Algorithmique et Développement web

Eric VEKOUT, Professeur d'Informatique

TABLE DES MATIERES

PARTIE I : ALGORITHMIQUE

I) Eléments de base

1) Structure générale d'un algorithme

La structure générale d'un algorithme est la suivante :

En-tête

Algorithme | fonction | procedure *Nom_algorithme*{ **(**{**var**}*var1 :Type_v1, …,*{**var**}*varn:Type_vn***)**{ **:** *Type_retour*}}

Déclarations

{**var** *liste_variables1:Type 1*

⋮

var *liste_variables n :Type_variables n*

const *const1=valeur1,…, const n=valeur n}*

Instructions

Début

.

.

.

{**retourner** *nom_variable_retour*}

Fin

N.B : les mots écrits en gras représentent des mots-clés du LDA (Langage de description d'algorithme) ; le contenu des accolades est optionnel.

- *Nom_algorithme* : c'est le nom de votre algorithme. Ça doit être un « nom parlant » qui indique ce que l'algorithme fait.
- *(var1 :Type_v1,…,var_n : Type_v_n)* : c'est la liste des paramètres (données en entrée ou sortie) avec leurs types de votre algorithme. Lorsque le mot-clé **var** est utilisé, c'est pour spécifier que ce paramètre est en sortie.
- *Type_retour* **:** c'est le type de donnée du résultat éventuel que l'algorithme fournit. C'est le type de sortie (si celui-ci est une fonction).
- *Liste_variables : Type*: c'est la liste des variables locales (et leurs types) à votre algorithme.

- *(const1=valeur1,…, const_n=valeur_n) :* c'est la liste des constantes (et leurs valeurs) locales à votre algorithme.
- *nom_variable_retour :* c'est la variable contenant la valeur de retour de votre algorithme (si celui-ci est une fonction).

Un algorithme est donc constitué de trois parties principales :

- **L'en-tête :** Contient la nature (fonction ou procedure), le nom et éventuellement les paramètres et le type de retour de d'algorithme que vous écrivez. Le mot-clé « Algorithme » est utilisé dans le cas où l'on veut écrire l'algorithme général qui se sert de toutes les fonctions et procédures définies pour résoudre le problème posé.
- **Les déclarations :** contient les déclarations des variables et des constantes de votre algorithme. Les mots-clés « var » et « const » précèdent respectivement une liste de variables ou une liste de constantes locales à votre algorithme.
- **Les instructions :** contient l'ensemble des instructions qui constituent votre algorithme. Les mots-clés « Début » et « Fin » marquent respectivement le début et la fin des instructions de l'algorithme. Le mot-clé « retourner » précède le nom de la variable contenant la valeur de retour de votre algorithme dans le cas où celui-ci est une fonction.

2) Constantes, Variables et Types de données

a) Les types de données élémentaires

Il existe quatre types de données élémentaires :

- Les entiers dont le nom du type est « **Entier** »
- Les réels dont le nom du type est « **Réel** »
- Les booléens dont le nom du type est « **Booléen** »
- Les caractères dont le nom du type est « **Car** »

N.B : Il existe d'autres types de données comme les chaînes de caractères **(Chaîne).** Mais ce n'est pas exactement un type de donnée élémentaire vu qu'il est construit à partir d'un ensemble de caractères. Une chaîne de caractère est délimitée par les caractères **"** et **".**

b) Les variables

Une variable est un objet (représentant un espace mémoire) contenant une valeur d'un type donné et dont le contenu est modifiable. Une variable est donc constituée :

- D'un identificateur représentant le nom de la variable.
- D'un type de donnée

- D'une valeur

L'identificateur d'une variable commence toujours par une lettre ou par un underscore (_)

Exemple de déclaration d'une variable locale appelée « nbr » et de type « Entier »:

var nbr : Entier

c) Les constantes

Une constante est un objet contenant une valeur fixe et non modifiable. La valeur affectée à une constante lors de son initialisation est donc définitive. Elle possède aussi un identificateur qui suit les mêmes règles que ceux des variables.

3) Les opérateurs

a) L'affectation

C'est l'opération qui permet d'affecter une valeur à une variable.

Syntaxe : *nom_variable*←**valeur ou expression ;**

L'expression est une suite d'opérations sur des constantes ou des variables déjà déclarées.

Cette valeur ou cette expression doit être du même type que la variable.

b) Opérateurs sur les entiers et les réels

Arithmétiques	
+	Addition
-	Soustraction
*	Multiplication
/	Division
DIV	Division entière
↑	Puissance

comparaisons	
>	Supérieur
<	Inférieur
≥	Supérieur ou égal
≤	Inférieur ou égal
=	Egal
≠	Différent

c) Opérateurs sur les entiers et les booléens

Fonctions logiques	
Mot clé	
et	Fonction ET
ou	Fonction OU
oux	Fonction OU exclusif
non	Fonction NON
non et	Fonction NON ET
non ou	Fonction NON OU
>>	Décalage à droite
<<	Décalage à gauche

Fonctions de comparaison pour les booléens	
=	Egal
≠	Différent

d) Opérateurs sur les caractères et les chaînes

Fonctions de concaténation	
+	Concaténation

Fonctions de comparaison pour les chaînes	
=	Egalité
≠	Différent
>	Supérieur
<	Inférieur

e) Priorité des opérateurs

Priorité à la multiplication et la division.

4) Les structures conditionnelles

a) La structure alternative

Elle permet d'exécuter une suite d'instructions selon un ensemble de condition définies.

Syntaxe :

Syntaxe 1 :

Si *condition* **alors**

Action ;

Fsi

Interprétation :

Action ne s'exécute que si *condition* est vrai.

Syntaxe 2 :

Si *condition* **alors**

Action1 ;

Sinon

Action2 ;

Fsi

Interprétation :

Action1 ne s'exécute que si *condition* est vrai sinon c'est *Action2* qui s'exécute.

N.B : *condition* est une expression booléenne ; *Action* est une suite d'instructions.

b) La structure de choix multiple

Elle permet d'exécuter une suite d'instructions selon la valeur contenue dans la variable passée en paramètre à la structure.

Syntaxe :

Selon le cas de *nom_variable* **faire**

cas *valeur 1* : *Action 1 ;*
.
.
.
cas valeur n : *Action n ;*
défaut : *Action par défaut ;*
Fin cas

Interprétation :
Action {*1…n*}s'exécute respectivement si la valeur de *nom_variable* est *valeur* {*1…n}*.
Action par défaut s'exécute si *nom_variable* n'a aucune des valeurs énumérées plus haut.

c) Les structures répétitives

Elles permettent de répéter l'exécution d'une suite d'instructions jusqu'à ce que la condition de fin de boucle soit vraie. La condition de fin de boucle peut s'exprimer d'une manière directe ou indirecte.

Il existe trois structures répétitives selon les syntaxes suivantes :

Tant Que *condition* **Faire**

Action ;

FinTantQue

Interprétation :

Action s'exécute tant que *condition* est vrai.

N.B : Ici la condition de fin de boucle est *non(condition)*.

Répéter

Action ;

Jusqu'à *condition*

Interprétation :

Action s'exécute jusqu'à ce que *condition* prenne la valeur vrai.

Pour *indice* **allant de** *valeur_initiale* **à** *valeur_finale* ***par pas de*** *increment* **Faire**

Action ;

FinPour

Interprétation :

- *Action* s'exécute tant que *indice* n'a pas atteint *valeur_finale*.
- A la première itération de la boucle, *indice* contient *valeur_initiale.* Au fur et à mesure que la boucle s'exécute, *indice* prend une valeur égale à la valeur qu'elle (la variable indice) avait à l'itération précédente plus *increment*. La boucle fini de s'exécuter lorsque *indice* atteint ou dépasse *valeur_finale* selon l'incrément.
- Si l'expression «**par pas de…** » n'est pas précisée, *increment* à la valeur 1.

N.B : Ici la condition de fin de boucle est *indice* <= *valeur_finale* ou *indice* >= *valeur_finale* selon que (*increment* < 0 et *valeur_finale* < *valeur_initiale*) ou (*increment* > 0 et *valeur_finale* > *valeur_initiale*)

Remarque :

- Lorsque la structure « **Tant Que…Faire** » est utilisée il est possible que *Action* ne s'exécute pas, si *condition* est déjà faux.
- Lorsque la structure « **Répéter…Jusqu'à** » est utilisée *Action* s'exécute toujours au moins une fois.

5) Les commentaires

Pour faciliter sa compréhension, un bon algorithme doit être commenté. Un commentaire n'est pas pris en compte dans l'exécution de l'algorithme. Pour insérer un commentaire dans votre algorithme, les syntaxes sont les suivantes :

// Commentaire tenant sur une seule ligne

/ Commentaire tenant sur*
*plusieurs lignes */*

II) Les structures de données

Une structure de données est un moyen de stocker et organiser des données pour faciliter l'accès à ces données et leur modification. Il existe plusieurs types de structure de données.

Dès sa définition, une structure de donnée est considérée comme un type de donnée au même titre que les entiers par exemple. Donc la manière de déclarer une variable d'un type enregistrement est la même que celle d'un entier.

Il n'y a aucune structure de données qui réponde à tous les besoins, de sorte qu'il importe de connaître les forces et limitations de plusieurs de ces structures.

1) Les enregistrements

Un enregistrement est une structure de donnée composée d'une ou de plusieurs propriétés la caractérisant.

Pour définir un enregistrement, la syntaxe est la suivante :

Type *nom_structure* = **Enregistrement de**

Propriete_1 : Type_1 ;
.
.
.
Propriete_n : Type_n ;

Fin Enregistrement

- *nom_structure* représente le nom de la structure de donnée à définir.
- *Propriete_{1…n} : Type_{1…n}* représente la liste des propriétés de la structure de donnée et leurs types (les propriétés sont des variables).

Exemple :

Type Individu = **Enregistrement de**
Nom : Chaîne ;
Prenom : Chaîne ;
Age : Entier ;
Fin Enregistrement

var *id :* Individu

L'accès à aux propriétés d'une variable d'un type enregistrement se fait de la manière suivante :

nom_variable.propriete

Exemple :

var *n1, n2* : Chaîne
Début
n1 ← "Roland" ;
id.Prenom ← *n1;*
n2 ← *id.Prenom ;*
Fin

2) Les ensembles

Un ensemble est une structure de donnée qui représente un sous-ensemble d'un type de donnée.

Syntaxe :

Type *Nom_ensemble*=**Ensemble de** *Type_donnee*

Exemples :

Type *Age* = **Ensemble de** *Entier*
Type *Personne* = **Ensemble de** *Individu*

3) Les tableaux

Un tableau est une structure de donnée contenant un ensemble séquencé de valeurs d'un type de données bien précis.

Syntaxe :

Type *nom_tableau* = **Tableau[1…n] de** *Type_donnee*

[1..n] représente la plage d'indices des cellules du tableau contenant les données.

Exemples :

Type *Groupe* = **Tableau[1…n] de** *Chaîne*

Type *Immeuble* = **Tableau[1…n] de** *Bureau* (*Bureau* étant une structure de donnée ayant été définie à l'avance).
Ici, *n* est une valeur qu'on peut définir explicitement (par exemple remplacer *n* par 5).

On peut déclarer une variable d'un type tableau sans définir à l'avance une structure de donnée représentant ce type tableau.

Exemple :

var *tab1 :* **Tableau[1…n] de** *Entier*

Pour accéder à une donnée d'un tableau, la syntaxe est la suivante :

*var_tableau***[***indice***]**

- *var_tableau* représente le nom de la variable tableau.
- *indice* représente l'indice de la case du tableau à laquelle on veut accéder.

On peut donc stocker, lire et modifier les données contenues dans les cellules d'un tableau.

Exemple1 :

```
var tab1 : Tableau[1…n] de Entier
var n1, n2 : Entier
Début
        n1 ← 5 ;
        tab1[1] ← n1 ;
        n2 ← tab1[1] ;
        tab1[1] ← 9 ;
Fin
```

Exemple2 :

```
var tab2 : Groupe
var c1, c2 : Chaîne
```

Début

c1 ← "Mvondo" ;
tab2[1] ← c1 ;
c2 ← tab2[1] ;
tab2[1] ← "Ndogmo" ;
tab2[2] ← "Moussa"

Fin

N.B : L'affectation des valeurs dans un tableau se fait de façon séquentielle. Donc lorsqu'on initialise un tableau, il faut remplir les valeurs en allant de la première cellule à la dernière.

III) Les fonctions et les procédures

1) Les fonctions

Une fonction est une opération ayant zéro ou plusieurs paramètres en entrée et retournant un résultat d'un type donné en sortie.

Syntaxe :

Fonction *nom_fonction* **(**{*var_1 :Type_1,…,var_n :Type_n*}**) :** *Type_retour*

var *nom_variable_retour :Type_retour*

{**var** *liste_variables1:Type 1*
.
.
.
var *liste_variables n :Type_variables n*
const *const1=valeur1,…, const n=valeur n}*

Début

Suite d'instructions ;
Retourner *nom_variable_retour ;*

Fin

Exemple :

Fonction max (a : Entier, b : Entier) : Entier

// Cette fonction retourne le plus grand des nombres en entrée

var res : Entier

Début
Si (a<b) **alors**
res ← b ;
Sinon
res ← a ;
Fsi
Retourner res ;
Fin

2) Les procédures

Une procédure est une opération ayant zéro ou plusieurs paramètres en entrée et zéro ou plusieurs résultats en sortie.

Syntaxe :

Procedure *nom_procedure* **(**{{**var**}*var_1 :Type_1,…,* {**var**}*var_n :Type_n*}**)**

{**var** *liste_variables1:Type 1*
.
.
.
var *liste_variables n :Type_variables n*
const *const1=valeur1,…, const n=valeur n}*

Début
Suite d'instructions ;
Fin

Le mot-clé « var » utilisé sur un paramètre signifie que ce dernier représente une sortie pour la procédure : c'est ainsi qu'une procédure peut fournir plusieurs résultats en sortie.

Exemple :

Procedure Addition (a : Entier, b : Entier, **var** c : Entier)

/* *Cette procédure retourne dans la variable c le résultat de l'addition des paramètres a et b*/

Début

c ← a + b;

Fin

3) Appel de fonctions et de procédures

On peut faire appel à une ou plusieurs fonctions ou procédures dans un algorithme. Il suffit juste d'écrire le nom de la fonction ou de la procédure à appeler à l'endroit voulu, en lui passant les paramètres requis pour l'exécution de celle-ci en faisant attention de respecter les types de donnée de chacun de ces paramètres.

Il existe des fonctions ou procédures prédéfinies prêtes à l'emploi comme les procédures de **lecture et d'écriture** qui permettent respectivement de lire les données que l'utilisateur entre à partir de l'entrée standard (le clavier) et d'écrire un texte à la sortie standard (l'écran): ce sont les procédures

- **Lire (** *nom_variable* **) :** la valeur lue au clavier est stockée dans la variable *nom_variable*
- **Ecrire (***var_chaine***) :** le contenu de la variable chaîne de caractère *var_chaine* est affiché à l'écran. Pour afficher le contenu des variables de types Entier et Réel à l'aide de la procédure écrire, on sépare ces variables des chaînes de caractères par une virgule. Exemple : Ecrire(a, ''+'', b,''='',c)

Exemple :

Procedure Test( )

// Cette procédure teste si au moins l'une des valeurs entrées au clavier est négative

var a,b,c : Entier

Début

(1) Lire(a) ; // La valeur entrée est affectée à *a*

(2) Lire(b) ;

(3) *Addition(a,b,c)* ; // Appel de la procédure addition définie plus haut

(4) **Si** (c > *max(a,b)*) alors // Appel de la fonction max définie plus haut

(5) **Ecrire**(a,‘‘ et ’’,b, ‘‘sont positifs’’) ; /* les valeurs de *a* et de *b* sont affichées à l’écran*/

(6) **Sinon**

(7) **Ecrire**(‘‘Il existe au moins une valeur négative entre ’’, a,‘‘ et ’’,b) ;

Fsi

Fin

Remarque : On voit bien ici que l’appel d’une fonction fourni directement en retour le résultat de cette fonction. Voir ligne n°4.

IV) La récursivité

Une définition récursive est une définition dans laquelle intervient ce que l’on veut définir. Un algorithme est dit récursif lorsqu’il est défini en fonction de lui-même. Cela veut dire qu’un algorithme est récursif s’il fait appel à lui-même dans le corps de ses instructions.

Le danger avec un algorithme récursif est qu’il faut toujours s’assurer que la **condition de terminaison** de l’algorithme est bien définie et sera atteinte sinon on se retrouve dans une boucle infinie. Prenons par exemple le calcul des puissances d’un réel :

Fonction puissance(x :Réel, n : Entier) : Réel

// Cette fonction retourne x^n

var p : Réel

Début

p ← x*puissance (x, n-1) ; //Selon que $x^n = x*x^{n-1}$

retourner p ;

Fin

Ici, on remarque que la condition de terminaison de la récursivité est n=0 car $x^0=1$. Donc lorsqu’on obtient ce résultat, on remonte dans les calculs jusqu’à obtenir la valeur de x^n. Or dans cet algorithme il n’y rien qui contrôle la valeur de n donc il ne se terminera jamais !

Une version corrigée de cet algorithme est : (On garde l'en-tête et les déclarations et on suppose ici que n est toujours positif ou nul)

Début
    Si n > 0 **alors**
        p ← x*puissance (x, n-1) ;
    Sinon
        p ← 1 ;
    Fsi
    Retourner p ;
Fin

On voit bien ici que lorsque n prend la valeur 0, la fonction retourne la valeur 1 (car $x^0 = 1$). Donc à partir de là on peut remonter jusqu'à l'obtention de x^n.

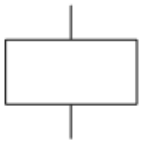
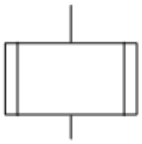
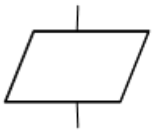
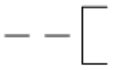
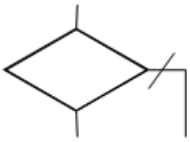
V) Algorigrammes et exécution « à la main » d'algorithmes

1) Algorigrammes

C'est une représentation graphique de l'algorithme. Pour le construire, on utilise des symboles normalisés.

Ainsi, les instructions, les structures conditionnelles, les commentaires et etc. ont des symboles de représentation.

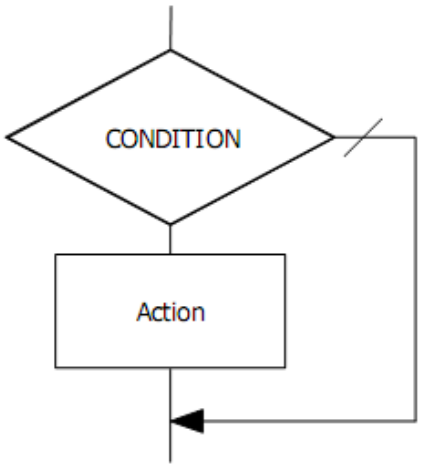
Quelques symboles utilisés dans la construction d'un algorithme :

<table>
<tr><th>SYMBOLE</th><th>DESIGNATION</th><th>SYMBOLE</th><th>DESIGNATION</th></tr>
<tr><td colspan="2">Symboles de traitement</td><td colspan="2">Symboles auxiliaires</td></tr>
<tr><td>  </td><td>Symbole général
Opération ou groupe d'opérations sur des données, instructions, pour laquelle il n'existe aucun symbole normalisé.</td><td>  </td><td>Renvoi
Symbole utilisé deux fois pour assurer la continuité lorsqu'une partie de ligne de liaison n'est pas représentée.</td></tr>
<tr><td>  </td><td>Sous-programme
Portion de programme considérée comme une simple opération.</td><td>  </td><td>Début, fin , interruption
Début, fin ou interruption d'un algorigramme.</td></tr>
<tr><td>  </td><td>Entrée-Sortie
Mise à disposition d'une information à traiter ou enregistrement d'une information traitée.</td><td>  </td><td>Commentaire
Symbole utilisé pour donner des indications sur les opérations effectuées.</td></tr>
<tr><td colspan="2">Symbole de test</td><td colspan="2" rowspan="2">Les différents symboles sont reliés entre eux par des lignes de liaisons.</td></tr>
<tr><td>  </td><td>Branchement
Exploitation de conditions variables impliquant un choix parmi plusieurs.</td></tr>
<tr><td colspan="4">Sens conventionnel des liaisons</td></tr>
<tr><td colspan="4">Le sens général des lignes de liaison doit être :
• De haut en bas
• De gauche à droite
Lorsque le sens général ne peut pas être respecté, des pointes de flèche à cheval sur la ligne indiquent le sens utilisé.</td></tr>
</table>

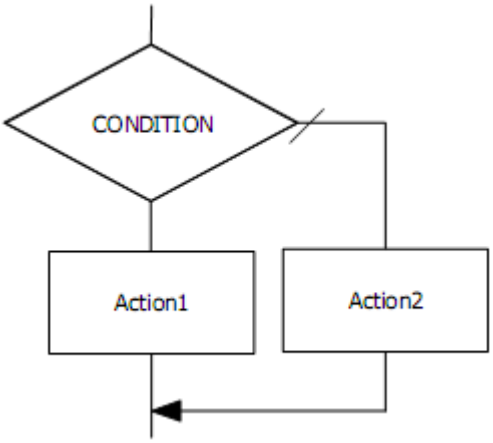
Voici donc les représentations symboliques des structures conditionnelles déjà étudiées plus haut :

a) Structure alternative

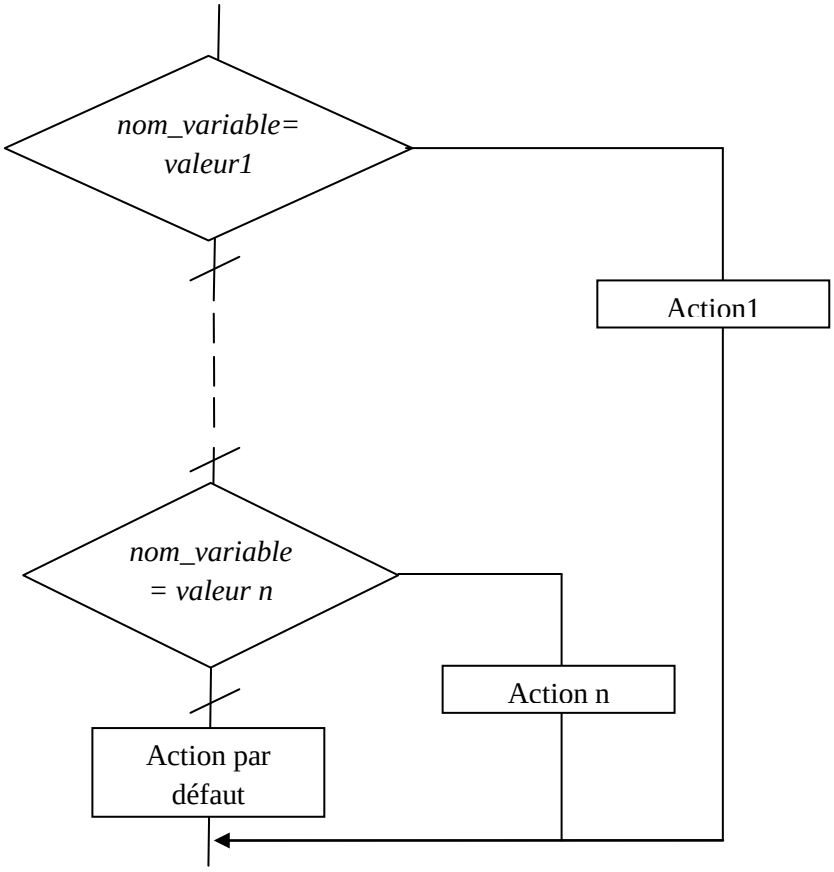
Si *condition* **Alors** *Action ;* **Fsi**



Si *condition* **Alors** *Action1 ;* **Sinon** *Action2 ;* **Fsi**



b) Structure de choix multiples



Selon le cas de *nom_variable* **faire**

cas *valeur 1* : *Action 1 ;*

.

.

.

cas *valeur n* : *Action n ;*

défaut : *Action par défaut ;*

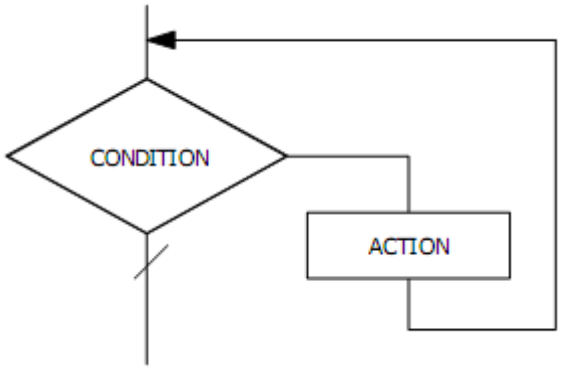
Fin cas

c) Structures répétitives

Tant Que *condition* **Faire**

Action ;

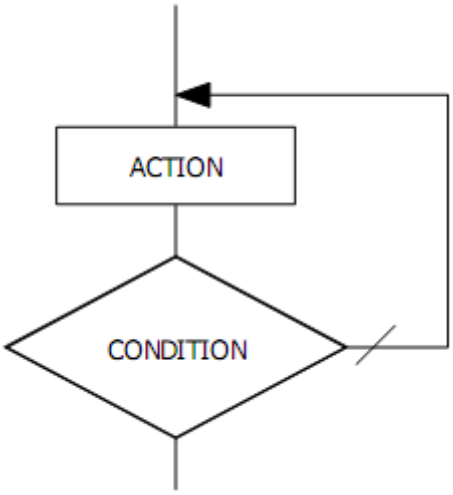
FinTantQue



Répéter

Action ;

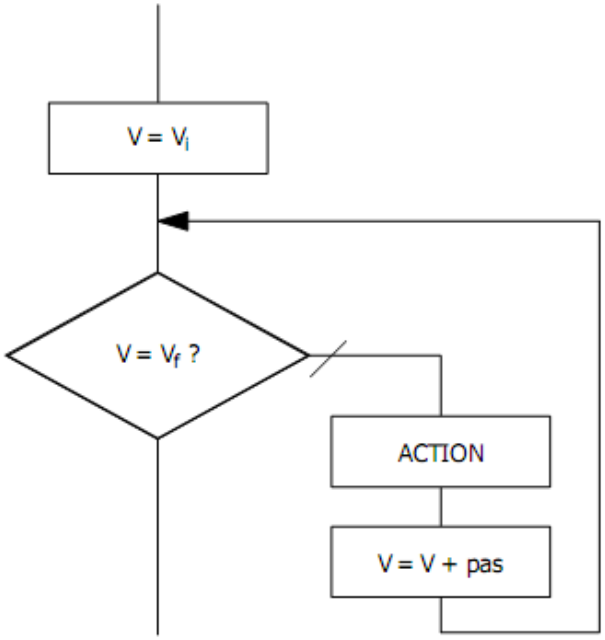
Jusqu'à *condition*



Pour *indice* **allant de** *valeur_initiale* **à** *valeur_finale* ***par pas de*** *increment* **Faire**

Action ;

FinPour



Ici, on considère que :

V_i= valeur_initiale

V_f= valeur_finale

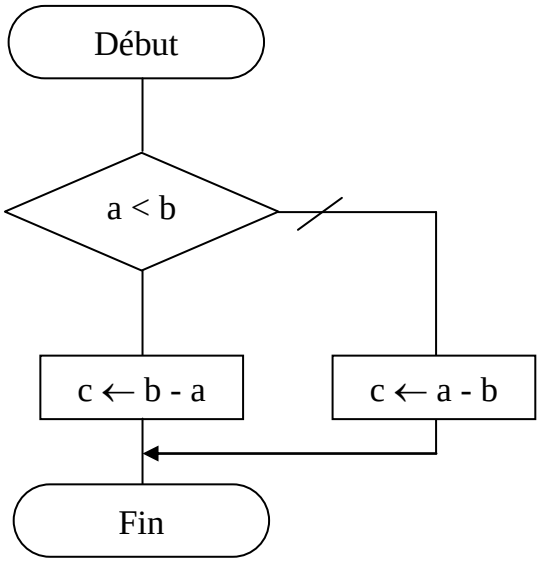
pas= increment

d) Un exemple complet

Soit l'algorithme suivant :

Procedure exemple (a : Réel, b : Réel, var c : Réel)
Début
Si (a < b) **Alors**
c ← b – a ;
Sinon
c ← a – b ;
Fsi
Fin

L'algorigramme correspondant à cet algorithme est le suivant :



2) Exécution à la main d'un algorithme

Exécuter un algorithme à la main c'est établir l'état de ses variables après chaque instruction et en fonction des données en entrée. Pour se faire, il suffit premièrement de numéroter chaque instruction de l'algorithme, puis de tracer un tableau croisé ayant une entrée pour chaque variable et une autre pour chaque instruction (chaque instruction est identifiée par son numéro d'ordre et elles sont classées dans le tableau par ordre d'apparition) ; à l'intersection de ces deux entrées, on a la valeur de la variable après l'exécution de l'instruction en cours.

N.B : On utilise aussi l'expression « dérouler» lorsqu'on veut dire exécuter à la main.

Exemple :

```
Procedure Essai(x : Entier, y : Entier, var t : Tableau[1…n] de Entier)
    var i : Entier
    Début
(1)         i ← 2 ;
(2)         Pour i allant de 1 à y faire
(3)                 t[i] ← x + i;
            FinPour
    Fin
```

Pour pouvoir dérouler cet algorithme, il faut premièrement attribuer des valeurs aux données en entrée (ces valeurs sont attribuées dans le programme faisant appel à cet algorithme). On prendra donc x=1 et y=4. L'instruction 0 sert à vérifier l'état des variables pré-exécution. ; On obtient donc le tableau suivant :

Instruction	0	1	2	3	2	3	2	3	2	3
Variable										
x	1	1	1	1	1	1	1	1	1	1
y	4	4	4	4	4	4	4	4	4	4
i		2	0	0	1	1	2	2	3	3
t				t[0]=1	t[0]=1	t[1]=2	t[1]=2	t[2]=3	t[2]=3	t[3]=4

Remarque : Ici on retrouve quatre fois les instructions (2) et (3) parce que la *boucle pour* représentée par ces instructions s'exécute quatre fois (*i* va de 1 à 4).

PARTIE 2 : HTML ET FEUILLES DE STYLE (CSS)

I) Présentation

1) HTML

Le **HTML** (« *HyperText Mark-Up Language* ») est un langage dit de « marquage » (de « structuration » ou de « balisage ») dont le rôle est de formaliser l'écriture d'un document avec des balises de formatage. Les balises permettent d'indiquer la façon dont doit être présenté le document et les liens qu'il établit avec d'autres documents. Le HTML n'est pas un langage de programmation. Il s'agit d'un langage permettant de décrire la mise en page et la forme d'un contenu rédigé en texte simple.

Le langage HTML permet notamment la lecture de documents sur Internet à partir de machines différentes, grâce au protocole HTTP, permettant d'accéder via le réseau à des documents repérés par une adresse unique, appelée URL (*Uniform Resource Locator*).

Une page HTML est un simple fichier contenant du texte formaté avec des balises HTML. Par convention l'extension donnée au fichier est .htm ou .html, mais une page web peut potentiellement porter d'autres extensions notamment :

- **.asp** pour une page générée dynamiquement en ASP (Active Server Pages) ;
- **.cgi** pour une page générée dynamiquement avec des CGI (Common gateway Interface) ;
- **.php**, **.php3** ou **.php4** pour une page générée dynamiquement en PHP ;
- **.pl** pour une page générée dynamiquement en Perl (Practical Extraction and Report Language) ;
- etc.

Une page web peut être construite à partir du plus basique des éditeurs de texte (une application bloc-notes par exemple), mais il existe des éditeurs beaucoup plus évolués.
Les éditeurs WYSIWYG («What You See Is What You Get», littéralement «ce que vous voyez est ce que vous obtenez») sont des éditeurs graphiques permettant de travailler sur une page web telle qu'elle sera affichée sur un navigateur à quelques détails près. Grâce à ce genre d'éditeurs il est possible d'ajouter des balises par simple clic et d'en modifier les

attributs en éditant leurs propriétés dans un formulaire. Pour autant, afin d'utiliser au mieux ce genre d'éditeur, une connaissance préalable du HTML est tout de même très utile.

Il existe également des éditeurs permettant d'éditer le code HTML en affichant les balises, les attributs et leurs valeurs avec différentes couleurs pour une meilleure lecture et proposant parfois des outils pour vérifier la validité du code HTML (par exemple *Dreamweaver* d'Adobe).

2) Les feuilles de style

Le principe des feuilles de style consiste à regrouper dans un même document des caractéristiques de mise en forme associées à des groupes d'éléments. Il suffit de définir par un nom un ensemble de définitions et de caractéristiques de mise en forme, et de l'appeler pour l'appliquer à un texte.

Les feuilles de style ont été mises au point afin de compenser les manques du langage HTML en ce qui concerne la mise en page et la présentation. En effet, le HTML offre un certain nombre de balises permettant de mettre en page et de définir le style d'un texte, toutefois chaque élément possède son propre style, indépendamment des éléments qui l'entourent. Grâce aux feuilles de style, lorsque la charte graphique d'un site composé de plusieurs centaines de pages web doit être changée, il suffit de modifier la définition des feuilles de style en un seul endroit pour changer l'apparence du site tout entier !

Elles sont appelées « *feuilles de style en **cascade*** » (en anglais « **Cascading Style Sheets** ») car il est possible d'en définir plusieurs et que les styles peuvent être hérités en cascade.

Les feuilles de style permettent notamment :

- d'obtenir une présentation homogène sur tout un site en faisant appel sur toutes les pages à une même définition de style ;
- de permettre le changement de l'aspect d'un site complet entier par la seule modification de quelques lignes ;
- une plus grande lisibilité du HTML, car les styles sont définis à part ;
- des chargements de page plus rapides, pour les mêmes raisons que précédemment ;
- un positionnement plus rigoureux des éléments.

II) Concepts et éléments de base du langage HTML

1) Les balises

Une balise est un *marqueur* produisant un effet particulier dans une page HTML et selon les valeurs implicites ou explicites de ses attributs (si elle en possède).

Les balises HTML fonctionnent par paire afin d'agir sur les éléments qu'elles encadrent. La première est appelée « *balise d'ouverture* » (parfois *balise ouvrante*) et la seconde « *balise de fermeture* » (ou *fermante*). La balise fermante est précédé du caractère */*) : **<balise>** *contenu* **</balise>** . Toutefois il existe des balises jouant à la fois les deux rôles (donc ne fonctionne pas par paire) et elles ont la forme : **<balise / >.** Ce sont des balises de la norme XHTML.

Un attribut est un élément, présent au sein de la balise ouvrante, permettant de définir des propriétés supplémentaires permettant le paramétrage la balise. Les attributs se présentent la plupart du temps comme une paire clé=valeur, mais certains attributs ne sont parfois définis que par la clé. Les valeurs sont généralement mises en deux guillemets ("et") : **<balise *attr="val"*>** *contenu* **</balise>** (*attr* est la clé de l'attribut et *val* sa valeur).

Les balises peuvent être imbriquées de manière hiérarchique et dans ce cas, leurs effets sont cumulés : **<balise1><balise2>** *Contenu 1* **</balise2>** *Contenu 2* **</balise1>.**

Remarque :

- *Contenu 1* est affecté par balise1 et balise2 alors que *Contenu 2* n'est affecté que par balise2.
- L'imbrication hiérarchique des balises imposes qu'elles soient fermées dans l'ordre inverse de leur ouverture (balise1 est la première ouverte dans la hiérarchie d'imbrication donc ça doit être la dernière à être fermée : balise2 se ferme donc avant balise1 bien qu'ayant été ouverte après).

2) Structure d'un document HTML

Un document HTML se présente généralement de la manière suivante :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
  <HTML>
    <HEAD>
          <TITLE>Titre de la page</TITLE>
          …
```

```
    </HEAD>
    <BODY>
        Contenu de la page
    </BODY>
</HTML>
```

- < ! DOCTYPE …> est utilisé pour déclarer le *prologue du type de document*.
- <HTML> …</HTML> indique que nous allons écrire du HTML. En effet, le HTML n'est pas le seul langage connu par les navigateurs, il existe notamment le JavaScript.
- <HEAD>…</HEAD> indique l'en-tête du document qui peut contenir par exemple le titre de la page, les méta-informations, etc.
- <BODY>…</BODY> indique le contenu de la page à afficher dans le navigateur.

La déclaration du prologue de type de document indique la DTD (*Document Type Definition*) utilisée, c'est-à-dire la référence des caractéristiques du langage utilisé. Le tableau ci-dessous récapitule les déclarations pour les principales versions du langage HTML :

Version	Déclaration
HTML 2.0	<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
HTML 3.2	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
HTML 4.01	• **Strict** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd"> • **Transitional** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> • **Frameset** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0	• **Strict** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> • **Transitional** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-loose.dtd"> • **Frameset** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

3) Espaces, saut de ligne et tabulations

Le langage HTML ne tient pas compte des espaces, des tabulations et des sauts de ligne (ci-après appelés ou plus exactement il considère une suite d'un ou plusieurs espaces/tabulations/saut de ligne comme un seul espace. Cela permet notamment d'indenter le code HTML pour plus de lisibilité, sans modifier l'apparence de la page HTML dans le navigateur Il existe une exception pour le code contenu dans des balises, dont l'objectif est justement de conserver le formatage du texte (espaces, sauts de lignes, etc.)

N.B : il existe par ailleurs une exception pour le code contenu dans des balises <PRE>, dont l'objectif est justement de conserver le formatage du texte (espaces, sauts de lignes, etc.)

Les deux codes HTML suivant produisent donc le même résultat :

```
Apprendre le
HTML
```

```
Apprendre le HTML
```

Le langage HTML possède par contre des éléments permettant expressément de définir chacun de ces éléments de mise en forme :

- **Espace insécable** : il s'agit d'un espace ne pouvant être brisé par une fin de ligne. Sa représentation en HTML est *** *** .
- **Saut de ligne manuel** : il s'agit d'un saut de ligne explicite. Sa représentation en HTML est *
* (*
* pour être conforme au XHTML).

4) Les commentaires

Il est possible d'ajouter des éléments d'information dans une page web sans que ceux-ci soient affichés à l'écran grâce à un jeu de balises spécifique, appelé balises de commentaires :

```
<!-- Voici un commentaire -->
```

Les balises de commentaires permettent de mettre en commentaire du texte mais peuvent également servir à commenter du code HTML.

N.B : il existe une exception pour le code contenu dans des balises <PRE>, dont l'objectif est justement de conserver le formatage du texte (espaces, sauts de lignes, etc.)

5) Les caractères spéciaux

Les normes HTML demandent de respecter le codage des caractères ASCII 7 bits, c'est-à-dire que les caractères accentués ne sont pas autorisés.

Pourtant, les navigateurs actuels reconnaissent les caractères accentués, ainsi vous pouvez entrer des caractères accentués directement sous votre éditeur de texte, mais votre page sera vraisemblablement illisible dans la plupart des pays du monde ...

Pour coder un caractère accentué, on entre une combinaison précédée du caractère & et terminée par un point-virgule (;). Voici la liste représentations HTML des caractères ASCII de 128 à 255 :

Caractère	Code ISO	Code HTML
"	&#34;	&quot;
&	&#38;	&amp;
€	&#128;	&euro;
•	&#129;	
’	&#130;	
f	&#131;	
”	&#132;	
…	&#133;	
†	&#134;	
‡	&#135;	
ˆ	&#136;	
‰	&#137;	
Š	&#138;	
‹	&#139;	&lt;
Œ	&#140;	

•	&#141;	
Ž	&#142;	
•	&#143;	
•	&#144;	
‘	&#145;	
’	&#146;	
“	&#147;	
”	&#148;	
•	&#149;	
–	&#150;	
—	&#151;	
˜	&#152;	
™	&#153;	
š	&#154;	
›	&#155;	&gt;
œ	&#156;	&oelig;
•	&#157;	
ž	&#158;	
Ÿ	&#159;	&Yuml;
espace		
¡	&#161;	&iexcl;
¢	&#162;	&cent;
£	&#163;	&pound;
¤	&#164;	&curren;
¥	&#165;	&yen

¦	&#166;	&brvbar;
§	&#167;	&sect;
¨	&#168;	&uml;
©	&#169;	&copy;
ª	&#170;	&ordf;
«	&#171;	&laquo;
¬	&#172;	&not;
	&#173;	&shy;
®	&#174;	&reg;
¯	&#175;	&masr;
°	&#176;	&deg;
±	&#177;	&plusmn;
²	&#178;	&sup2;
³	&#179;	&sup3;
´	&#180;	&acute;
µ	&#181;	&micro;
¶	&#182;	&para;
·	&#183;	&middot;
¸	&#184;	&cedil;
¹	&#185;	&sup1;
º	&#186;	&ordm;
»	&#187;	&raquo;
¼	&#188;	&frac14;
½	&#189;	&frac12;
¾	&#190;	&frac34;

¿	&#191;	&iquest;
À	&#192;	&Agrave;
Á	&#193;	&Aacute;
Â	&#194;	&Acirc;
Ã	&#195;	&Atilde;
Ä	&#196;	&Auml;
Å	&#197;	&Aring;
Æ	&#198	&Aelig
Ç	&#199;	&Ccedil;
È	&#200;	&Egrave;
É	&#201;	&Eacute;
Ê	&#202;	&Ecirc;
Ë	&#203;	&Euml;
Ì	&#204;	&Igrave;
Í	&#205;	&Iacute;
Î	&#206;	&Icirc;
Ï	&#207;	&Iuml;
Ð	&#208;	&eth;
Ñ	&#209;	&Ntilde;
Ò	&#210;	&Ograve;
Ó	&#211;	&Oacute;
Ô	&#212;	&Ocirc;
Õ	&#213;	&Otilde;
Ö	&#214;	&Ouml;
×	&#215;	&times;

Ø	&#216;	&Oslash;
Ù	&#217;	&Ugrave;
Ú	&#218;	&Uacute;
Û	&#219;	&Ucirc;
Ü	&#220;	&Uuml;
Ý	&#221;	&Yacute;
Þ	&#222;	&thorn;
ß	&#223;	&szlig;
à	&#224;	&agrave;
á	&#225;	&aacute;
â	&#226;	&acirc;
ã	&#227;	&atilde;
ä	&#228;	&auml;
å	&#229;	&aring;
æ	&#230;	&aelig;
ç	&#231;	&ccedil;
è	&#232;	&egrave;
é	&#233;	&eacute;
ê	&#234;	&ecirc;
ë	&#235;	&euml;
ì	&#236;	&igrave;
í	&#237;	&iacute;
î	&#238;	&icirc;
ï	&#239;	&iuml;
ð	&#240;	&eth;

ñ	ñ	ñ
ò	ò	ò
ó	ó	ó
ô	ô	ô
õ	õ	õ
ö	ö	ö
÷	÷	÷
ø	ø	ø
ù	ù	ù
ú	ú	ú
û	û	û
ü	ü	ü
ý	ý	ý
þ	þ	þ
ÿ	ÿ	ÿ
"	"	"
&	&	&
‹	‹	<
›	›	>

6) Niveaux de titre

Il existe des balises spécialement conçues pour réaliser des titres : ce sont les tags <Hn> où n est un nombre entre 1 et 6. Ce sont des commandes d'en-têtes ; elles sont utilisées pour différencier les niveaux d'un document comme les sections d'un article de journal.

Code HTML	Aperçu dans le navigateur
<HTML> <HEAD> <TITLE>des titres</TITLE> </HEAD> <BODY> <H1>Titre 1</H1> <H2>Titre 2</H2> <H3>Titre 3</H3> <H4>Titre 4</H4> <H5>Titre 5</H5> <H6>Titre 6</H6> </BODY> </HTML>	**Titre 1** **Titre 2** **Titre 3** **Titre 4** **Titre 5** **Titre 6**

Remarque :
La taille de la police de caractères et les styles pour représenter les balises <Hn> ne sont pas uniformisées parmi les navigateurs. Il ne faut donc pas en abuser et préférer les balises <FONT>.

7) Les listes

Il existe 6 catégories de listes en HTML :

1. **Listes numérotées** *(Ordered Lists)* :
   Chaque ligne de texte est précédée généralement par un numéro. HTML renumérotera automatiquement la liste si vous modifiez l'ordre des éléments.
2. **Listes non ordonnées ou à puces** *(Numbered lists)* :
   Chaque élément est précédé par une pastille.
3. **Listes imbriquées** *(Nested lists)* :
   Vous pouvez insérer des sous-listes dans des listes.
4. **Listes de définitions** *(Definition list)* :
   Affiche une liste de mots avec leur définition en-dessous.
5. **Listes de répertoires** *(Directory lists)* :
   Une liste de répertoires apparait en plusieurs colonnes sur une page.
6. **Liste de menus** *(Menu list)* :
   Les éléments dans une liste de menus sont justifiés à gauche l'un en-dessous de l'autre, et ne sont pas indentés.

Description	Commandes	Visualisation
Liste ordonnée	<OL> <LI>Element1 <LI>Element2 </OL>	1. Element1 2. Element2
Liste non ordonnée	<UL> <LI>Element1 <LI>Element2 </UL>	• Element1 • Element2
Liste de répertoires	<DIR> <LI>Element1 <LI>Element2 </DIR>	• Element1 • Element2
Liste de menus	<MENU> <LI>Element1 <LI>Element2 </MENU>	• Element1 • Element2
Liste de définition	<DL> <DT>Element1 <DD>description1 <DT>Element2 <DD>description2 </DL>	Element1 description1 Element2 description2
Liste imbriquées	<OL> <LI>ElementA <UL> <LI>Element1 <LI>Element2 </UL> </OL>	1. ElementA • Element1 • Element2

Remarques :

- Pour les listes non ordonnées, il est possible de remplacer les puces par un autre symbole en utilisant l'option TYPE dans la balise <UL>.

par exemple : <UL TYPE=*disc*,*circle* ou *square*> ... </UL>

- Pour les listes ordonnées, il est possible de changer le type de numérotation avec l'option TYPE.On aura ainsi soit des lettres, soit des chiffres romains.
par exemple : <OL TYPE=*A*,*a*,*I* ou *i*> ... </OL>
Il est aussi possible de commencer l'incrémentation à la valeur voulue grâce à la commande START.
par exemple : <OL START=4> ... </OL> commencera la liste au quatrième élément.
- Il est possible de donner un titre à une liste avec la commande <LH> qui se place avant la balise de début de liste. Par exemple : <UL><LH>Titre de la liste> ... </UL>

8) Les tableaux

L'affichage de tableaux dans des documents codés en HTML peut sembler complexe, mais ils se révèlent très utiles dans de nombreux cas.

Les tableaux sont constitués de cellules, situées dans des colonnes (éléments verticaux) et des rangées (éléments horizontaux). Les cellules peuvent contenir tous les éléments Html déjà passés en revue (texte, images, liens, arrière-plans, tableaux).

a) Les balises de tableaux

1. **<TABLE>...<TABLE>** :
Permet d'ouvrir un tableau
2. **<TR>...<TR>** ***(table Row)*** :
Permet de définir une rangée du tableau.
3. **<TD>...<TD>** ***(table Data)*** :
Permet d'insérer des éléments qui seront affichés dans la cellule.
4. **<TH>...<TH>** ***(Table Header)*** :
Affiche un titre de colonne ou de rangée qui apparait centré en gras sans une cellule donnée.
5. **<CAPTION>...<CAPTION>** :
insère un titre global au tableau.

b) Options du tableau

- **BORDER** : option de la balise <TABLE>. Epaisseur du cadre extérieur en pixels. Avec la valeur 0, le cadre est invisible.
- **CELLSPACING** : option de la balise <TABLE>. Epaisseur en pixels autours de chaque cellule.
- **CELLSPADDING** : option de la balise <TABLE>. Epaisseur en pixels entre l'élément de la cellule et le cadre.
- **WIDTH** : option de la balise <TABLE>. Largeur occupée par le tableau en pixels ou en pourcentages.

- **WIDTH** : option de la balise <TD>. Largeur occupée par une colonne en pixels ou en pourcentages.
- **ALIGN** : option de la balise <CAPTION>. pour la valeur TOP, affiche le titre au-dessus ; BOTTOM, affiche en-dessous.
- **ALIGN** : option des balises <TD>,<TR>,<TH>. pour les valeurs LEFT, RIGHT et CENTER, affiche les éléments de la cellule à gauche, à droite ou au centre.
- **VALIGN** : option des balises <TD>,<TR>,<TH>. Aligne le contenu de la cellule en haut, en bas ou au milieu pour les valeurs TOP, BOTTOM et MIDDLE.
- **NOWRAP** : option des balises <TD>, <TH>. Empêche le retour à la ligne dans la cellule.
- **COLSPAN** : option de la balise <TD>, <TH>. prend pour valeur un chiffre qui est le nombre de colonnes qu'occupera une seule et unique cellule.
- **ROWSPAN** : option de la balise <TD>, <TH>. prend pour valeur un chiffre qui est le nombre de lignes qu'occupera une seule et unique cellule.

Faisons un exemple pour être + clair.

Prenons le code suivant :

```
<TABLE BORDER=1 CELLSPACING=2 CELLPADDING=1>
<CAPTION ALIGN=TOP>titre du tableau</CAPTION>
<TR ALIGN=LEFT VALIGN=TOP>
<TH COLSPAN=3 ROWSPAN=1 NOWRAP>Elements A</TH>
<TH>Elements B</TH>
</TR>
<TR>
<TD>Element A1</TD>
<TD WIDTH="33%">Element A2</TD>
<TD>Element A3</TD>
<TD>Element B1</TD>
</TR>
</TABLE>
```

Cela donne :

titre du tableau

Elements A			**Elements B**
Element A1	Element A2	Element A3	Element B1

Les tableaux, c'est tout simple ! Pour qu'il soit encore plus clair, on peut rajouter de la couleur en utilisant l'option BGCOLOR. par exemple : <TD BGCOLOR="#C0C0C0"> met en gris le fond de la cellule.

9) Les liens hypertextes

Html (Hyper Text Markup Language) est un langage hypertexte (et hypergraphique) qui vous permet en cliquant sur un mot, généralement souligné (ou une image) de vous transporter :

- Vers un autre endroit du document.
- Vers un autre fichier html situe sur votre ordinateur.
- Vers un autre ordinateur situe sur le web.

L'hypertexte permet, dans le cas qui nous intéresse, de faire des choix par le simple clic de la souris. Un sujet vous intéresse en particulier ? Cliquez ici pour accéder à un autre document correspondant à ce sujet. L'hypertexte réseau consiste en un livre éclaté. On peut ainsi pointer des textes, des images, du son, de la vidéo ...

a) Créer un lien

La commande qui permet de réaliser des liens hypertextes est toujours la même, <A>. Il faut toutefois respecter la syntaxe correcte ; ainsi il faut connaitre l'URL (l'adresse) exacte d'un document.

b) Le lien externe

Tout ordinateur situé sur le réseau Internet possède une adresse ou un URL (Universal Ressource Locator). Html permet d'accéder à toutes les machines et toutes les ressources du Net. Pour peu qu'Internet vous soit un peu familier, ce sont les adresses du type :
http://www.yahoo.fr
http://www.minfopra.gov.cm/carto

Elles commencent toujours par le protocole *http://* pour Internet. Ainsi pour pointer vers le site Yahoo, il suffit de taper :
<A HREF="http://www.yahoo.fr"> page yahoo </A>

c) Le lien local :

L'organisation classique, et plus que conseillée, d'un site Web consiste à regrouper l'ensemble des éléments de celui-ci (fichiers htm, images, ...) dans un même répertoire. Vous pourrez ainsi "transporter" aisément votre site pour le présenter sur un autre ordinateur et but ultime, le charger sur un serveur. Cette façon de procéder est la plus aisée et vous évitera pas mal de problèmes. Le code à écrire est donc par exemple :
<A HREF="chemin_acces/page.htm"> page locale </A>

d) Le lien interne :

Des liens peuvent aussi pointer vers un endroit précis du même document ou d'un autre fichier. C'est ce qu'on appelle les ancres, ancrages ou pointeurs [Anchor].

Point d'ancrage	<A NAME="***"> ... </A>	Ceci est une cible
Lien vers une ancre dans la même page	<A HREF="#***"> ... </A>	Lien vers la cible *** dans la même page
Lien vers une ancre dans une autre page	<A HREF="URL#***"> ... </A>	Lien vers la cible *** dans une autre page

Plusieurs liens à l'intérieur d'un même document supposent que ce document présente une certaine longueur sinon une longueur certaine (et donc un temps de chargement assez long). Ainsi, on préférera généralement à cette technique le découpage d'un longue page en un ensemble de plusieurs pages de dimension plus réduite.

Remarques :

- pour modifier la couleur des liens hypertextes, il faut modifier la balise <BODY> en ajoutant des options TEXT,LINK,VLINK et ALINK pour le texte de la page, les liens, les liens déjà cliqué et les liens au moment du clic.
Par exemple : <BODY TEXT="#000000" LINK="#FF9900" VLINK="#66FF99" ALINK="#66FF99>

10) Les images

a) Afficher une image

C'est la commande <IMG> qui permet d'afficher des images au sein d'une page codée en HTML et de la positionner à l'endroit voulu dans le texte. Cette commande n'a pas besoin d'être fermée.

Par exemple, pour afficher l'image logomain.gif,on utilise la commande : <IMG SRC="../logomain.gif">

Par défaut, le bas de l'image est aligné avec le bas du texte. Ce sont les options qui vont permettre d'aligner les images comme désiré.

- **SRC**
  Cette option contient l'adresse ou est stockée l'image. En effet, en Html, l'image ne fait pas partie de votre document. Le browser va la chercher à l'adresse indiquée. Généralement, on place les images dans le même répertoire que les pages Html. On peut donc utiliser la même image plusieurs fois dans un même document.

- **ALT**
  Cette option permet d'afficher un texte à la place de l'image quand elle n'a pas encore été chargée. Cela est très utile pour des images lourdes. <IMG SRC="../logomain.gif" ALT="logo">

- **WIDTH et HEIGHT**
  Ce sont la largeur et la hauteur de l'image en pixels. Cela permet la fluidité de l'affichage car le navigateur doit connaitre l'emplacement à réserver à l'image pour pouvoir continuer à afficher le texte. <IMG SRC="../logomain.gif" HEIGHT=65 WIDTH=66>
- **BORDER**
  Taille du cadre autour de l'image en pixels (par défaut 1). Cela permet de mettre en valeur des images quand elles sont dans un lien. <IMG SRC="../logomain.gif" BORDER=0>

- **ALIGN**
  Alignement de l'image. prend les valeurs LEFT, RIGHT, MIDDLE, TOP et BOTTOM. On préfère utiliser cette balise pour centrer l'image verticalement ; pour centrer horizontalement, on encadre l'image par les balises <CENTER><IMG SRC="../logomain.gif" ALIGN=TOP><CENTER>

- **VSPACE – HSPACE**
  Espace vertical et espace horizontal. Permet de déterminer en pixels l'espace laissé libre autour de l'image.

b) Lien sur une image

Il est sympathique de pouvoir cliquer sur des images pour avancer dans un site. Pour cela, il suffit de combiner les balises vues précédemment :

<A HREF="http://www.yahoo.fr"><IMG SRC="../logomain.gif" ALT="logo" HEIGHT=65 WIDTH=66></A>

On remarque que l'image est entourée d'une bordure. Pour la supprimer, on utilise l'option BORDER=0.

<A HREF="http://www.yahoo.fr"><IMG SRC="../logomain.gif" BORDER=0 ALT="logo" HEIGHT=65 WIDTH=66></A>

c) Les arrières plans

Il est possible d'afficher une couleur ou une image en guise de fond sur votre page.

- **couleur de fond**
  Il suffit de modifier la balise <BODY> en <BODY BGCOLOR>. L'option BGCOLOR est suivie dans une couleur au format RGB (Ce code est constitué d'un dièse (#) suivi de trois nombres au format hexadécimal compris entre 00 et FF.) Par exemple : < BODY BGCOLOR="#555555"> ... <BODY> définit un fond gris.
- **image de fond**
  Semblable à la commande <BODY BGCOLOR>, la commande <BODY BACKGROUND> permet d'afficher une image qui sera répétée sans cesse en motif de fond.
  Il faut comprendre que cette image doit être enregistrée auparavant dans un des deux formats, JPEG ou GIF. Aussi, il faut veiller à ce qu'elle soit d'une couleur approprié afin que le texte qui viendra s'ajouter par-dessus puisse être lisible malgré tout.

<BODY BACKGROUND> doit être placée au tout début de la section <BODY>, remplaçant cette dernière. Par exemple < BODY BACKGROUND="*fichier*"> ... </BODY>

il est conseiller de combiner les deux options pour que le navigateur affiche une couleur le temps qu'il charge l'image : <BODY BACKGROUND="fond.jpg BGCOLOR="FFFFFF">

Les attributs de la balise <BODY> sont :

Attribut	Effet Visuel
BACKGROUND="image"	Affiche l'image en arrière-plan
BGCOLOR="nom_de_la_couleur ou #XXXXXX"	Affiche la couleur demandée en arrière-plan
LINK="couleur"	Couleur des liens hypertexte
ALINK="couleur"	Couleur du lien actif
VLINK="couleur"	Couleur des liens déjà visités
TEXT="couleur"	Couleur du texte par défaut

Les couleurs sont définies selon la notation #RRVVBB, où RR, VV et BB représentent respectivement un nombre hexadécimal entre 00 et FF pour le Rouge, le Vert et le Bleu.

11) Les couleurs

Les couleurs en HTML sont définies par 3 nombres hexadécimaux représentant les tons de Rouge, de Vert et de Bleu (selon le codage RGB (Red Green Blue, en français : RVB) de la couleur choisie. Ainsi la syntaxe de codage d'une couleur en HTML est la suivante :

couleur="#RRVVBB"

RR, VV et BB représentent respectivement un nombre hexadécimal entre 00 et FF pour le Rouge, le Vert et le Bleu.

Ainsi, plus de 16 millions de couleurs sont disponibles pour colorer les pages web. Toutefois, étant donné que tous les navigateurs ne reconnaissent pas les couleurs de la même façon, le W3C conseille l'utilisation des couleurs ci-dessous, pour lesquelles un nom intelligible a été donné. Il est donc possible (et conseillé) d'appeler une couleur de la façon suivante :

couleur="nom_de_la_couleur"

N.B : couleur représente le nom d'un attribut de couleur de n'importe quelle balise le prenant en compte.

La table des couleurs usuelles est donc la suivante :

Nom de la couleur	Codage RVB
aliceblue	#F0F8FF
antiquewhite	#FAEBD7
aqua	#00FFFF
aquamarine	#7FFFD4
azure	#F0FFFF
beige	#F5F5DC
bisque	#FFE4C4
black	#000000
blanchedalmond	#FFEBCD
blue	#0000FF
blueviolet	#8A2BE2
brown	#A52A2A
burlywood	#DEB887
cadetblue	#5F9EA0
chartreuse	#7FFF00
chocolate	#D2691E
coral	#FF7F50
cornflowerblue	#6495ED
cornsilk	#FFF8DC
crimson	#DC143C
cyan	#00FFFF
darkblue	#00008B
darkcyan	#008B8B
darkgoldenrod	#B8860B

darkgray	#A9A9A9
darkgreen	#006400
darkkhaki	#BDB76B
darkmagenta	#8B008B
darkolivegreen	#556B2F
darkorange	#FF8C00
darkorchid	#9932CC
darkred	#8B0000
darksalmon	#E9967A
darkseagreen	#8FBC8F
darkslateblue	#483D8B
darkslategray	#2F4F4F
darkturquoise	#00CED1
darkviolet	#9400D3
deeppink	#FF1493
deepskyblue	#00BFFF
dimgray	#696969
dodgerblue	#1E90FF
firebrick	#B22222
floralwhite	#FFFAF0
forestgreen	#228B22
fuchsia	#FF00FF
gainsboro	#DCDCDC
ghostwhite	#F8F8FF
gold	#FFD700

goldenrod	#DAA520
gray	#808080
green	#008000
greenyellow	#ADFF2F
honeydew	#F0FFF0
hotpink	#FF69B4
indianred	#CD5C5C
indigo	#4B0082
ivory	#FFFFF0
khaki	#F0E68C
lavender	#E6E6FA
lavenderblush	#FFF0F5
lawngreen	#7CFC00
lemonchiffon	#FFFACD
lightblue	#ADD8E6
lightcoral	#F08080
lightcyan	#E0FFFF
lightgoldenrodyellow	#FAFAD2
lightgreen	#90EE90
lightgrey	#D3D3D3
lightpink	#FFB6C1
lightsalmon	#FFA07A
lightseagreen	#20B2AA
lightskyblue	#87CEFA
lightslategray	#778899

lightsteelblue	#B0C4DE
lightyellow	#FFFFE0
lime	#00FF00
limegreen	#32CD32
linen	#FAF0E6
magenta	#FF00FF
maroon	#800000
mediumaquamarine	#66CDAA
mediumblue	#0000CD
mediumorchid	#BA55D3
mediumpurple	#9370DB
mediumseagreen	#3CB371
mediumslateblue	#7B68EE
mediumspringgreen	#00FA9A
mediumturquoise	#48D1CC
mediumvioletred	#C71585
midnightblue	#191970
mintcream	#F5FFFA
mistyrose	#FFE4E1
moccasin	#FFE4B5
navajowhite	#FFDEAD
navy	#000080
oldlace	#FDF5E6
olive	#808000
olivedrab	#6B8E23

orange	#FFA500
orangered	#FF4500
orchid	#DA70D6
palegoldenrod	#EEE8AA
palegreen	#98FB98
paleturquoise	#AFEEEE
palevioletred	#DB7093
papayawhip	#FFEFD5
peachpuff	#FFDAB9
peru	#CD853F
pink	#FFC0CB
plum	#DDA0DD
powderblue	#B0E0E6
purple	#800080
red	#FF0000
rosybrown	#BC8F8F
royalblue	#4169E1
saddlebrown	#8B4513
salmon	#FA8072
sandybrown	#F4A460
seagreen	#2E8B57
seashell	#FFF5EE
sienna	#A0522D
silver	#C0C0C0
skyblue	#87CEEB

slateblue	#6A5ACD
slategray	#708090
snow	#FFFAFA
springgreen	#00FF7F
steelblue	#4682B4
tan	#D2B48C
teal	#008080
thistle	#D8BFD8
tomato	#FF6347
turquoise	#40E0D0
violet	#EE82EE
wheat	#F5DEB3
white	#FFFFFF
whitesmoke	#F5F5F5
yellow	#FFFF00
yellowgreen	#9ACD32

12) Les balises META

Les balises META sont des balises très particulières du langage HTML. Elle permet notamment de référencer votre site sur des moteurs de recherche tels que Altavista ou Hotbot. Quand on sait que plus de la moitié des utilisateurs du Web passent par des moteurs de recherche, il est important de soigner le référencement de son site pour y apparaître de façon correcte et si possible en rang utile.

Comme ces balises apportent des informations sur le contenu de la page, il faut qu'elles soient placées dans la zone d'en-tête. Les balises META doivent donc se situer entre les balises <HEAD> et </HEAD>.

La liste des balises META est la suivante :

1. **<META NAME="keywords" CONTENT="mot-clé1,mot-clé2,mot-clé3,...">**
   Cette balise permet d'indiquer les principaux mots-clés qui se réfèrent à la page. Les mots-clés ne doivent pas être séparés par des espaces mais juste des virgules. On peut mettre jusqu'à 1000 mots ; c'est largement suffisant pour décrire la page. Les majuscules et minuscules ont une importance pour les moteurs de recherche ; il faut donc préférer mettre les mots courants en minuscule. La difficulté est de trouver les mots correspondants à la page. Mettez-vous à la place de vos lecteurs potentiels. Quels mots, quels synonymes, quelles alternatives peuvent être utilisés pour décrire votre site?

2. **<META NAME="description" CONTENT="une brève description de la page">**
   La description doit décrire le plus fidèlement possible le contenu du site. Vous pouvez utiliser jusqu'à 150 mots.

3. **<TITLE>...</TITLE>**
   Bien qu'elle n'en ait pas l'air, cette balise est sûrement la plus important car c'est la première d'un lecteur potentiel lira. Le titre ne doit pas excéder 8 mots ; soyez donc concis. On dit également que le fait de reprendre un ou des mot(s)-clé(s) dans le titre de toutes les pages d'un site est très favorable pour un meilleur classement.

4. **<META NAME="author" CONTENT="nom de l'auteur">**
   Cette balise est très peu utilisée par les moteurs de recherche. Seul Nomade indique le nom de l'auteur. Néanmoins qu'elle fierté de signer une page !

5. **<META NAME="Copyright" CONTENT="Copyright © date nom">**
   Cette balise est peu utilisée mais invite à respecter le copyright lorsqu'elle est présente.

6. **<META NAME="Generator" CONTENT="nom de l'éditeur Html utilisé">**
   Cette balise n'a aucun intérêt pour le référencement sur les moteurs de recherche. Elle est seulement utile pour les sociétés des éditeurs de logiciel HTML.

7. **<META NAME="Distribution" CONTENT="Global ou Local">**
   Indique si le document est largement diffusé (Global) ou s'il est à diffusion restreinte (local).

8. **<META NAME="Rating" CONTENT="Destination de votre audience">**
   Permet de définir le contenu de votre site. Les appréciations sont General ou Mature ou Restricted ou 14 years pour respectivement tout public, adulte, accès restreint ou 14 ans.

9. **<META NAME="Robots" CONTENT="instructions pour les robots">**
   Par cette balise vous pouvez indiquez aux robots de recherche automatique si vous souhaitez que votre site soit ou ne soit pas indexé par eux.
   Les instructions sont :
   - *All* (défaut) permet aux robots d'indexer vos pages et de suivre les liens hypertextes d'une page à l'autre.
   - *None* dira aux robots de ne pas indexer vos pages et de ne pas suivre les liens.

- *Index* indique que vos pages peuvent être indexées par les robots.
- *NoIndex* pour que le robot ne procèdent à aucune indexation.
- *Follow* donne la permission aux robots de suivre les liens hypertextes des pages
- *NoFollow* pour le contraire.

Exemples :

<META NAME="Robots" CONTENT="Index,NoFollow">

10. **<META HTTP-EQUIV="Content-language" CONTENT="fr">**
    Cette balise déclare la langue utilisée dans le document Html. Elle est très utile maintenant que des moteurs de recherche anglophones ont inclus la langue dans leurs critères de recherche.

11. **<META HTTP-EQUIV="Reply-to" CONTENT="votre adresse e-mail ou URL de départ">**
    Permet aux internautes qui ont enregistré votre page de vous recontacter.

12. **<META HTTP-EQUIV="Refresh" CONTENT="x;URL="adresse">**
    Cette balise META charge automatiquement la page spécifiée à l'attribut URL après un délai de x secondes. Elle est fréquemment utilisée pour redirectionner un visiteur lors que l'adresse du site a été modifiée.

13. **<META HTTP-EQUIV="expires" CONTENT="Sat, 01 Apr 2000 13:15:00 GMT">**
    Cette balise META "dit" au navigateur la date à laquelle la page Html doit être considérée comme périmée. C'est aussi la date à laquelle votre page va être supprimée de la base de données des moteurs de recherche.

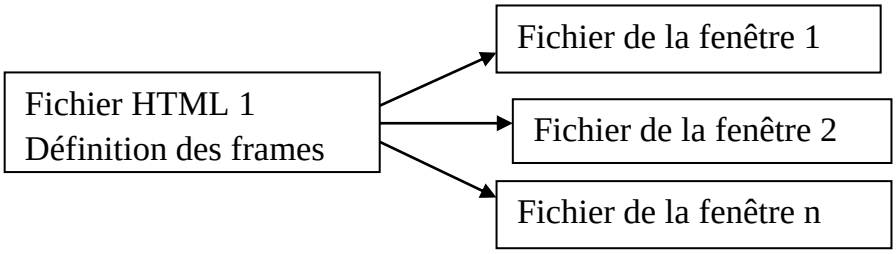
14. **<META HTTP-EQUIV="Pragma" CONTENT="no-cache">**
    Si vous introduisez cette petite ligne dans votre page, youpla boum, Netscape de gardera pas en local une version de votre page, cela le forcera donc à charger la page à chaque fois et vous serez sûr que le visiteur verra la bonne page ! (Attention c'est plus lent aussi...)

15. **<META HTTP-EQUIV="Window-target" CONTENT="_top">**
    Ça c'est très pratique ! Cela force le browser à charger la page dans une nouvelle fenêtre. (Pour votre page d'accueil par exemple)

III) Les frames

Il existe deux façons de structurer un site : utiliser un tableau dans lequel une colonne reprend le contenu du site ou utiliser des frames. Ils permettent de diviser l'écran en fenêtres. Les frames ont l'avantage évident que l'on peut facilement rajouter des pages ou modifier le site facilement ; il n'y qu'une seule page à changer. Contrairement aux tableaux où il faut changer chaque page. Mais il faut aussi dire que les frames sont délicats et dangereux à utiliser (risques de plantage) et tous les browsers n'ont pas la possibilité de les afficher.

Il faut comprendre que le fichier HTML dans lequel vous allez définir vos fenêtres grâce aux Frames est indépendant des autres fichiers. Il va diviser votre écran en fenêtres et appeler d'autres fichiers dans ces fenêtres.



Il n'y aura donc pas qu'un fichier mais plusieurs, selon le nombre de fenêtres dans l'écran.

Les balises permettant de définir des frames sont les suivantes :

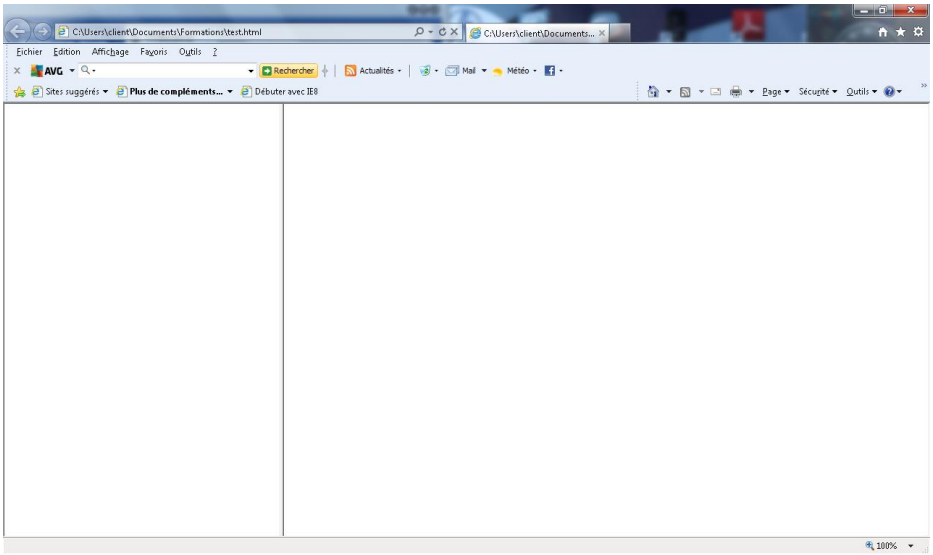
Zone avec des fenêtres	
<FRAMESET>...</FRAMESET>	Début et fin de zone avec des fenêtres
<FRAME>	Définition d'une fenêtre.
Agencement des fenêtres	
<FRAMESET ROWS="...">	Fenêtres horizontales
<FRAMESET COLS="...">	Fenêtres verticales

Exemple 1 :

Soit le code :

```
<HTML>
<HEAD></HEAD>
<FRAMESET COLS="30%,70%">
<FRAME>
<FRAME>
</FRAMESET>
</HTML>
```

Dans notre browser, on obtient :



Attention! <FRAMESET></FRAMESET> remplace <BODY></BODY>.

L'attribut COLS="largeur1,largeur2,...,largeurN" définit la largeur des différentes fenêtres. La largeur s'exprime en pixels ou en %. Dans ce cas, on veillera à ce que le total soit égal à 100%.
Il est aussi possible d'utiliser l'astérix "*" pour définir une taille quelconque. Cela permet d'avoir une taille fixe pour une fenêtre et une mobile pour une autre. Par exemple, <FRAMESET COLS="50,*"> définit une colonne de 50 pixels et une autre variable.

Exemple 2

Jusqu'à maintenant nous frames étaient vides ; nous allons les remplir avec l'attribut <FRAME SRC="URL du document HTML à afficher ">.

On construit trois fichiers A.html, B.html, C.html que l'on place dans le même répertoire le fichier de Frames. Ces fichiers sont tous de la forme:

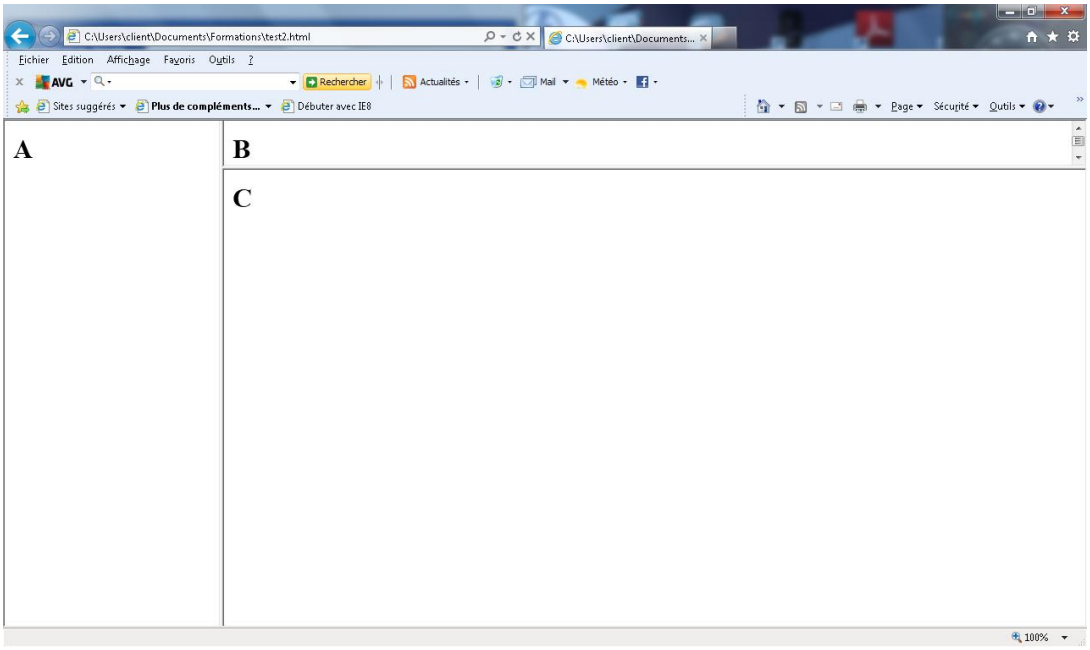
```
<HTML>
<BODY><H1>A</H1></BODY>
</HTML>
```

Pour le fichier de Frames, on définit le code suivant :

```
<HTML>
<HEAD></HEAD>
<FRAMESET COLS="20%,80%">
<FRAME SRC="A.htm">
<FRAMESET ROWS="30%,70%">
<FRAME SRC="B.htm">
<FRAME SRC="C.htm">
</FRAMESET>
```

```
</FRAMESET>
</HTML>
```

Dans notre browser, on obtient :



Dans la fenêtre B, des ascenseurs sont apparus. On peut les supprimer ou les rendre obligatoires en utilisant l'attribut SCROLLING dans la balise <FRAME> : <FRAME SCROLLING="yes/no/auto">. Par défaut, la barre de défilement est sur auto.

Les attributs de la balise frames sont les suivants :

- **l'attribut <FRAME NAME="Nom">**
  Name indique le nom de la fenêtre de telle sorte que cette frame puisse être utilisée comme cible d'un lien hypertexte. Ainsi pour ouvrir un document dans cette fenêtre, on utilisera l'attribut TARGET dans la balise de lien. Par exemple : <A HREF="doc.htm" TARGET="Nom">...</A>
  L'attribut TARGET peut aussi prendre certaines valeurs prédéfinies :

  - **_blank** qui indique au browser qu'il doit créer une nouvelle fenêtre afin d'y afficher le fichier. Dans ce cas, vous ouvrez en fait un nouveau browser.
  - **_self** qui indique que le fichier sera chargé dans la même fenêtre que celle dans laquelle se trouve le lien.
  - **_top** qui implique l'affichage du fichier sur toute la surface de la fenêtre du browser.

- **l'attribut <FRAME FRAMEBORDER=no FRAMESPACING=0 BORDER=0>** permet de supprimer les bordures qui séparent les fenêtres. Netscape utilise l'attribut "border=0" et Explorer, les attributs "frameborder=no" et "framespacing=0". Il faut donc mettre les trois pour avoir la même configuration sur tous les navigateurs.

- **l'attribut <FRAME NORESIZE>**
  Cet attribut empêche le lecteur de pouvoir modifier la taille des cadres avec sa souris.
- **la balise <NOFRAMES>...</NOFRAMES>**
  Il convient de mettre cette balise après la définition de vos Frames car elle permet d'indiquer le texte, que les vieux navigateurs incapables de gérer les frames, doivent afficher. Il est convenable d'indiquer une page pour que ces visiteurs puissent quand même voir votre site.

IV) Les formulaires

Les formulaires interactifs permettent aux auteurs de pages Web de doter leur page web d'éléments interactifs permettant par exemple un dialogue avec les internautes, à la manière des coupons-réponses présents dans certains magazines.

Le lecteur saisit des informations en remplissant des champs ou en cliquant sur des boutons, puis appuie sur un bouton de soumission (submit) pour l'envoyer soit à un URL, c'est-à-dire de façon générale à une adresse e-mail ou à un script de page web dynamique tel que PHP, ASP ou un script CGI.

1) La balise FORM

Le contenu d'un formulaire est compris entre les balises <FORM>... </FORM>.

Le dernier élément d'un formulaire est un bouton d'envoi du formulaire (type =submit ou button).

Les attributs de l'élément <FORM> :

- **id** (ou **name**) nom du formulaire pour les scripts JavaScript.
- **action** donne l'URL du programme auquel est envoyé le contenu du formulaire (en général un programme cgi).
- **method** indique sous quelle forme seront envoyées les réponses « *POST* » est la valeur qui correspond à un envoi de données stockées dans le corps de la requête, tandis que « *GET* » correspond à un envoi des données codées dans l'URL, et séparées de l'adresse du script par un point d'interrogation.
- **enctype** par défaut application/x-www-form-urlencoded

2) Les champs de saisie (balise INPUT)

<INPUT> (pas de balise de fermeture, ce n'est pas un container)

Attributs :

- **Type** :
  - **text** (défaut) : zone de saisie d'une ligne, on précise la taille par size=xx (en nombre de caractères). Exemple :
    <input type=text size=40 name="user" value="votre nom">
  - **password** : le contenu est masqué par des points sur l'écran. Exemple :
    <input type=password size=12 name="pw">
  - **checkbox** : chaque boite à cocher génère une paire nom/valeur. Exemple :
    <input type=checkbox checked name="etudiant" value="oui">
  - **radio** : permet de choisir une valeur parmi plusieurs. (l'ensemble des boutons radios devant porter le même attribut *name)* Exemple :
    <input type=radio name=age value="0-20">
    <input type=radio name=age value="20-40" checked>
    <input type=radio name=age value="40+">
  - **submit :** envoie à l'URL définit dans l'attribut action le contenu du formulaire sous la forme de paires nom/valeur. il s'agit du *bouton de soumission* permettant l'envoi du formulaire. Le texte du bouton est indiqué par l'attribut **value**.
  - **Reset :** replace les champs du formulaire dans leur état initial.
  - **File :** permet d'attacher un fichier au contenu du formulaire (apparaît comme un champ de texte, avec un bouton de balayage pour désigner le fichier à attacher.
  - **Hidden :** champ caché, qui sert au suivi de session. Exemple :
    <input type=hidden name="utilisateur" value="0123-542">

3) Les champs de sélection sur liste (balise SELECT)

<SELECT> menu de sélection au sein d'un formulaire. Il permet de faire apparaître un menu déroulant.

Exemple :

<SELECT name="couleur">

<OPTION value="rouge">Rouge

<OPTION value="vert">Vert

<OPTION value="bleu">Bleu

</SELECT>

Attributs de l'élément de sélection :

- **Name :** Représente l'identifiant du champ.
- **Size :** représente le nombre de lignes dans la liste (cette valeur peut être plus grande que le nombre d'éléments effectifs dans la liste).
- **disabled**: permet de créer une liste désactivée, c'est-à-dire affichée en grisée .
- **multiple**: marque la possibilité pour l'utilisateur de choisir plusieurs champs dans la liste.

Attributs de l'élément OPTION

- **value :** contient l'information qui est envoyée au programme de traitement du formulaire (la paire nom de la sélection, value de l'option choisie).
- **Selected** : option qui apparaît dans le sélecteur avant toute action de l'utilisateur

4) Les champs de texte (balise TEXTAREA)

<TEXTAREA> champ de texte comportant plusieurs lignes.

Attributs de l'élément <TEXTAREA> :

- **cols**: représente le nombre de caractères que peut contenir une ligne.
- **rows**: représente le nombre de lignes.
- **name**: représente le nom associé au champ, c'est le nom qui permettra d'identifier le champ dans la paire nom/valeur.
- **readonly**: permet d'empêcher l'utilisateur de modifier le texte entré par défaut dans le champ.
- **value**: représente la valeur qui sera envoyée par défaut au script si le champ de saisie n'est pas modifié par une frappe de l'utilisateur.

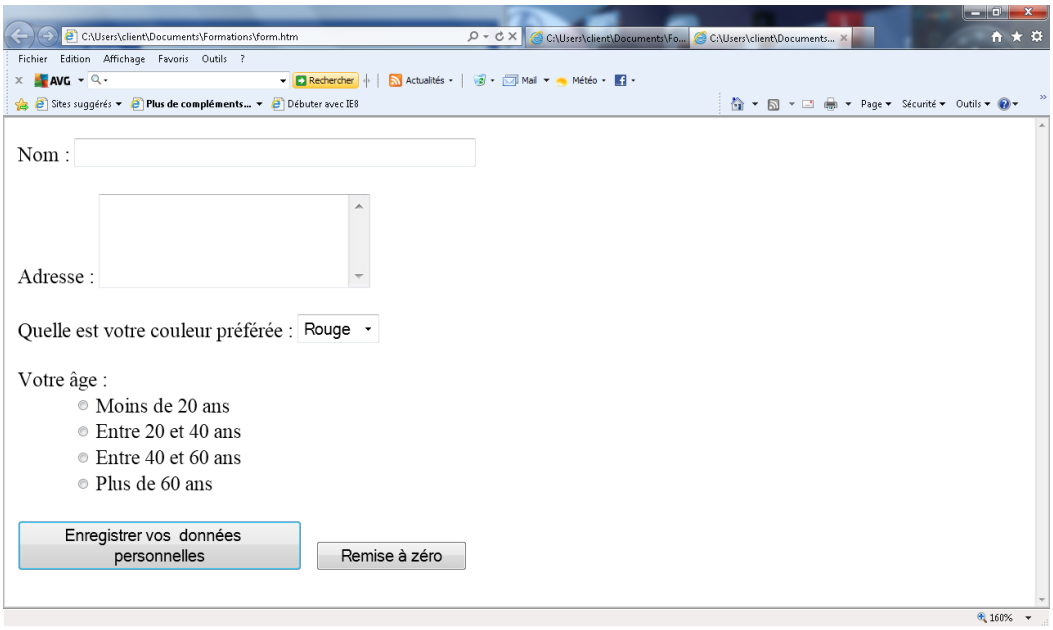
Le texte d'une zone de texte est compris entre la balise <TEXTAREA> et la balise fermante (</TEXTAREA>)

5) Un exemple de formulaire

```
<HTML>
 <BODY>
    <FORM name = "form1"action = "test.cgi" method = post>
      <P>Nom : <INPUT type=text size="50" name="nom" value=""> </P>
      <P>Adresse :
      <TEXTAREA name="adresse" rows=5 cols=25
border=0> </TEXTAREA></P>
      <P>Quelle est votre couleur préférée :
```

```
<SELECT name="couleur">
  <OPTION value="rouge" selected>Rouge
  <OPTION value="vert">Vert
  <OPTION value="bleu">Bleu
 </SELECT>
<DL><DT>Votre âge : </DT>
<DD><INPUT type=radio name="age" value="1">Moins de 20 ans <BR>
  <INPUT type=radio name="age" value="2">Entre 20 et 40 ans <BR>
  <INPUT type=radio name="age" value="3">Entre 40 et 60 ans <BR>
  <INPUT type=radio name="age" value="4">Plus de 60 ans </DD>
</DL>
 <P><INPUT type=submit name="go" value="Enregistrer vos  données
personnelles">
  <INPUT type=reset name="raz" value="Remise à zéro"></P>
 </FORM>
 </BODY>
</HTML>
```

Nous obtenons dans notre browser :



V) Les feuilles de style : CSS

1) Syntaxe

La syntaxe du CSS est très simple :

- Un **sélecteur de balises**, permettant de préciser à quelles balises du document le style s'applique ;

- Une **déclaration** de style, définie entre accolades, permettant de préciser le style à appliquer aux balises sélectionnées. La déclaration est elle-même constituée :
    - d'une ou plusieurs **propriété**(s), suivie du caractère « : » (double point),
    - d'une ou plusieurs **valeur**(s) associée(s) à chaque propriété, entourée de guillemets s'il s'agit de plusieurs mots ou séparés par des virgules s'il y en a plusieurs, suivie d'un point virgule.

On obtient donc :

selecteur { propriété : valeur }

Exemple : body { background: #eeeeee; }

Chaque sélecteur (ici body) peut avoir plusieurs propriétés avec des valeurs indépendantes.

Exemple :

```
body {
 background: #eeeeee;
 font-family: Trebuchet MS, Verdana, Arial;
}
```

On remarquera un point virgule entre chaque propriétés.

a) Sélection multiple

Il est également possible d'appliquer le style à plusieurs balises en séparant le nom de ces balises par une virgule (,). La syntaxe d'un tel sélecteur, appelé sélecteur multiple, est la suivante :

selecteur-de-balise1, selecteur-de-balise2 { /* style */ }

b) Sélection Universelle

Grâce au sélecteur universel (« * ») il est possible de définir un style s'appliquant à tous les éléments HTML. La syntaxe du sélecteur universel est la suivante :

*** { /* style */ }**

c) Sélection d'éléments imbriqués

Il est possible de sélectionner une balise dans un contexte donné, c'est-à-dire en fonction des éléments qui l'entourent, grâce aux sélecteurs **sélecteurs contextuels**.

Il existe plusieurs types de sélecteurs contextuels :

- Le **sélecteur d'éléments imbriqués** permet de créer une règle ne s'appliquant que lorsqu'un élément Y est imbriqué dans un élément X. Sa syntaxe est la suivante :

  selecteur_X selecteur_Y { /* style; */ }

  Soit le code HTML suivant :

    Attention , ceci est un avertissement

   Prière d'en tenir compte

  La règle suivante ne sélectionne que le mot « avertissement » (le seul entouré de balises <B>, elles-mêmes imbriquées dans une balise <P>) :

  Exemple: P B { font-weight: bold; color: red; }

- Le **sélecteur d'éléments consécutifs** permet de créer une règle ne s'appliquant que lorsqu'un élément Y suit immédiatement un élément X. Sa syntaxe est la suivante :

  selecteur_X + selecteur_Y { /* style; */ }

  Soit le code HTML suivant :

    Attention , ceci est un avertissement

   Prière d'en tenir compte

  La règle suivante ne sélectionne que le mot « avertissement » (le seul entouré de balises <B>, elles-mêmes suivant une balise <I>) :

  Exemple: I + B { font-weight: bold; color: red; }

- Le **sélecteur d'éléments père-fils** permet de créer une règle ne s'appliquant que lorsqu'un élément Y est fils direct d'un élément X. La règle ne s'applique pas si Y est encapsulé dans une ou plusieurs autres balises intermédiaires. Sa syntaxe est la suivante :

  selecteur_X > selecteur_Y { /* style; */ }

  Soit le code HTML suivant :

    Attention , ceci est un avertissement

   Prière d'en tenir compte

  La règle suivante ne sélectionne que l'élément « * Attention * » (le seul entouré de balises <B>, elles-mêmes directement imbriquées dans une balise <P>) :

  Exemple: P > B {font-weight: bold; color: red; }

2) Implantation du code

a) Déclaration du type de document

Il est nécessaire d'indiquer dans la page HTML le prologue du type de document, c'est-à-dire une référence à la norme HTML utilisée. Cette déclaration se fait par une ligne du type :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
    <HEAD>...</HEAD>
    <BODY>Contenu de la page</BODY>
</HTML>
```

De plus, une balise Meta permet d'indiquer au navigateur ou aux moteurs de recherche le langage utilisé pour la définition des feuilles de style. Cette balise Méta, à inclure dans l'en-tête HTML du document est la suivante :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
    <HEAD>
        <META HTTP-EQUIV="Content-Style-Type" CONTENT="text/css">
    </HEAD>
    <BODY>Contenu de la page</BODY>
</HTML>
```

b) Style interne

Les feuilles de style d'une page web sont déclarées grâce à la balise STYLE, au sein des balise <HEAD> et </HEAD>.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
    <HEAD>
        <STYLE type="text/css">
        <!--

        Définition des styles;

        -->
        </STYLE>
    </HEAD>
<BODY></BODY>
</HTML>
```

L'attribut type="text/css" de la balise *<STYLE>* permet de spécifier le type de feuille de style utilisée. La balise de commentaire *<!*-- ... --> sert à éviter que des navigateurs peu récents, donc ne supportant pas les feuilles de style, affichent ces informations.

c) Style en ligne

Il est également possible de définir le style au sein même d'une balise d'un document. On appelle ce type de déclaration une déclaration en ligne.

Cette façon de définir les feuilles de style est peu recommandée car elle va à l'encontre de l'intérêt des feuilles de style, dans la mesure où le style est embarqué au sein même de chaque élément. Cela peut néanmoins servir pour définir de façon exceptionnelle un style pour un élément HTML particulier, ne nécessitant pas une définition globale.

Pour définir un style en ligne, il suffit de renseigner l'attribut STYLE de la balise HTML à laquelle on souhaite appliquer un style particulier :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
    <HEAD>
    ...
    </HEAD>
<BODY>
...
<BALISE Style="style:valeur;"> ... </BALISE>
...
</BODY>
</HTML>
```

N.B : Il est possible d'appliquer un style "en ligne" à toutes les balises HTML, hormis les balises suivantes : BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, STYLE, TITLE

Voici un exemple de style appliqué à une balise *<H1>* :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
    <HEAD>
    ...
    </HEAD>
<BODY>
...
<H1 Style="Font: 18px Verdana; font-weight:bold;"> Titre </H1>
```

```
...
</BODY>
</HTML>
```

d) Style externe

Le fait de pouvoir stocker la définition des feuilles de style à l'extérieur du document est un "plus" car il est ainsi possible, en modifiant le fichier contenant les feuilles de style, de changer l'allure de toutes les pages web s'y référant !

Il s'agit dans un premier temps de créer un fichier texte contenant les feuilles de style et dont l'extension est **.css**, par exemple *style.css* :

```
<!--
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
        body   {background-image: home.gif;}
        LI              {font: 13px Verdana;}
        B               {font: 14px Verdana; font-weight: bold;}
        A               {
                         font:12px Verdana;
                         font-weight: bold;
                         color=black;
                         text-decoration: none;
                        }
        H1              {font: 16px Arial;font-weight: bold;color=black;}
        H2              {font: 14px Arial;font-weight: bold;color=black;}
-->
```

Dans un second temps il suffit de créer dans chaque page HTML le raccourci vers cette page de définition de style :

```
<HTML>

        <HEAD>

                <LINK rel="stylesheet" type="text/css" href="style.css">

        </HEAD>

...
```

La balise <*LINK*> avertit le navigateur qu'il doit chercher un document situé à l'extérieur de la page HTML.

- L'attribut *rel="stylesheet"* précise que le document en question est une feuille de style externe.

- L'attribut *type="text/css"* précise le type de feuille de style.
- L'attribut *href=" URL "* donne l'URL de la feuille de style, c'est-à-dire son emplacement sur Internet.

e) Style importé

Les recommandations du W3C offrent une dernière façon d'inclure des feuilles de style dans un document: en important des feuilles de style. Il est en effet possible d'importer des feuilles de style externes au niveau de la déclaration du style de document, en insérant la commande @IMPORT immédiatement après la balise style :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
    <HEAD>
        <STYLE type="text/css">
        <!--
        @IMPORT URL(url de la feuille à importer);

        Définition des styles du document;

        -->
        </STYLE>
    </HEAD>
<BODY></BODY>
</HTML>
```

N.B : Si plusieurs définitions importées concernent la même balise, seule la dernière sera prise en compte par le navigateur.

f) Styles en cascade

Il est possible de définir plusieurs styles en utilisant les différents moyens qu'offrent les CSS. Ainsi, lorsque plusieurs feuilles de style externes sont appelées, on obtient ce que l'on appelle une cascade de styles, c'est-à-dire une combinaison de styles pour divers éléments HTML. Si plusieurs styles concernent le même élément, seul le dernier style sera conservé.

```
<LINK rel=stylesheet type="text/css" href="style1.css">

<LINK rel=stylesheet type="text/css" href="style2.css">

<LINK rel=stylesheet type="text/css" href="style3.css">
```

Dans le cas où plusieurs styles sont redondants entre différentes feuilles de style externes, les recommandations CSS permettent également d'offrir le choix entre plusieurs feuilles de styles alternatives grâce à l'attribut rel de la balise STYLE, combiné à un attribut TITLE permettant de les choisir nominativement :

<LINK rel=" alternate stylesheet" type="text/css" href="style1.css" title="style1">

<LINK rel="alternate stylesheet" type="text/css" href="style2.css" title="style2">

<LINK rel="stylesheet" type="text/css" href="stylepardefaut.css">

D'autre part, lorsque plusieurs styles sont appelés dans une page en utilisant les différents moyens d'inclusion possibles, la prise en compte des styles, lorsque plusieurs styles sont redondants, est telle que le style le plus proche du contenu est maintenu. Ainsi, l'ordre de priorité est le suivant :

Style en ligne > Style du document > Style importé > Style externe

3) Unités de mesure et positionnement des CSS

Grâce aux feuilles de style il est possible de définir des valeurs numériques pour les propriétés de style de plusieurs façons :

- de façon absolue, c'est-à-dire dans une unité indépendante du format de sortie (en centimètres par exemple) ;
- de façon relative, c'est-à-dire dans une unité relative à un élément ;

Les valeurs des feuilles de style sont soit des nombres entiers, soit des nombres réels, c'est-à-dire des chiffres ayant une partie entière et une partie décimale.

D'une manière générale il est à noter l'utilisation du point (« . ») dans les notations décimales en lieu et place de la virgule (« 8.5 cm« et non « 8,5 cm«).

Les valeurs peuvent par ailleurs dans certains cas être négatives (précédées du signe « - »). Certaines propriétés peuvent néanmoins accepter un intervalle restreint de valeurs.

a) Unités absolues

Les unités absolues proposées par le standard CSS sont récapitulées dans le tableau suivant :

unité	description
cm	Le centimètre
in	Le pouce (en anglais « inch ») correspondant à 2,54 cm
mm	Le millimètre
pt	Le point

pc	Le pica (correspondant à 12 pt)

b) Unités relatives

Les unités relatives proposées par le standard CSS sont récapitulées dans le tableau suivant :

unité	description
em	Unité relative à la taille de police de l'élément sélectionné. Seule exception à cette règle : lorsque la propriété *font-size* est définie, elle se rapporte à la taille de la police de l'élément parent.
ex	Unité relative à la hauteur de la minuscule de l'élément sélectionné. Seule exception à cette règle : lorsque la propriété *font-size* est définie, elle se rapporte à la hauteur de la minuscule de l'élément parent.
px	Le pixel. Il s'agit d'une unité dont le rendu dépend de la résolution du périphérique d'affichage.
%	Le pourcentage est une unité relative à la taille de l'élément ou de son parent.

c) Positionnement relatif et absolu

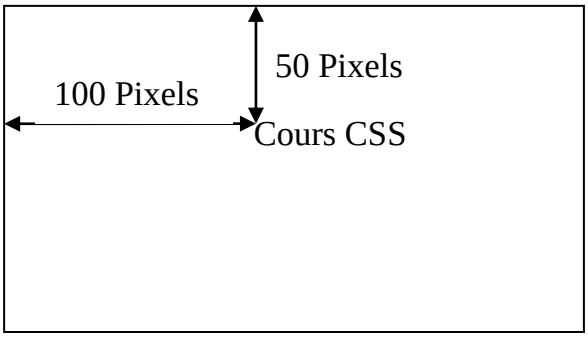
Le positionnement absolu {position: absolute} se détermine par rapport au coin supérieur gauche de la fenêtre du navigateur. Les coordonnées d'un point s'expriment alors de haut en bas (top) et de gauche à droite (left).

La position relative se fait par rapport à d'autres éléments, comme une image, c'est-à-dire que les éléments contenus dans la balises *DIV* ou *SPAN* seront positionnés à la suite des éléments HTML après lesquels ils se trouvent.

d) Positionnement d'un texte

Positionnons le texte "Cours CSS" à 50 pixels du haut de la fenêtre et à 100 pixels à gauche de la fenêtre :

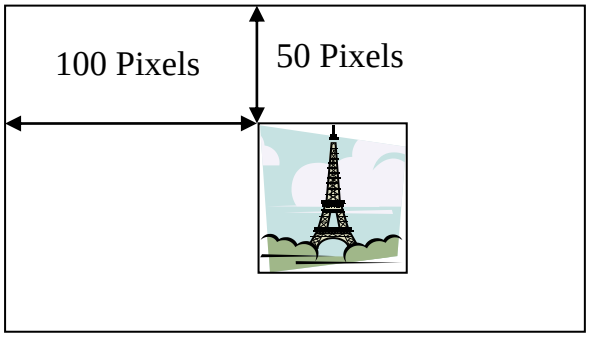
```
<HTML>
<BODY>
<SPAN style="position: absolute; top: 80 px; left: 100 px;">
Cours CSS
</SPAN>
</BODY>
</HTML>
```



e) Positionnement d'un texte

Positionnons l'image "eiffel.jpg" à 50 pixels du haut de la fenêtre et à 100 pixels à gauche de la fenêtre:

```
<HTML>
<BODY>
<SPAN style="position: absolute; top: 50 px; left: 100 px;width: 103px; height: 61px">
<IMG SRC="eiffel.jpg" >
</SPAN>
</BODY>
</HTML>
```



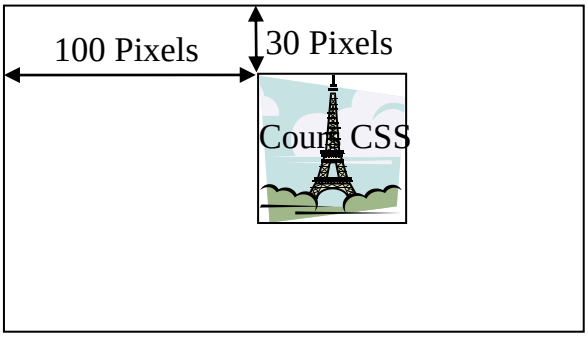
Il est important de spécifier la taille de l'image avec les feuilles de style, pour des raisons de non-compatibilité des navigateurs.

f) Superposition des éléments

Superposons le texte "Cours CSS" à l'image "eiffel.jpg":

```
<HTML>
<BODY>
<SPAN style="position: absolute; top: 30 px; left: 100px;width: 103px; height: 61px">
<IMG SRC="eiffel.jpg" >
</SPAN>
<SPAN style="position: absolute; top: 50 px; left: 100 px;">
Cours CSS
</SPAN>
```

</BODY>
</HTML>



4) Classes et ID

Il peut s'avérer intéressant d'affecter des styles différents à des mêmes balises. Pour cela les spécifications CSS ont introduit le concept de **classe**.

La définition des classes est aussi simple que celles des styles. Elle consiste à préciser la balise sélectionnée (comme pour une déclaration de style), puis de lui ajouter un point (.) et le nom que l'on souhaite donner à la classe. Le nom de la classe peut-être composé de lettres (accentuées ou non), de chiffres et de tirets :

Selecteur_de_balise.Nom-de-la-classe {

propriété de style: Valeur;

propriété de style: Valeur;

...;

}

Où « Nom-de-la-classe » représente le nom donné à la classe.

Pour appeler une classe dans le code HTML, il suffit de rajouter un attribut *class* à la balise :
Soit la classe *Rouge* appliquée à la balise *b* :

B.rouge {font: Verdana 12px; color: #FF0000; }

L'appel à cette classe dans le code se fera de la façon suivante :

<B class="Rouge"> Texte à mettre en rouge et en gras </B>

Les pseudo-classes permettent d'affiner le style appliqué à un certain nombre de balises en définissant une réaction à un événement ou bien à la position relative de la balise au sein des autres balises.

Contrairement aux classes, le nom des pseudo-classes est prédéfini, il n'est donc pas possible de créer ses propres pseudo-classes. Il existe plusieurs types de pseudo-classes :

- Les pseudo-classes dynamiques,
- Les pseudo-classes de lien,
- Les pseudo-classes de langue,
- Les pseudo-classes first-child,
- Les pseudo-classes de page,
- Les pseudo-éléments.

Le sélecteur d'ID (identifiant) permet de faire référence à un élément unique d'une page repéré par son identifiant. Les ID servent notamment à localiser des éléments HTML grâce au JavaScript.

La syntaxe d'un sélecteur d'ID est la suivante :

#nom_ID { style }

Un tel style s'appelle de la manière suivante :

<BALISE ID="nom_ID" > ... </BALISE>

N.B : Il ne peut exister qu'un seul ID par page ! Notez également l'absence de # dans l'appel au sélecteur d'ID.

a) Les classes universelles

Il est possible de ne pas préciser de balise, auquel cas la classe pourra être utilisée dans n'importe quelle balise pour laquelle le style sélectionné a un sens ! On parle alors de **classe universelle** (parfois *classe indépendante*). La définition d'une telle classe se fait en précédant tout simplement le nom de la classe d'un point :

.Nom-de-la-classe {propriété de style: Valeur; propriété de style: Valeur ...}

Soit la classe «Cours » suivante :

.Cours {font-type: arial; color: red; font-weight: bold}

L'appel de cette classe peut être fait à partir de n'importe quel élément HTML pour lequel la définition est valide :

<h1 class="Cours">Cours CSS</h1>

Ceci est un cours

N.B : Il n'y a pas de point lors de l'appel de la classe.

b) Les pseudo-classes dynamiques

Les pseudo-classes dynamiques permettent de modifier le style d'une balise en fonction d'un événement (mouvement de la souris, clic, ou bien appui sur une touche du clavier). Il en existe trois :

- La pseudo-classe *:hover* permet d'affecter un style à la balise sélectionnée lors d'un survol par le curseur de la souris :

  Exemple : A:hover {font-decoration: underline;}

- La pseudo-classe :focus permet de définir un style à la balise sélectionnée lorsque le focus lui est donné (par exemple lors d'un clic dans un élément de formulaire) :

  Exemple : TEXTAREA:focus {color: #FF0000;}

- La pseudo-classe :active permet de définir un style à la balise sélectionnée lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche) :

  Exemple : A:active {color: #FF0000;}

N.B : Les pseudo-classes dynamiques ne sont pas reconnues de la même façon par tous les navigateurs.

c) Les pseudo-classes de lien

Les pseudos-classes de lien sont des pseudo-classes spécifiques à la balise <*A*> :

- La pseudo-classe *:link* permet de définir le style des liens hypertextes n'ayant pas encore été consultés par le client
- La pseudo-classe *:visited* permet de définir le style des liens hypertextes que le client a déjà visités

N.B : Il est possible que certains navigateurs considèrent un lien comme n'ayant pas été visité s'il n'est pas consulté pendant une longue période de temps.

d) La pseudo-classe descendante

Une pseudo-classe *descendante* permet d'appliquer un style à la première balise au sein d'un élément. La syntaxe de cette pseudo-classe est la suivante :

Element_Parent > Balise:first-child {style;}

Ainsi la déclaration suivante s'applique à la première balise <A> situé dans un jeu de balises <P> </P> :

```
P > A:first-child {color: #00FF00;}
```

De cette manière, la balise <A> suivante possèdera le style ci-dessus :

```
<P align="center">
  <A href="http://www.Google.fr">GOOGLE</A>
  </P>
```

La balise <A> suivante ne sera par contre pas prise en compte car elle n'est pas la première balise après la balise <P> :

```
<P align="center">
  <BR/>
  <A href="http://www.Google.fr">GOOGLE</A>
  </P>
```

e) Les pseudo-classes de texte

Les pseudo-classes de texte permettent d'appliquer un style à une partie du texte délimité par les balises auxquelles ils s'appliquent. Ainsi les pseudo-classes de texte s'utilisent généralement conjointement avec la balise de paragraphe (*<P>*).

Il existe deux pseudo-classes de texte :

- *:first-line* : permet d'appliquer un style à la première ligne d'un paragraphe.

  Exemple : P:first-line { text-transform: uppercase }

- *:first-letter* : permet d'appliquer un style à la première lettre d'un paragraphe afin de produire un effet typographique. L'exemple suivant par exemple double la taille et met en gras la première lettre d'un paragraphe :

  Exemple : P:first-letter {font-size: 200%; font-weight: bold; }

f) Les pseudo-classes de langue

Il est possible de définir un style en fonction de la langue du document (spécifiée dans les en-têtes HTTP ou dans les balises méta) ou de la langue d'un élément HTML ou XML (spécifié grâce à l'attribut optionnel *LANG*) si celle-ci est précisée.

Une pseudo classe de langue utilise la notation suivante :

:lang(Langue).

Exemple : La pseudo classe de langue suivante permet de définir les guillemets selon la notation française :

HTML:lang(fr) { quotes: '« ' ' »' }

g) Les pseudo-classes de page

Le sélecteur @page permet de modifier les définitions de mise en page d'une page HTML (taille, marge, etc.) à l'impression, telles que les marges (margin-left, margin-top, margin-right, margin-bottom), la taille (*size*). Il faut alors imaginer la page web comme un ensemble de pages constituant un ouvrage papier.

Les pseudo-classes de page permettent ainsi de sélectionner les pages de gauche, de droite ou bien la première page d'un document.

Il existe différentes pseudo-classes de page :

- *@page:left* : permet de définir les propriétés des pages de gauche.

  Exemple : @page:left { size: landscape; margin-left: 2cm; }

- *@page:right* : permet de définir les propriétés des pages de droite.

  Exemple : @page:right { size:landscape; margin-left: 2.5cm; }

- *@page:first* : permet de définir les propriétés de la première page d'un document.

  Exemple :@page:first { size: portrait;

  margin-left: 2.5cm;
  margin-right: 2cm;
  margin-bottom: 1cm;
  margin-top: 4cm;}

5) Les propriétés CSS

a) Propriétés de police

Propriété	Valeur	Description
font-family	Police précise (Arial, Times, Verdana) Famille (serif, sans-serif, fantasy, monospace, cursive)	Définit un ou plusieurs noms de polices ou de familles de polices. Si plusieurs polices sont définies, la première trouvée sur le système de l'utilisateur sera utilisée.
font-style	normal, italic, oblique	Définit le style d'écriture
font-weight	lighter, normal, bold ou	Définit la graisse (épaisseur) de la police

	bolder. valeur numérique (100, 200, 300, 400, 500, 600, 700, 800, 900)	
font-size	xx-small, x-small, small, medium, large, x-large, xx-large taille en points (pt), cm, %	Définit la taille de la police
font-variant	normal, small-caps	Définit une variante (petites majuscules)
font	font: Verdana, Arial, bold italic 8px;	Raccourci permettant de mettre toutes les propriétés

b) Propriétés de Textes et Paragraphes

Propriété	Valeur	Description
color	"#RRGGBB"	Définit la couleur du texte
line-height	line-height: 12pt;	Définit l'interligne
text-align	left, center, right ou justify	Définit l'alignement du texte
text-indent	text-indent: 5px;	Définit l'indentation (retrait du texte)
text-decoration	*blink* (clignotement), *underline* (souligné), *line-through* (barré), *overline* (surligné) ou *none* (aucune décoration)	Définit une décoration
text-shadow	text-shadow: 1px 2px 4px black;	Définit l'ombrage texte, respectivement décalage à droite, en bas, rayon de l'effet de flou et couleur.
text-transform	*uppercase* (majuscule), *lowercase* (minuscule) ou *capitalize* (première lettre en majuscule)	Définit la casse du texte
white-space	normal (passage à la ligne automatique), pre (idem saisie), nowrap (pas de passage à la ligne automatique)	Césure
word-spacing	word-spacing: 6px;	Définit l'espacement des mots
width	En points (pt), pouces (in), en cm, en pixels (px), ou en %	Définit la longueur d'un élément de texte ou d'une image
height	En points (pt), pouces (in), en cm, en pixels (px), ou en %	Définit la hauteur d'un élément de texte ou d'une image

c) Propriétés de couleurs et arrières plans

Propriété	Valeur	Description
background-color	"#RRGGBB"	Définit la couleur d'arrière plan
background-image	url(http://url)	Définit l'image d'arrière-plan

background-repeat	repeat, repeat-x, repeat-y, no-repeat	Définit la façon de répéter l'arrière-plan
background-attachment	scroll, fixed	Spécifie si l'image reste fixe avec les déplacements de l'écran
background-position	top, middle, bottom, left, center ou right	Position de l'image par rapport au coin supérieur gauche
background	background: url(test.jpg) fixed repeat;	Raccourci pour les propriétés d'arrière-plan

d) Propriétés de marges

Propriété	Exemple	Description
margin-top	margin-top: 5px;	Valeur de la marge supérieure
margin-right	margin-right: 0.5em;	Valeur de la marge de droite
margin-bottom	margin-bottom: 2pt;	Valeur de la marge inférieure
margin-left	margin-left: 0;	Valeur de la marge de gauche
margin	margin: 5px 0.5em 2pt 0;	Raccourci pour les propriétés de marge

e) Propriétés de bordures

Propriété	Valeur	Description
border[-top -left -bottom -right]-width	En points (pt), pouces (in), en cm, en pixels (px), ou en %	Epaisseur de la bordure [supérieure, de gauche, inférieure ou de droite]
border[-top -left -bottom -right]-color	border-left-color: #RRGGBB;	Couleur de la bordure [supérieure, de gauche, inférieure ou de droite]
border[-top -left -bottom -right]-style	*solid* (pleine), *dashed* (en tirets), *dotted* (en pointillés), *double* (double et remplie) ou *ridge* (en 3D)	Style de la bordure [supérieure, de gauche, inférieure ou de droite]
border-collapse	collapse separate	Effet » 3D » ou non
border	border: 1px 0 0 2px dotted green;	Raccourci global les propriétés de bordure

f) Propriétés des espaces intérieurs

Propriété	Valeur	Description
padding-top	padding-top: 3px;	Espace intérieur entre l'élément et la bordure supérieure
padding-right	padding-right: 0.25em;	Espace intérieur entre l'élément et la bordure droite
padding-bottom	padding-bottom: 0;	Espace intérieur entre l'élément et la bordure inférieure

padding-left	padding-left: 2pt;	Espace intérieur entre l'élément et la bordure gauche
padding	padding: 3px 0.25em 0 2pt;	Raccourci vers l'ensemble des propriétés d'espace intérieur

g) Propriétés des tableaux

Propriété	Valeur	Description
border-collapse	*separate* ou *collapse*	Fusion des bordures des cellules (*collapse*) ou non (*separate*)
border-spacing	border-spacing: 4px;	Espacement des cellules
caption-side	top, bottom, left ou right	Positionnement de la légende du tableau
empty-cells	*show* ou *collapse*	Affichae (*show*) ou masquage (*collapse*) des cellules vides
table-layout	*fixed* (indépendamment du contenu des cellules) ou *auto* (selon le contenu des cellules)	Largeur fixe ou variable
speak-headers	*alwats* (systématiquement avant chaque cellule) ou *once* (une seule fois)	Propriété pour sourds et malentendants indiquant le comportement lors de la lecture des cellules d'en-tête d'un tableau

h) Propriétés des listes

Propriété	Valeur	Description
list-style-type	decimal, upper-roman, lower-latin, disc, circle, square ou none	Type de puces et de numérotation
list-style-image	list-style-image: url(image.png);	Permet de personnaliser les puces avec une image
list-style-position	inside ou outside	Spécifie le retrait des puces
list-style		Raccourci vers les propriétés de liste

i) Propriétés de mise en page

Propriété	Valeur	Description
@page	@page(size: portrait)	Définit la mise en page de l'impression
size	auto, landscape ou portrait	Format de l'impression
margin-top	margin-top: 3 cm;	Marge supérieure
margin-right	margin-right: 1.5 cm;	Marge de droite
margin-	margin-right: 1 cm;	Marge inférieure

bottom		
margin-left	margin-left: 2 cm;	Marge de gauche
marks	crop (traits de coupe), cross (repère de montage), none (pas de marque)	Traits de coupe et repères de montage
page-break-before	Always, avoid	Force le saut de page avant un élément
page-break-after	Always, avoid	Force le saut de page après un élément
orphans	orphans: 2;	Evite les lignes orphelines en fin de page. Définit le nombre de ligne minimal à partir un renvoi en page suivante est effectué
widows	widows: 1;	Evite les lignes veuves en début de page. Définit le nombre de ligne minimal à partir un renvoi en page précédente est effectué

Bibliographie

(s.d.). Récupéré sur CSS débutant: http://www.cssdebutant.com

Lapoire, D. (2006, Octobre 12). Initiation à l'Algorithmique.

Pillou, J.-F. (2007). Récupéré sur Comment ça Marche: http://www.commentcamarche.net

Thomas Cormen, C. L. (2004). *Introduction à l'Algorithmique.* Paris: Dunod.