programmes.
- Aller plus loin avec un langage de programmation

Programme

- **TD Séance #3 : Manipuler les combinaisons de structures**
- Exercice #1 : Version de la somme des n premiers nombres avec contrôle des données saisies
- Exercice #2 : Analyser une chaîne de caractères et compter le nombre d'occurrences de certains caractères spéciaux
- Exercice #3 : Déterminer parmi les 100 premiers nombres ceux qui sont parfaits.
- Exercice #4 : Lister les propriétés d'un objet
- Exercice #5 : Vérifier qu'un nombre est bien aléatoire

Concept d'imbrication

- Exemple d'une structure alternative imbriquée dans une autre.
- Cette imbrication ne pose aucun problème du fait que chaque structure et chaque bloc au sein de la structure n'ont qu'un point d'entrée et un point de sortie



© Trigger

Exercice imbrication No 1

- Imbrication d'une structure alternative dans une structure alternative (schéma précédent)
- Il s'agit de déterminer le nombre de points obtenus par un joueur qui tire une carte dans un jeu de cartes selon les attributions de points suivantes :
  - 3 points pour un as de carreau ou un 4 de trèfle
  - 2 points pour un quatre de cœur
  - 1 point pour un quatre de pique ou de carreau
  - 1 point pour un as autre que l'as de carreau
  - 0 point pour toute autre carte

© Trigger

Exercice imbrication No 1

	♣	♦	♥	♠
AS	1	3	1	1
4	3	1	2	1
Autre	0	0	0	0

© Trigger

Exercice imbrication No 1

```
$: /*Jeu de cartes */
§1: Tirer une carte :1§
$2: SI la carte est un as
        ALORS $21:  SI la couleur est  carreau
                        ALORS  $211: 3 points :211$
                        SINON  $212: 1 point  :212$  :21$
         SINON $22:  (la carte n'est pas un as)
                        SI la carte est un Quatre
                            ALORS $221: SI la couleur est trèfle
                                                ALORS $2211: 3 points :2211$
                                                SINON $2212: SI la couleur est cœur
                                                                    ALORS  $22121  2 points
                                                                    SINON  $22122  1 point
                                                                FIN SI :2212$
                                                FIN SI  :221$
                                SINON (la carte n'est ni un as ni un 4)  $222: 0 point :222$
                        FIN SI  :22$
            FIN SI :§2
$3: Affichage sore  :3$
:$
```


Exercice imbrication No 2

- Améliorer une des solutions précédentes de la somme des n premiers nombes pour résoudre de manière satisfaisante le problème de la saisie de nombres négatifs, et d'une manière plus générale, celui du contrôle de la valeur saisie.

© Trigger

Exercice imbrication No 2

Exemple No 6 : Somme de n entiers. Version avec contrôle entrée

```
$:  /* Exercice #6 Somme de n entiers avec contrôle entrées*/
$1:  /* Initialisations */
Variables N (variable), Nref (variable de référence), Somme (Résultat recherché) initialisées à 0
:1
$2:  /* Boucle d'attente d'une variable acceptable */
REPETE
  $21: /* Bloc dans boucle */
     $211:   /* 1 séquence : saisie */
     Afficher « Saisissez un nombre entier»
     N <- valeur saisie  :211$
     $212:  /*  2 structure test et calcul */
     SI N n'est pas numérique
        ALORS   $2121: Message d'anomalie N doit être numérique :2121$
        SINON   $2122:
                   $21221:    Nref <- N  :21221$
                   $21222:
                   SI N<= 0
                      ALORS   $212221: Message d'anomalie N doit être >0  :212221$
                      SINON   $212222:
                                 $2122221:   /* Boucle de calcul */
                                 TANT QUE N > 0
                                 REPETE $2122221: Somme <- Somme + N
                                 N <- N-1  :2122221$
                                 FIN TANT QUE     :2122221$
                                 $2122222:  /* Livraison résultats */
                                 Afficher «La somme des »  Nref « premiers entiers est » Somme :2122222$
                               :212222$
                    :21222$
                 :2122$
     :212$
   :21$
TANT QUE Somme = 0
:2$
:$
```

Exercice imbrication No 2

```
<HTML>
<HEAD>
   <title>Programmatique Exo 6'</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Somme des n premiers nombres
</B><p>
<SCRIPT LANGUAGE="JavaScript">
/* Exercice #6 Somme de n entiers avec contrôle entrées */
/* Initialisation */
var N = 0;
var NRef = 0;
var Somme = 0;
do
{
N = prompt("Saisissez un nombre entier positif",0);
N=N*1;
if (isNaN(N))
{
window.alert("Vous devez saisir un valeur numérique");
}
else
{
Nref = N;
if (N<=0)
  {
  window.alert ("Vous devez saisir un nombre entier positif");
  }
  else
  {
  /* Boucle de calcul */
  while (N > 0)
     {
     Somme = Somme + N;
      N = N-1 ;
      }
     /* Livraison résultats */
      document.write("La somme des " + Nref + " premiers entiers est " + Somme + ".<BR>");
       document.write ("Fin du processus.<BR>");
   }
  }
 }
    while (Somme==0);
</SCRIPT>
 </FONT>
</BODY>
</HTML>
```

© Trigger

Exercice imbrication No 2

- Test et analyse
  - Imbrication répétitive (calcul) et séquence (fourniture résultat)
  - dans une structure alternative (test nombre positif)
  - elle-même imbriquée dans une structure alternative (test numéricité)
  - venant elle-même enchaînée à une séquence (saisie)

Exercice imbrication No 3

- Analyser une chaîne de caractères -terminée par le caractère $- et compter le nombre d'occurrences de certains caractères spéciaux (, ; : . -).

Exercice imbrication No 3

Exemple No 7 : Comptage caractères

```
$:  /* Exercice #7 Comptage caractéres */
$1:  /* Initialisations */
Variables : Texte à analyser, compteur de progression dans le texte i et compteurs
pour les différents caractères     :1$
$2:   /* Boucle d'analyse */
TANT QUE caractère de rang i <> "$" /* l'incrément de i pourra se faire à ce niveau
*/
      REPETE $21: /* bloc dans boucle */
            $211: /* structure SELON pour analyse caractère */
            SELON caractère
            CAS caractère = " , "
                  $2111: +1 sur compteur virgules  :2111$
            CAS caractère = " . "
                  $2112: +1 sur compteur points  :2112$
            CAS caractère = " : "
                  $2113: +1 sur compteur deux points  :2113$
.....
            CAS tous autres caractères
                  $211n: +1 sur compteur autres caractères  :211n$
            FIN SELON      :211$
            $212:  +1 sur i   :212$
            :21$
        FIN TANT QUE :2$
$3:   /* Publication des résultats */
Afficher ……
:3$   :$
```

Exercice imbrication No 3

```
<HTML><HEAD>
                    <title>Programmatique Exo 7 _ Noter programme dans Entête'</title>
                    <SCRIPT LANGUAGE="JavaScript">
var virgule=0;
var point = 0;
var tiret =0;
var deuxpoints = 0;
var apostrophe = 0;
var pointvirgule=0;
var pointexclamation=0;
var pointinterrogation=0;
var lettre = 0;
var i = 0;

texte="Faîtes des phrases courtes. Un sujet, un verbe, un complément. Quand vous voudrez ajouter un objectif, vous viendrez me voir." + "C'est ainsi que Georges Clemenceau
s'adressait aux journalistes de l'Aurore - Enfin, c'est ce qu'on dit !..$";

while ((car = texte.charAt(i++)) != "$")
  {
  switch(car)
    {
                    case "," : virgule++;
                               break;
                    case "." : point++;
                                                break;
                    case "-" : tiret++;
                                                break;
                    case ":" : deuxpoints++;
                                                break;
                    case "'" : apostrophe++;
                                                break;
                    case ";" : pointvirgule++;
                                                break;
                    case "!" : pointexclamation++;
                                                break;
                    case "?" : pointinterrogation++;
                                                break;
                    default : lettre++;
                                                break;
                   }
    }
document.write("Ce texte contient :<BR>");
document.write( "  " + virgule + " virgule(s).<BR>");
document.write( "  " +  point + " point(s).<BR>");
document.write( "  " +  deuxpoints + " deux points.<BR>");
document.write( "  " +  tiret + " tiret(s).<BR>");
document.write( "  " +  apostrophe + " apostrophe(s).<BR>");
document.write( "  " +  pointvirgule + " point(s) virgule(s).<BR>");
document.write( "  " +  pointexclamation + " point(s) d'exclamation.<BR>");
document.write(  "  " + pointinterrogation + " point(s) d'interrogation.<BR>");
document.write( "  " +  lettre + " autres caractères.<BR>");

</SCRIPT>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> C'était "compter certains caratères dans une phrase"
</B><BR>
</BODY></HTML>
```


Exercice de synthèse

Analyser le fonctionnement d'un distributeur de billets de banques qui utilise une carte de crédit.

Doivent être vérifiés les points suivants :

- La réserve de billets
- La date de péremption de la carte
- Le nombre maximum de billets autorisés
- Le code secret de la carte

Noter que l'exercice analyse le fonctionnement d'une machine dont on connaît le compétences.

Il ne prend pas en compte le comportement humain, celui de l'utilisateur devant le distributeur

© Trigger

Exercice de synthèse

```
$: Distributeur de billets
   $1: Le distributeur est mis en service
       La réserve est alimentée en billets :1$
   $2: Fonctionnement du distributeur :2$
   $3: Le distributeur est mis hors service :3$
:$
```

Exercices de programmation
Exercice de synthèse

$2: Fonctionnement du distributeur
    TANT QUE la réserve n'est pas vide REPETE
        $21: L'utilisateur introduit sa carte dans l'appareil
                Traitement de la carte
        :21$
:2$

Exercice de synthèse

$21: L'utilisateur introduit sa carte dans l'appareil
SI la carte n'est pas périmée
ALORS $211: SI le nombre de billets autorisés n'est pas atteint
ALORS $2111: Traitement de la demande valide
:2111$
SINON $2112: La carte est restituée avec message « PLAFOND ATTEINT »
:2112$
:211$
SINON $212: La carte est restituée avec un message « carte périmée »
:212$
Le distributeur se prépare à traiter une nouvelle demande
:21$

Traitement des tableaux

- Les **tableaux de données** jouent un grand rôle dans les processus de traitement de l'information
- La généralisation du modèle relationnel fait que toutes les données sont organisées en tables (tableaux de structure particulière), interrogeables au moyen de requêtes exprimées dans un langage particulier : SQL
- Le succès de la micro-informatique tient en grande partie au succès du tableur qui organise données et algorithmes de traitement sous forme de tableaux.

Tableau à 1 dimension

- Vecteur V(n)
- Exemple Vecteur Valeur(6) contient 6 éléments

23	4	0	56	0	77

- Nous identifierons l'élément Valeur[5] d'indice 5=77
- Nous identifierons l'élément Valeur[0] d'indice 0=23
- En Javascript un tel tableau s'écrit Valeur = new Array(5) et l'élément de rang 0 Valeur[0]
- Attention : les conventions relatives à l'emploi des parenthèses () et les crochets [] changent pour chaque langage

© Trigger

Programmation dans un tableau à 1 dimension

- Spécifier le programme assurant le comptage des éléments non nuls dans ce tableau
- Le tableau est initialisé dans le programme
- Dans la pratique il serait sous forme d'un fichier, mis à jour par un autre programme, qui serait lu pour être chargé en mémoire sous forme d'un tableau à 1 dimension.

Programmation dans un tableau à 1 dimension

Exemple No 8 : Comptage éléments tableaux non nuls

```
$:  /* Exercice #8 Comptage tableau */
$1:  /* Initialisations */
N <- nombre d'éléments du tableau
Valeur(N)  est le tableau
Valeur[0] <- 23; Valeur[1] <- 4; Valeur[2] <- 0;
Valeur[3] <- 56; Valeur[4] <- 0; Valeur[5] <- 77;
I <- 1 ; Compteur <- 0
 :1$
$2:   /* Boucle d'analyse */
TANT QUE I < N
      REPETE $21: /* bloc dans boucle */
               $211: /* test nullité */
               SI Valeur[I-1] <> 0 ALORS $2111:     +1 sur Compteur     :2111$
                                   SINON $2112:     Rien     :2112$
               FIN SI    :211$
               $212: /* Incrément boucle */
               +1 sur I        :212$
              :21$
       FIN TANT QUE :2$
$3:   /* Publication des résultats */
Afficher "Il y a " Compteur " éléments non nuls dans le tableau".
Afficher "Fin traitement"
:3$
:$
```

© Trigger

Programmation dans un tableau à 1 dimension

```
<HTML>
<HEAD>
        <title>Programmatique Exo 8'</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Comptage éléments non nuls tableau
</B><BR>
<SCRIPT LANGUAGE="JavaScript">
var N = 6; // nombre d'éléments du tableau
Valeur = new Array(N);  //tableau
Valeur[0] = 23;
Valeur[1] = 4;
Valeur[2] = 0;
Valeur[3] = 56;
Valeur[4] = 0;
Valeur[5] = 77;
I = 1 ;
Compteur = 0;

/* Boucle d'analyse */
while ( I < N)
      { /* bloc dans boucle */
            /* test nullité */
           if (Valeur[I-1] != 0) {Compteur++; }
            /* Incrément boucle */
           I++;

      }
/* Publication des résultats */
document.write ("Il y a " + Compteur + " éléments non nuls dans le tableau.<BR>");
document.write ("Fin traitement<BR>");
</SCRIPT>
 </FONT>
</p>
</BODY>
</HTML>
```

© Trigger

Tableau à 2 dimensions

- Matrice V(n,p)
- Exemple Matrice Table(6,2) contient 12 éléments répartis sur 6 lignes et 2 colonnes
- Nous identifierons l'élément Table[5][0]=78
- Nous identifierons l'élément Table[0][1]="Chenonceaux"
- En Javascript, la création d'un tel tableau se fait en deux temps
- Table = new Array();
  Table[0]=new Array(6);
  Table[1]=new Array(6);
  Table[0][0]=3;

3	Chenonceaux
56	Chambord
17	Amboise
5	Langeais
13	Chinon
78	Azay le Rideau

Tableau à 2 dimensions

- Généralisation du problème analysé sur 1 dimension.
- Pas d'exigence particulière puisque tous les problèmes peuvent être résolus par l'utilisation systématique des structures définies.
- Schéma classique : Imbrication de 2 répétitives, l'une indéxée sur les lignes, l'autre sur les colonnes.
- De nombreux problèmes classiques de gestion : tri, interclassement, recherche, ...

© Trigger

Tri et interclassement

- **Tri par sélection** : on recherche le plus grand, on permule le 1er de la liste avec le plus grand trouvé, on recommence en excluant l'élément sélectionné de la liste

- **Tri par fusion** : On suppose que (a1, . . . , al1) et (b1, . . . , bl2) sont deux listes triées dans l'ordre croissant. L'objectif de l'algorithme est de construire la liste qui représente la fusion des deux listes précédentes de sorte que cette liste soit triée dans l'ordre croissant.

- **Tri par insertion dichotomique** : Le tri par insertion nécessite lui aussi l'utilisation d'un tableau intermédiaire. On prend un par un les éléments de la liste de départ pour les insérer dans une autre liste, initialement vide, de sorte que celle-ci reste constamment triée. On vide la première liste pour emplir la seconde. A chaque insertion d'un nouvel élément dans la liste triée, il faut chercher la position adéquate pour qu'elle reste triée. Cette étape utilise une recherche dichotomique.

Exercice synthèse

- **Programmation du jeu de carte**