Algorithme et programmation

Mathématiques 2nde
Module No 25

Les structures de base d'un programme

Programme

- **CM#5 Les structures de base d'un programme**
- Séquences logiques et structures logiques
- Opérations élémentaires : affichage, saisie, affectation, restitution
- Opérateurs arithmétiques
- Préalable pour les TD : choix d'un environnement de développement
  - Le couple HTML/JavaScript. Avantages du choix.
  - Un langage de balisage : HTML
  - Un langage de programmation : JavaScript
- Structure alternative No 1 (Si ... alors ... sinon ...)
- Structure alternative No 2 (Selon ... cas ...)
- Structure répétitives No 1 (Tant que ... répète ...)
- Structure répétitive No 2 (Répète ... tant que ...)
- Structure répétitive No 3 (Pour ... suivant ...)

Quelques questions

Programme

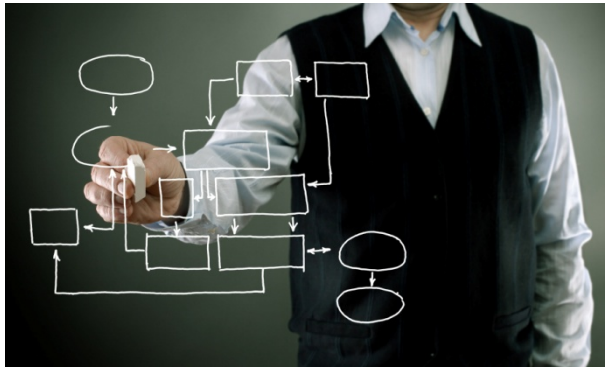
- **TD Séance #2 : Manipuler les structures de base**
- Exercice #1 : Mon premier programme HTML
- Exercice #1bis : Mon premier programme HTML+Javascript
- Exercice #2 : Somme des n premiers nombres (Implantation concrète du programme vu en classe)
- Exercice #3 : Afficher le reste et le quotient d'une division dont vous saisissez les opérandes
- Exercice #4 : Afficher le plus grand des 3 nombres saisis (application structure alternative No 1)
- Exercice #5, 6 et 7 : Somme des n premiers nombres (Versions 2, 2bis et 2 ter de la somme des n premiers nombres : application des structures répétitives)

Au commencement était l'artisanat

- La programmation comme un art : chacun fait selon sa propre vision

- Organigrammes « spaghetti » : les sauts multiples facilités par les instructions de branchement (Go To) font qu'il est impossible de se placer en un point quelconque du programme et de pouvoir dire quelles sont les conditions qui permettent de s 'y amener.
- « Ecrivons toujours, nous verrons bien si cela tourne ! »
- Difficulté pour chaque individu de maintenir un programme écrit par un tiers.
- Complexité entretenue => moindre fiabilité, développement et entretien malaisés

Programmation structurée



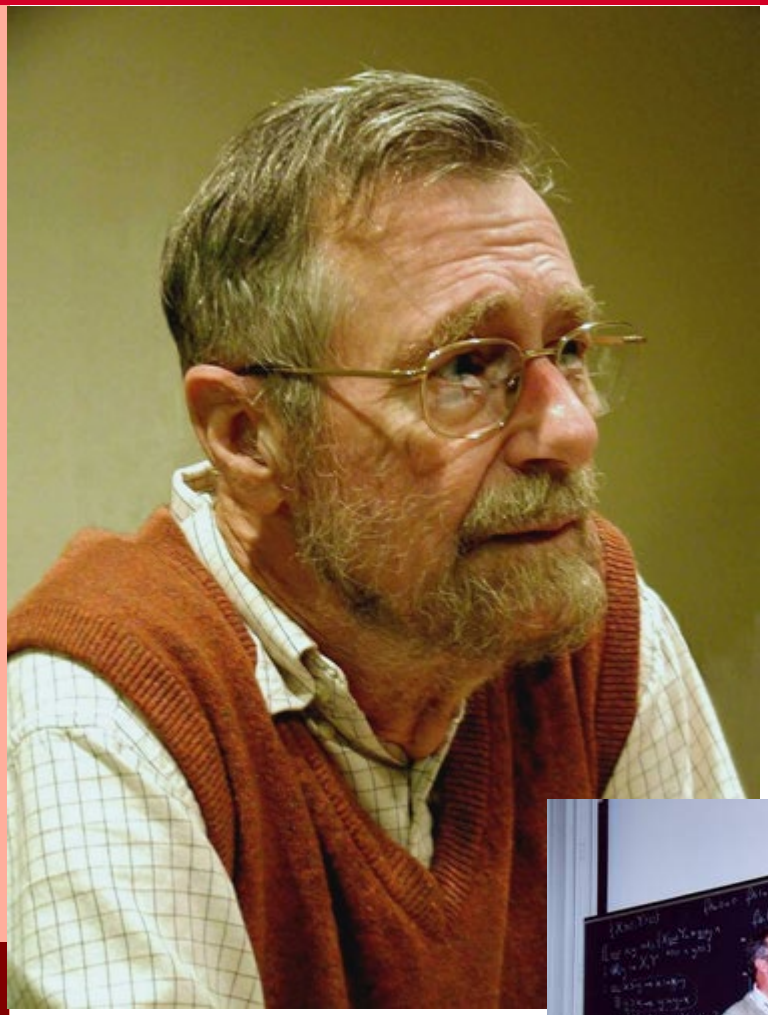
- Premières réflexions en 1958
- Théorème de Bohm et Jacopini : « Organigrammes de machine de Türing et langages utilisant seulement deux lois de construction de programmes », paru dans « *Communications of the ACM* » en Mai 1966
- Travaux de Edsger W Dijkstra (1968) : « *Go to statements considered harmful* » et « *Notes on structured programming* »
- De nouveau Bohm et Jacopini : Théorème démontrant que tout programme peut être écrit en utilisant uniquement les trois structures de base.
- Principe de Mills (IBM 1970) : Tout module doit avoir une seule entrée et une seule sortie.
- Spécification du langage Pascal en 1972 par N. Wirth.
- Travaux de J.D. Warnier en France (LCP-LCS).

“Go To Statement Considered Harmful ”

« Tester un programme peut démontrer la présence de bugs, jamais leur absence. »

« Les progrès ne seront possibles que si nous pouvons réfléchir sur les programmes sans les imaginer comme des morceaux de code exécutable. »

« La programmation par objets est une idée exceptionnellement mauvaise qui ne pouvait naître qu'en Californie. »



Edsger Dijkstra

Mathématicien et informaticien néerlandais

Un projet phare : Automatisation SI NY Times

- Premier grand programme mettant en application les principes de la Programmation Structurée (Chef de projet : F.T. Baker)
  - 83 000 lignes de programme
  - 11 années.hommes sur 22 mois
  - Productivité *5 vis à vis des standards de l'époque (de 5 à 8 lignes Cobol mises au point et documentées par jour et par programmeur, on passe à 30)
  - 95 % des programmes « tournent » dès le premier essai
  - Après validation, détection ultérieure de seulement 2 erreurs en vingt mois de fonctionnement

Règles de la programmation structurée

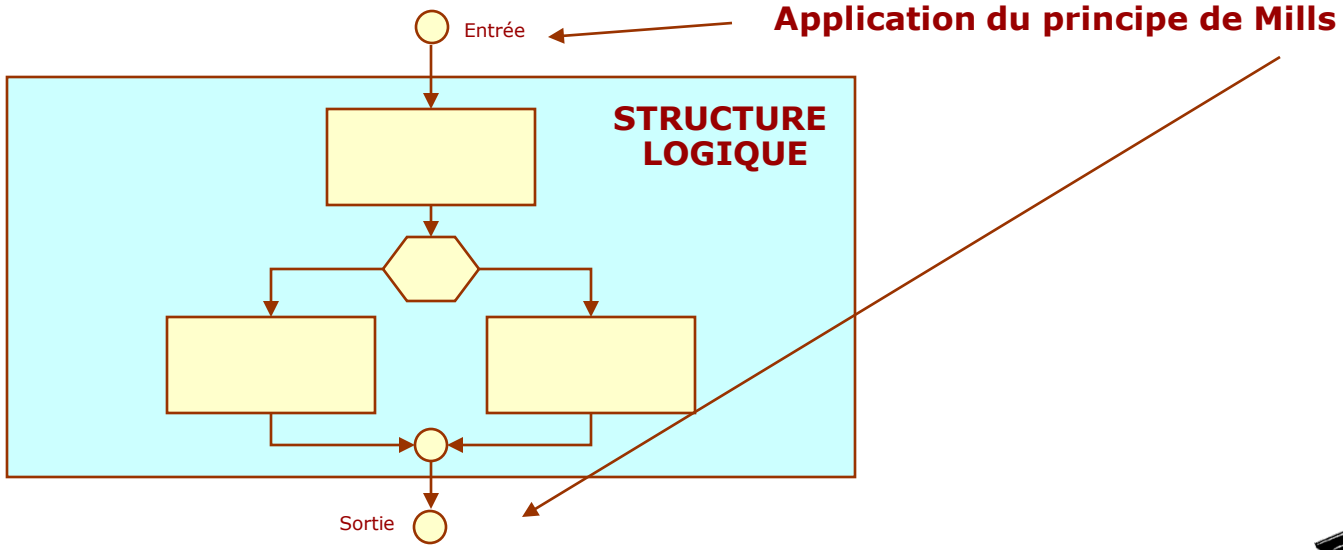
- Construction de programmes sans Go To et utilisation des trois structures de base de Dijkstra :
  - Séquence,
  - Structure alternative,
  - Structure répétitive.
- Utilisation des règles strictes d'une démarche « *Top Down* ».
- Concept de ressources abstraites, définies à un niveau donné, et utilisables aux niveaux supérieurs.

Structures de base

- **Séquence logique** : séquence d'opérations élémentaires

SEQUENCE LOGIQUE
Opération 1
Opération 2
.....
Opération n

- **Structure logique** : composition structurée de séquences logiques et d'opérateurs logiques



Eléments d'un langage

- Un **programme impératif** décrit les opérations en termes de séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme.
- Ce type de programme est le plus répandu et se différencie des productions de la programmation déclarative -ou logique- et de la programmation dite fonctionnelle.
- Un programme impératif est généralement constitué d'un ensemble de déclarations et d'un ensemble d'instructions.
- Les **déclarations** permettent d'annoncer et d'initialiser des constantes, types, variables, fonctions et procédures.

Eléments d'un langage

- Les **mots-clés** sont des mots particuliers qui ne peuvent pas être utilisés comme identificateurs.
- Par exemple, pour le langage Pascal : *program const var type function procedure begin end array record set of file if then else while do repeat until case for to downto and or not in div mod goto label nil packed with*
- Les **constantes** sont de mots qui désignent des valeurs d'un type prédéfini du langage. Typiquement, on trouve des constantes entières, réelles, booléennes (par exemple *true* et *false*), caractères et éventuellement chaînes.

Eléments d'un langage

- Pendant le déroulement d'un programme, on va avoir en permanence besoin de créer, faire évoluer et stocker provisoirement des valeurs.
- Il peut s'agir de données issues d'un fichier ou d'une base ou fournies par l'utilisateur (frappées au clavier.
- Il peut aussi s'agir de résultats obtenus par les calculs déroulés par le programme, intermédiaires ou définitifs.
- Ces données peuvent être de plusieurs types (nombres, texte).
- Ces données sont des **variables**.
- Pour accéder à une variable dans un programme, il suffit de mentionner son identificateur (étiquette).

Opérations élémentaires

Affichage de données

- Afficher l'intitulé « No commande »

Lecture de données

- Saisir le No de la commande
- Lire (dans la base de données) le prix unitaire

Affectation

- En *Etiq1* enregistrer -5 (*Etiq1* ← -5)
- En *Commande_No* enregistrer la valeur saisie (*Commande_No* ← valeur saisie)
- Attention : dans de nombreux langages confusion possible entre affectation et égalité (← et =)

Restitution = Affichage

- Afficher *Produit_prix*

Opérations élémentaires

- **Opérateurs arithmétiques**
  - *Produit_PrixBrut = Produit_PrixUnitaire* ***** *Commande_QteCommandee*
  - *Commande_PrixTTC = Commande_PrixHT* **+** *Commande_TVA*
  - *PartMarche = CAGlobal* **/** *MarcheGlobal*
  - *Produit_PrixNet= Produit_PrixBrut* **-** *Client_Remise*
- **Opérateurs logiques**
  - *((i >= 0)* ***ET*** *(i <= 9)* (Chaque langage affectera un symbole particulier à l'opérateur logique ET)
  - Opérateurs ***ET***, ***OU***, ***NON***, OU exclusif
- **Opérateurs relationnels**
  - **>** *strictement supérieur*
  - **>=** *supérieur ou égal*
  - **<** *strictement inférieur*
  - **<=** *inférieur ou égal*
  - **==** ***ou*** **=** *égal*
  - **!=** *différent*

Ne pas confondre

- **L'entité**
  - *No de la commande*
- **L'intitulé**
  - « No Commande »
- **L'étiquette symbolique**
  - *Commande_No*
- **L'opération élémentaire**
  - *Commande_No = 1234;*
- Les opérations élémentaires et les composants de base des structures logiques (opérateurs de comparaison, opérateurs logiques, commandes de base de type IF (SI), REPEAT UNTIL (REPETER JUSQU'À CE QUE), DO WHILE (FAIRE TANT QUE)) correspondent à des instructions du langage choisi.

L'écriture en pseudo-code d'une séquence logique

- Rappel : Le pseudo-code est un langage simplifié, proche du langage naturel, qui décrire un algorithme sans référence à un langage de programmation particulier.
- Chaque séquence est un bloc balisé par le couple $: :$
- Exemple du calcul de la somme des N premiers nombres entiers

```
$:      /* Calcul de la somme des N premiers
nombres entiers */
$1:  Afficher « Saisissez N »
N ← valeur saisie
SOMME ← N * (N+1) / 2
Afficher SOMME
:1$
:$
```

Pseudo-code

Choix de langages pour les exercices

- Ni pour apprendre un langage de programmation
- Ni pour mettre au point des programmes
- Mais
- Pour visualiser l'organisation et le déroulement des séquences et structures,
- Pour vérifier la logique de raisonnement,
- Pour mettre au point des processus de raisonnement.

Choix de HTML et Javascript

- Outils en standard dans toute configuration
  - Un éditeur de texte
  - Un navigateur Internet
- Un interpréteur et non un compilateur : identification immédiate de la ligne erronée
- Automatismes simplifiant l'écriture compte tenu de notre objectif particulier : exemple de l'affectation automatique d'un type (entier, réel, chaîne) en fonction de la valeur affectée.
- Attention : ces automatismes créent quelques pièges et peuvent s'avérer nuisibles si votre objectif est la réalisation d'un programme professionnel
- Attention : Javascript différencie minuscules et majuscules (*case sensitive*),
- Attention = pour affectation, == pour égalité.

Mon premier programme HTML

```
<HTML>
<HEAD>
    <title>Programmatique Exo 1</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B>Bienvenue sur ma page d'accueil</B><BR>
 </FONT>
</p>
</BODY>
</HTML>
```

Code HTML

Mon premier programme HTML/ Javascript

```
<HTML>
<HEAD>
    <title>Programmatique Exo 1bis</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B>Premier exemple de programme</B><BR>
<SCRIPT LANGUAGE="JavaScript">
/* Un premier exercice */
var texte="Bienvenue sur ma page d'accueil"
document.write(texte);
document.write("<BR>");
</SCRIPT>
 </FONT>
</p>
</BODY>
</HTML>
```

Code HTML

Code Javascript

Exo #2 : somme des n premiers nombres

```
<HTML>
<HEAD>
    <title>Programmatique Exo 2</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Calcul de la somme des N premiers nombres entiers
</B><BR>
<SCRIPT LANGUAGE="JavaScript">
N = prompt("Saisissez une valeur du nombre N ",0);
N=N*1 //pour transformer la chaine saisie en valeur numerique
SOMME = (N*(N+1))/2
document.write("Somme des " + N + " premiers nombres = " + SOMME);
document.write("<BR>");
</SCRIPT>
 </FONT>
</p>
</BODY>
</HTML>
```

```
$:    /* Calcul de la somme
des N premiers nombres entiers */
$1:
Afficher titre
Affiche « Saisissez une valeur N »
N ← valeur saisie
SOMME ← N * (N+1) / 2
AFFICHE SOMME
:1$
:$
```

Dans le cadre jaune, le pseudo-code

Exo #3

- Afficher le reste et le quotient d'une division dont vous saisissez les opérandes

Exo #3 : Ecriture du pseudo-code

- Afficher le reste et le quotient d'une division dont vous saisissez les opérandes

$: /* Détermination quotient et reste d'une division */
$1: Afficher « Saisissez dividende »
DVD **← valeur saisie**
Afficher « Saisissez diviseur »
DVS **← valeur saisie**
Quotient ← partie entière (*DVD*/*DVS*)
Reste ← *DVD* - (QuotientDVS*)**
Afficher « La division de *Dividende* par *Diviseur* a pour Quotient *Quotient* »
Afficher « La division de *Dividende* par *Diviseur* a pour Reste *Reste* »
:1$
:$

Exo #3 : Ecriture du programme

- Pour plus de rigueur, prenons l'habitude d'identifier différemment la chaîne saisie (comme DVD) et la valeur numérique associée DVDN

```
<HTML>
<HEAD>
   <title>Programmatique Exo 3</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Calcul du quotient et du reste d'une division
</B><BR>
<SCRIPT LANGUAGE="JavaScript">
DVD = prompt("Saisissez le dividende DVD ",0);
DVS = prompt("Saisissez le diviseur DVS ",0);
DVDN = DVD*1;
DVSN=DVS*1;
Quotient =  Math.floor(DVDN/DVSN);
Reste = DVDN - (Quotient*DVSN);
document.write("Le quotient de la division " + DVD + "/" + DVS + " est = " + Quotient);
document.write("<BR>");
document.write("Le reste de la division " + DVD + "/" + DVS + " est = " + Reste);
</SCRIPT>
 </FONT>
</p>
</BODY>
</HTML>
```

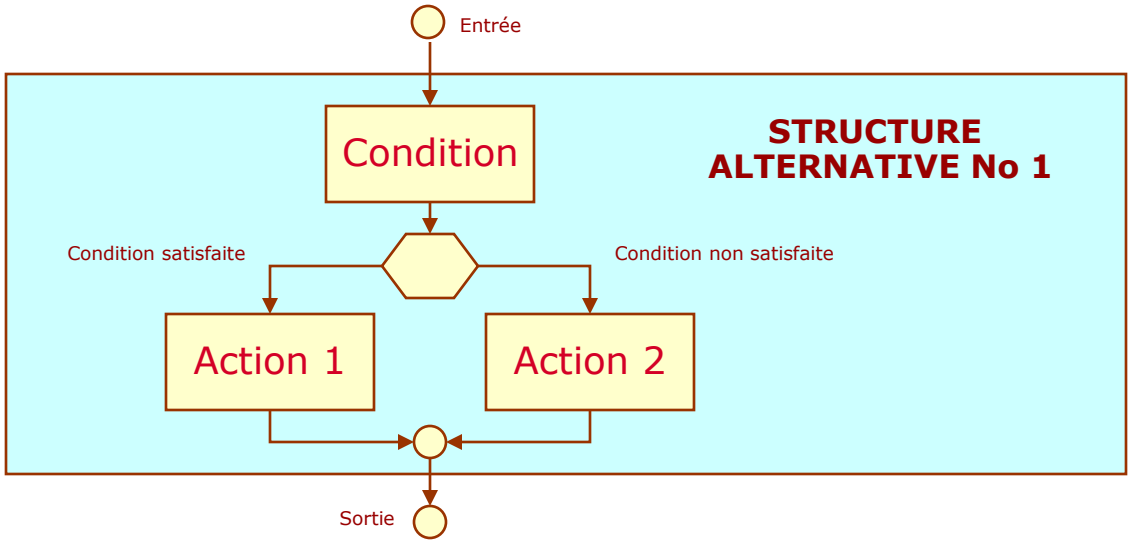
```
$:   /* Détermination quotient et reste d'une division */
$1:  Afficher « Saisissez dividende »
DVD <- valeur saisie
Afficher « Saisissez diviseur »
DVS <- valeur saisie
Quotient <- partie entière (DVD/DVS)
Reste <- DVD - (Quotient*DVS)
Afficher  « La division de Dividende par Diviseur a pour Quotient Quotient »
Afficher  « La division de Dividende par Diviseur a pour Reste Reste »
:1$
:$
```

Structures de la programmation structurée

- Les exercices précédents ont porté sur la structure la plus simple : la structure « Séquence » ou « Bloc ».
- Les instructions se déroulent les unes après les autres en séquence.
- D'autres structures permettent les ruptures de séquences en fonction de la valeur d'une variable.
- Ces ruptures de séquence permettent les processus de décision (si c'est pile, je fais …, sinon je fais ….).
- Ils permettent aussi les processus itératifs (répéter n fois une séquence).

La structure alternative No 1

- $i: **SI** condition (simple ou composée avec des opérateurs booléens)
- **ALORS** $i1: Action 1 (Séquence ou autre structure) :i1$
- **SINON** $i2: Action 2 (id) :i2$
- **FIN SI** :i$



Exemple pour la structure alternative No 1

Exemple : Affichage plus grand de 3 nombres

```
$:  /* Afficher le plus grand des 3 nombres saisis */
$1:  Afficher « Saisissez un premier nombre »
N1 ← valeur saisie
Afficher « Saisissez un deuxième nombre »
N2 ← valeur saisie
Afficher « Saisissez un troisième nombre »
N3 ← valeur saisie     :1$
$2:  SI N1 > N2
    ALORS   $21: PG ← N1 :21$
    SINON   $22: PG ← N2  :22$
FIN SI      :2$
$3:  SI N3 > PG
    ALORS   $31: PG ← N3   :31$
    SINON   $32: Rien        :32$
FIN SI      :3$
Afficher  « Le plus grand nombre saisi est PG »
:$
```


Programmation de la structure alternative No 1

- Si .... alors ..... sinon ..... fin si

```
if (condition)
   {
   …;
   }

if (condition)
   {
   …;
   }
else
   {
   …;
   }
```

A noter que le "end if", équivalent à "fin si", explicite dans certains langages, est implicite en javascript, du fait des bornes matérialisées par les accolades

Programme : le plus grand nombre de 3

```
<HTML>
<HEAD>
  <title>Programmatique Exo 4'</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Afficher le plus grand nombre parmi 3 nombres saisis
</B><BR>
<SCRIPT LANGUAGE="JavaScript">
N1 = prompt("Saisissez un premier nombre N1 ",0);
N2 = prompt("Saisissez un premier nombre N2 ",0);
N3 = prompt("Saisissez un premier nombre N3 ",0);
N1 = N1*1;
N2= N2*1;
N3=N3*1;
if (N1>N2)
  {
   PG = N1;
  }
  else
 {
  PG = N2;
 }
if (N3>PG)
  {
  PG = N3;
  }
document.write("Le plus grand nombre saisi est " + PG);
document.write("<BR>");
</SCRIPT>
 </FONT>
</p>
</BODY>
</HTML>
```

La structure alternative No 2

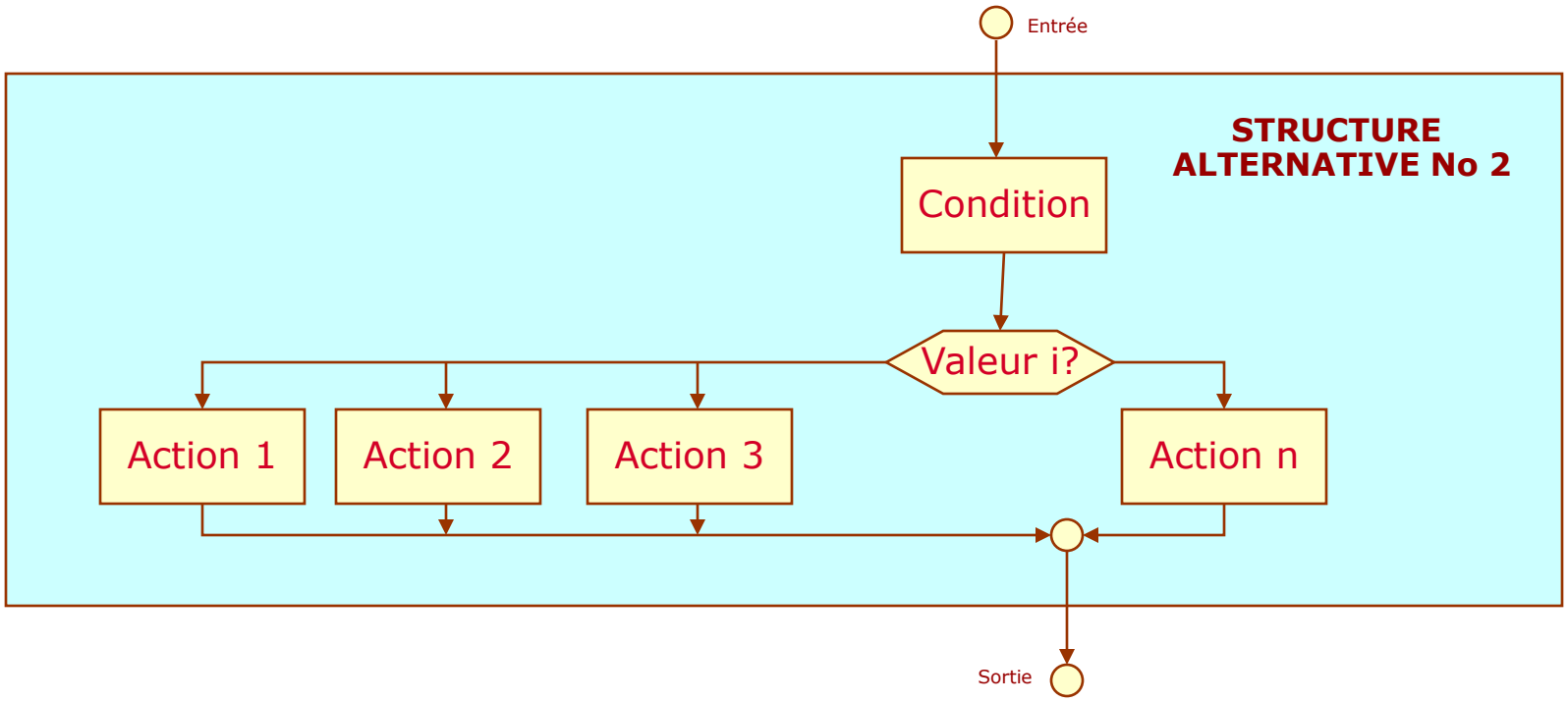
- $i: **SELON** indicateur
- **CAS** indicateur = i1

  $i1: Action 1 (Séquence ou autre structure) :i1$
- **CAS** indicateur = i2

  $i2: Action 2 (id) :i2$
- …….
- **CAS** toutes autres valeurs de l'indicateur

  $in: Action n (id) :in$
- **FIN SELON** :i$

La structure alternative No 2



Programmation de la structure alternative No 2

- La structure ***case of*** à choix multiples
- Selon la variable nom, cas nom = i1 …, cas i = i2 …, cas autres valeurs …..

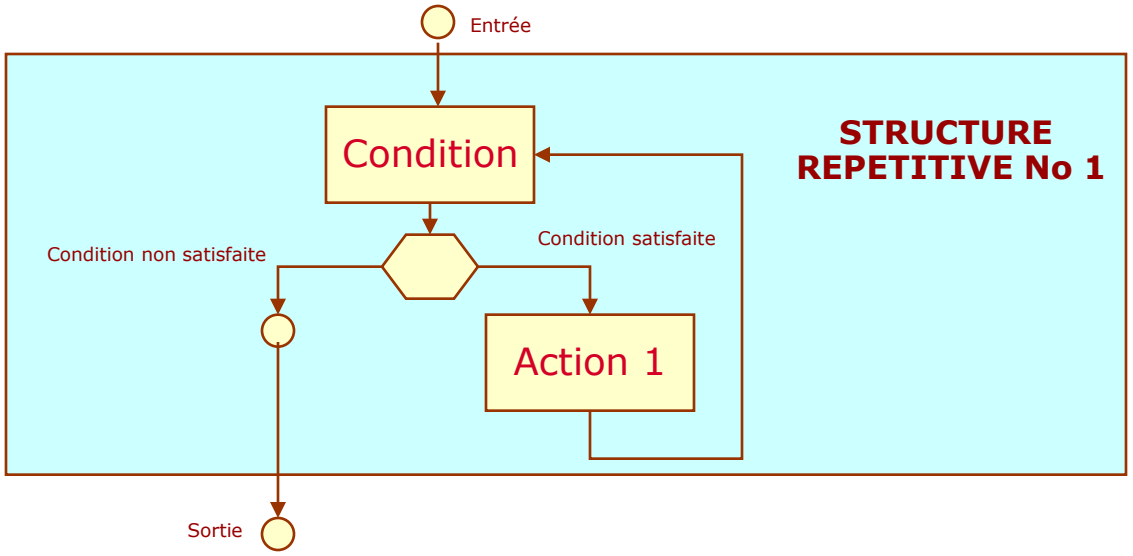
```
switch (nom)
   {
   case i1;
        … ;
        break;

    case i2;
        …;
        break;

    default;
        … ;
        break;
    }
```

L'utilisation de la structure SELON est souvent liée à un phénomène itératif.

Nous verrons une application de cette structure imbriquée dans une structure répétitive

La structure répétitive No 1

- $i: **TANT QUE** condition (simple ou composée avec des opérateurs booléens)
- **REPETE** $i1: Action 1 (Séquence ou autre structure) :i1$
- **FIN TANT QUE** :i$
- **Attention** : La condition peut ne pas être vérifiée lors de la première exécution, auquel cas le bloc Action 1 ne sera pas exécuté.
- **Attention** : Boucle sans fin si la condition est satisfaite et qu'elle n'évolue pas dans le cours d'Action 1



Exemple pour la structure répétitive No 1

Exemple No 5 : Deuxième version de la somme de n entiers

```
$:  /* Exercice #5 V2 de somme de n entiers */
$1:  /* Initialisation */
Afficher « Saisissez un nombre entier»
N ← valeur saisie
Nref ← N
Somme ← 0                    :1$
$2:  /* Boucle de calcul */
TANT QUE N > 0
     REPETE $21: Somme <- Somme + N
                 N ← N-1  :21$
FIN TANT QUE     :2$
$3:  /* Livraison résultats */
Afficher  « La somme des »  Nref « premiers entiers est » Somme  :3$
:$
```


Programmation de la structure répétitive No 1

- La structure répétitive Tant Que ..... Répète

```
while (condition)
  {
        ……;
        évolution condition;
         ….;
   }
```

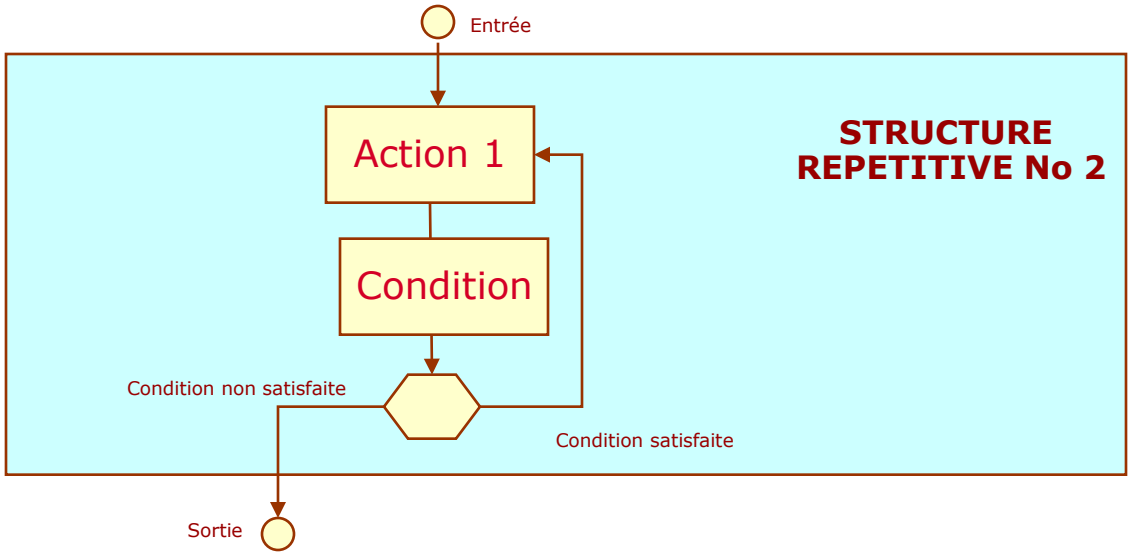
A noter que le "wend", équivalent à "fin tant que", explicite dans certains langages (par exemple dans notre exemple en small basic du chapitre 2), est implicite en javascript, du fait des bornes matérialisées par les accolades

Programme : Nouvelle version de la somme de n nombres

```
<HTML>
<HEAD>
  <title>Programmatique Exo 5'</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Somme des n premiers nombres
</B><p>
<SCRIPT LANGUAGE="JavaScript">
/* Exercice #5 V2 de somme de n entiers */
/* Initialisation */
var N = 0;
var NRef = 0;
var Somme = 0;
N = prompt("Saisissez un nombre entier",0);
N=N*1;
Nref = N;
Somme = 0 ;
/* Boucle de calcul */
while (N > 0)
    {
    Somme = Somme + N;
     N = N-1 ;
    }

/* Livraison résultats */
document.write("La somme des " + Nref + " premiers entiers est " + Somme +
"<BR>");
</SCRIPT>
 </FONT>
</BODY>
</HTML> </HTML>
```

La structure répétitive No 2

- $i: **REPETE** $i1: Action 1 (Séquence ou autre structure) :i1$
- **TANT QUE** condition (simple ou composée avec des opérateurs booléens)
- :i$
- **Attention** : Elle diffère de la précédente en ce que le bloc d'instructions sous contrôle est au moins exécuté une fois
- **Attention** : Boucle sans fin si la condition est satisfaite et qu'elle n'évolue pas dans le cours d'Action 1



Programmation de la structure répétitive No 2

- La structure répétitive Répète (Faire) ..... tant que .....

```
do
   {
   ……;
   évolution condition;
   }
while (condition);
```

- Variante Répète … jusqu'à ce que … (*Repeat … until* ….)

Programme : Nouvelle version de la somme de n nombres

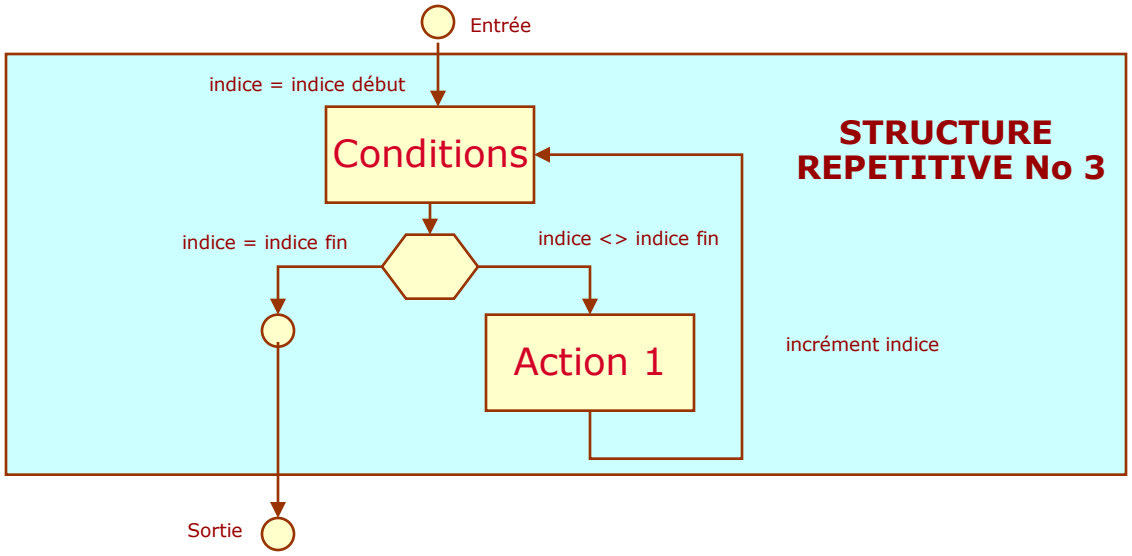
```
<HTML>
<HEAD>
  <title>Programmatique Exo 5bis'</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Somme des n premiers nombres
</B><p>
<SCRIPT LANGUAGE="JavaScript">
/* Exercice #6 V3 de somme de n entiers */
/* Initialisation */
var N = 0;
var NRef = 0;
var Somme = 0;
N = prompt("Saisissez un nombre entier",0);
N=N*1;
Nref = N;
Somme = 0;

/* Boucle de calcul */
do
   {
   Somme = Somme + N;
    N = N-1 ;
   }
while (N>0);

/* Livraison résultats */
document.write("La somme des " + Nref + " premiers entiers est " + Somme +
"<BR>");

</SCRIPT>
 </FONT>
</BODY>
</HTML> </HTML>
```


Programme : Nouvelle version de la somme de n nombres

- Test et interprétation
- Tester les 2 versions avec un nombre négatif

La structure répétitive No 3

- $i: **POUR** condition (indice départ, indice fin, nature incrémentation)
- $i1: Action 1 (Séquence ou autre structure) :i1$
- **SUIVANT** :i$
- **Attention** : Elle diffère des précédentes par la définition au préalable de toutes les conditions de la boucle : il n'est nul besoin de faire évoluer la condition de sortie dans le bloc d'instructions sous contrôle.



Répétitive 3 : Pour ..... Suivant

- La structure répétitive Pour ... Suivant

```
for (initialisation; condition; increment)
  {
  ...;
  }
```

A noter que le “next”, équivalent à “suivant”, explicite dans certains langages, est implicite en javascript, du fait des bornes matérialisées par les accolades

Programme : Nouvelle version de la somme de n nombres

```
<HTML>
<HEAD>
  <title>Programmatique Exo 5terprim'</title>
</HEAD>
<BODY >
<p ALIGN=left>
<FONT SIZE="2" FACE="arial" >
<B> Somme des n premiers nombres
</B><p>
<SCRIPT LANGUAGE="JavaScript">
/* Exercice #7 V4 de somme de n entiers */
/* Initialisation */
var N = 0;
var Nref = 0;
var Somme = 0;
N = prompt("Saisissez un nombre entier positif",0);
N=N*1;
Nref=N
Somme = 0 ;

/* Boucle de calcul */
for (N=Nref; N>=0; N--)
    {
    Somme = Somme + N;
    }

/* Livraison résultats */
document.write("La somme des " + Nref + " premiers entiers est " + Somme +
"<BR>");
</SCRIPT>
 </FONT>
</BODY>
</HTML>
```


Programme : Nouvelle version de la somme de n nombres

- Test et interprétation

Quelles réponses à nos questions ?

- La programmation est-elle un art ou une technique ?
- Longtemps considérée comme un art, la programmation est devenue une technique avec le développement de la théorie de la programmation structurée

Quelles réponses à nos questions ?

- Qu'est-ce que la programmation impérative ?
- Un programme impératif décrit les opérations en termes de séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme.
- Ce type de programme est le plus répandu et se différencie des productions de la programmation déclarative -ou logique- et de la programmation dite fonctionnelle.

Quelles réponses à nos questions ?

- Qu'est-ce qu'une opération d'affectation ?
- Affectation d'une valeur (nombre, texte, booléen) à une variable ou à une constante.

Quelles réponses à nos questions ?

- Quelles sont les structures de base de la programmation structurée ?
- Séquence / Bloc / Enchaînement
- Structures alternatives
  - SI condition ALORS action1 SINON action2 FIN SI
  - SELON indicateur CAS 1 action1 CAS 2 action2 CAS 3 action3 ... AUTRES CAS actionN FIN CAS
- Structures répétitives
  - TANT QUE condition REPETE action FIN TANT QUE
  - REPETE action TANT QUE condition
  - POUR condition, action SUIVANT