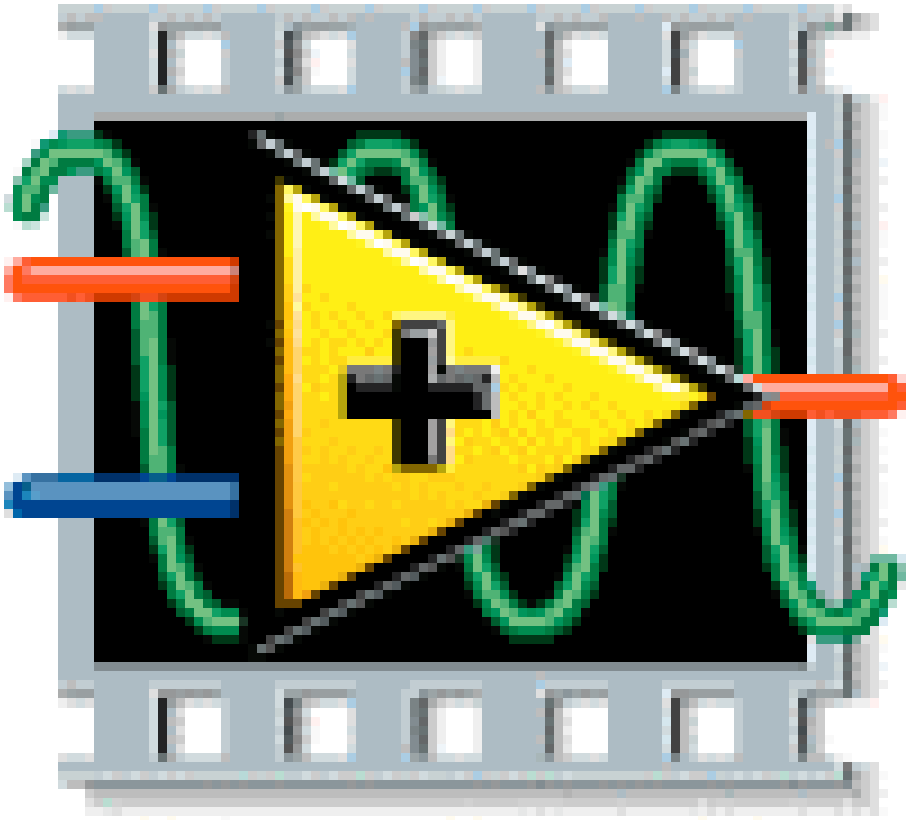
BELFORT
MONTBÉLIARD

Mesures Physiques



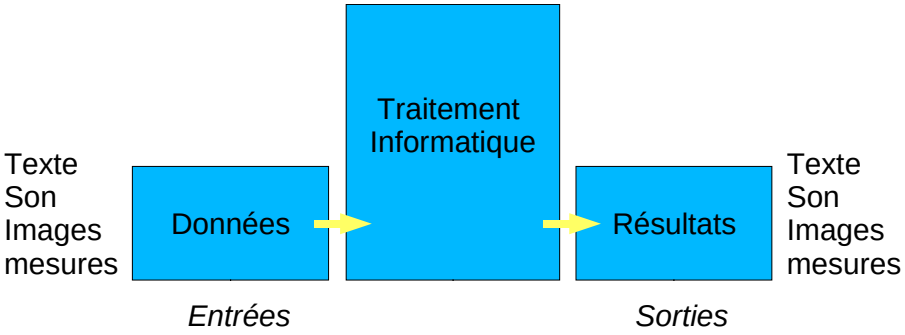
UNIVERSITÉ DE FRANCHE-COMTÉ

COURS LabVIEW

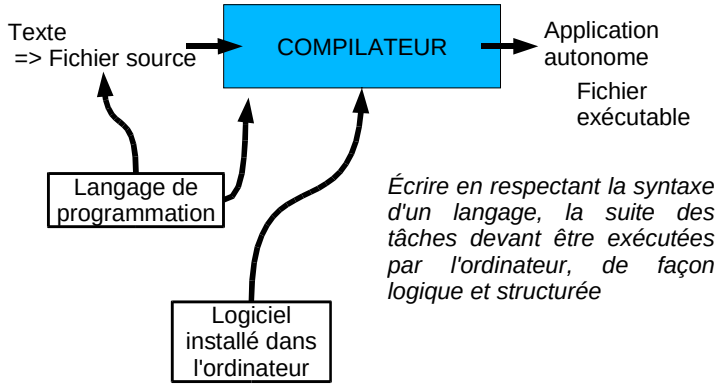
Chapitre 1 – INTRODUCTION

I – LA PROGRAMMATION

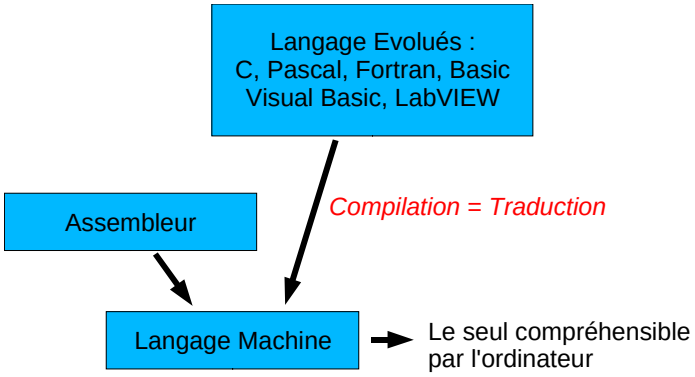
Un ordinateur en fonctionnement réalise des tâches pour lesquelles il a été programmé.



Le programme est écrit dans un langage de programmation, en respectant une syntaxe propre au langage choisi.



Exemples de langages de programmation :

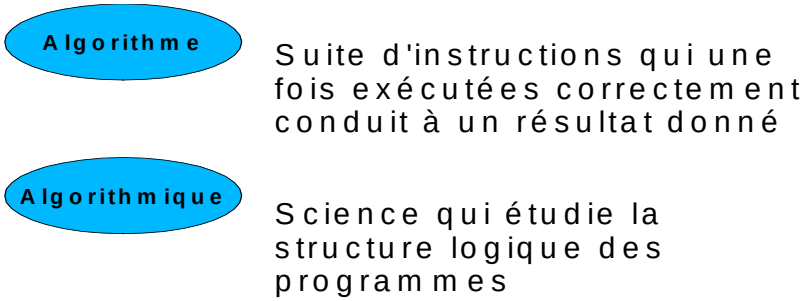


II – ALGORITHME

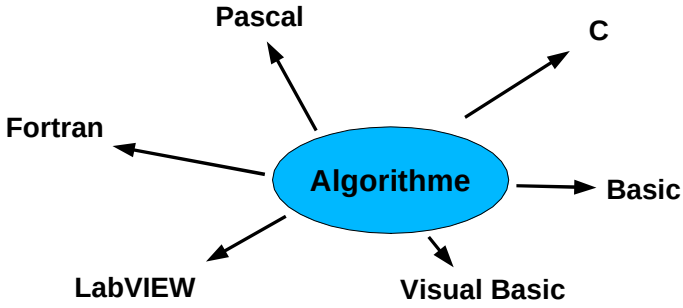
Programmer, c'est écrire en respectant la syntaxe d'un langage, la suite des tâches devant être exécutées par l'ordinateur, de façon logique et structurée.

On ne se lance pas à écrire un programme sans avoir réfléchi au problème posé, et à la façon de le résoudre.

Il faut élaborer un algorithme :



L'algorithme est indépendant du langage :



L'algorithme se ramène à une combinaison de 4 familles d'instructions :

- Affectation de variables
- Lecture écriture
- Tests
- Boucles

III – QU'EST CE QUE LabVIEW ?

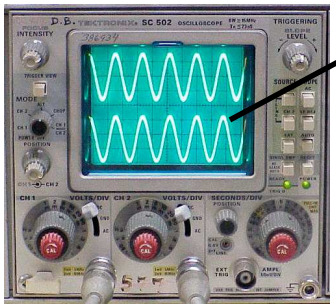
LabVIEW est un logiciel créé en 1985. C'est un logiciel de développement d'applications d'instrumentation plus particulièrement destiné à l'acquisition des données de mesure et à leur traitement.
Il utilise un langage de programmation graphique.

Cet enseignement a été mis en place en DUT MP à Montbéliard en 1995.
La version utilisée en TP est la version 7.1 sous Linux ou Windows.

IV – STRUCTURE D'UN PROGRAMME LABVIEW

LabVIEW étant plus particulièrement destiné à l'instrumentation, un programme LabVIEW apparaît constitué comme un appareil de mesure :

Une face avant :



La face avant est l'interface avec l'utilisateur.

Un diagramme :

Le diagramme correspond aux circuits internes de l'appareil et constitue le cœur du programme

V – LES ENTREES SORTIES

1°/ DEFINITION

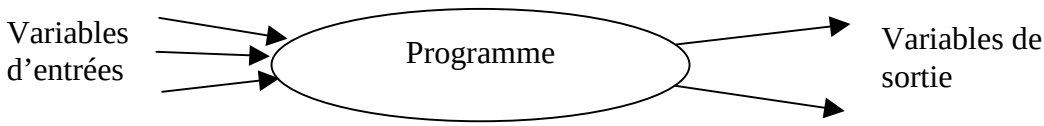
Un programme informatique manipule des données fournies par l'environnement extérieur :

- Entrées au clavier par l'utilisateur
- Acquises par une carte d'acquisition (mesure de Température par exemple)
- Délivrées par un appareil de mesure (Voltmètre, via une liaison informatique)
- Tirées d'un fichier tableur
- Etc…

Ces données constituent des variables d'entrée.

Leur traitement (par exemple : filtrage, analyse fréquentielle …) donne naissance à des résultats (affichage d'un graphe d'évolution de la température mesurée) ou engendre des actions de commande délivrées par l'ordinateur (commande d'arrêt de chauffage).

Ces informations constituent des variables de sortie.



L'élaboration d'un algorithme nécessite de bien identifier les entrées/sorties.

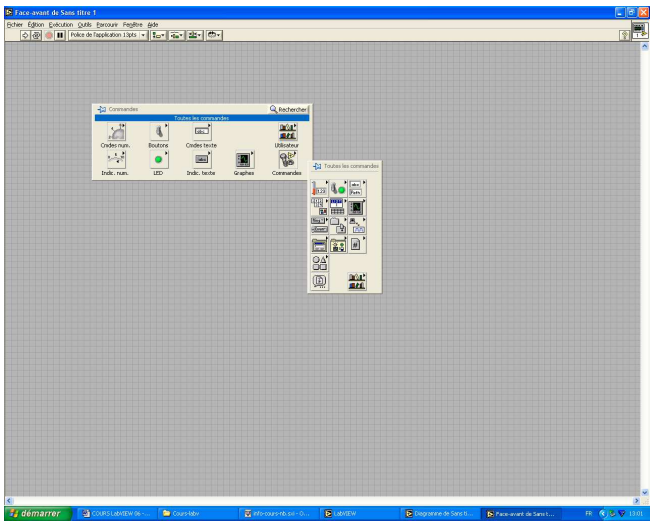
2°/ ENTREES SORTIE DANS LabVIEW

Les entrées sorties sont déposées sur la face avant sous la forme d'objets tels que :

entrées
- boutons poussoirs
- interrupteurs
- potentiomètres rotatifs ou à glissière
- Afficheur numérique en lecture.
- Etc …

sorties
- écrans d'oscilloscope
- Afficheurs numériques en écriture (affichage)
- Vu-mètres
- Etc …

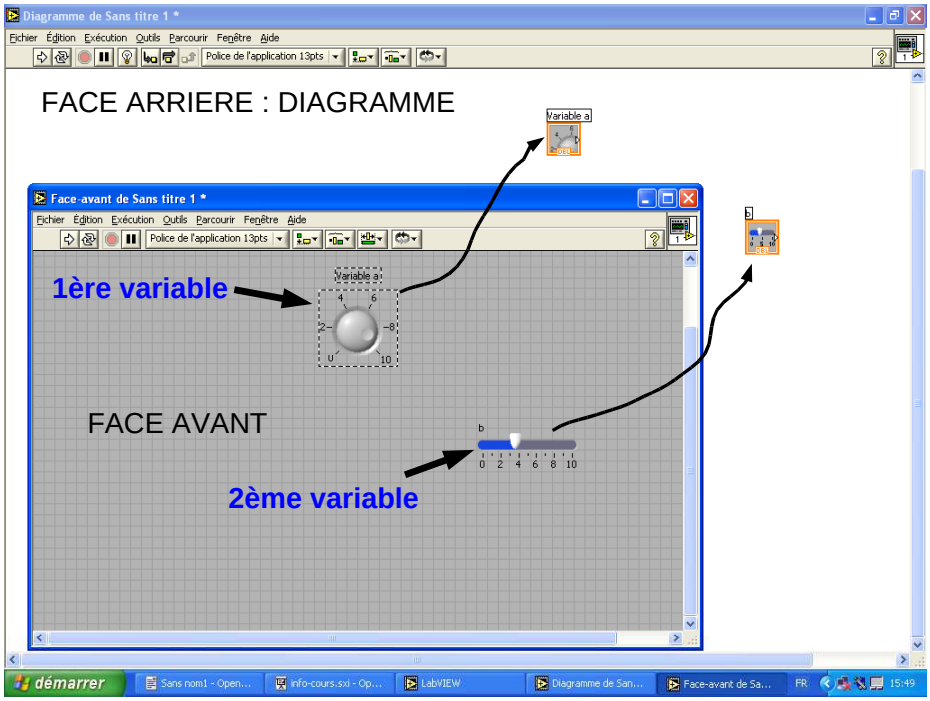
Ces objets sont disponibles dans la palette de commandes qui apparaît par clic droit sur la face avant. On les place par « glissé déposé ».



VI – LE PROGRAMME LabVIEW

Le programme est écrit en langage graphique dans une deuxième fenêtre «diagramme » indissociable de la face avant.

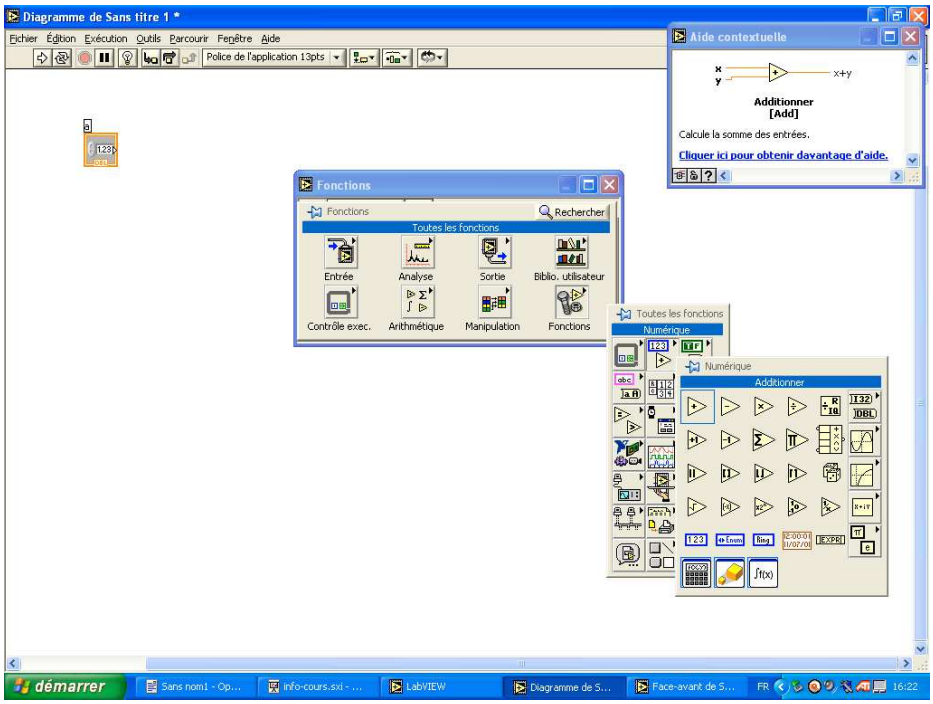
Les entrées sorties apparaissent alors sous la forme de terminaux.



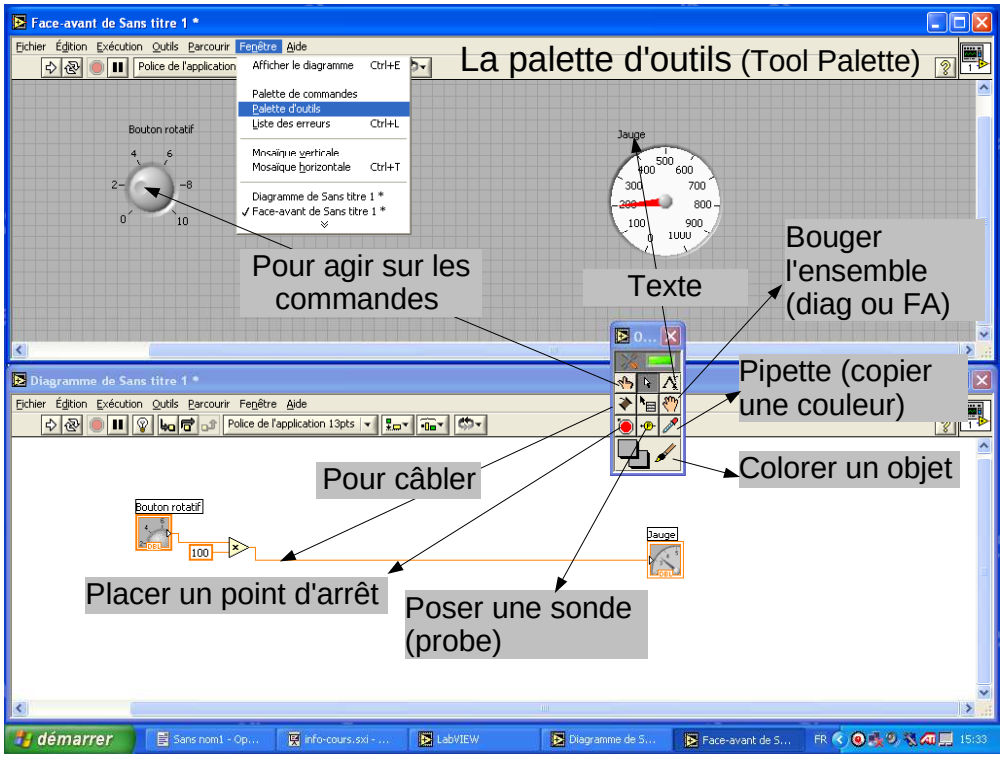
L'écriture du programme consiste à relier entre eux les terminaux d'entrées sorties en utilisant :

- des opérateurs arithmétiques
- des opérateurs booléens
- des fonctions de calcul toutes prêtes
- des structures de programmation (boucles, tests etc …)

Ces outils sont disponibles dans la palette fonction apparaissant par clic droit dans le diagramme.



Les liaisons sont réalisées à l'aide de la bobine de la palette d'outils disponible par le menu Fenêtre/palette d'outils (Windows/tools palette).



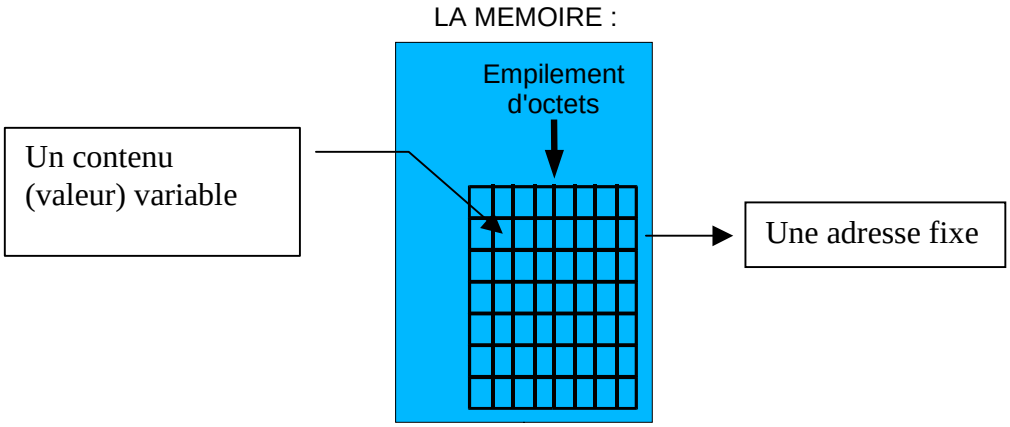
L'ensemble face avant + diagramme constitue un programme LabVIEW encore appelé VI (Virtual Instrument).

Ce VI peut être «encapsulé » c'est à dire mis en boîte sous la forme d'une icône constituant un sous programme, une fonction pouvant être utilisée dans un autre diagramme.

Chapitre 2 – LES VARIABLES ARITHMETIQUES

I – DECLARATION DE VARIABLE

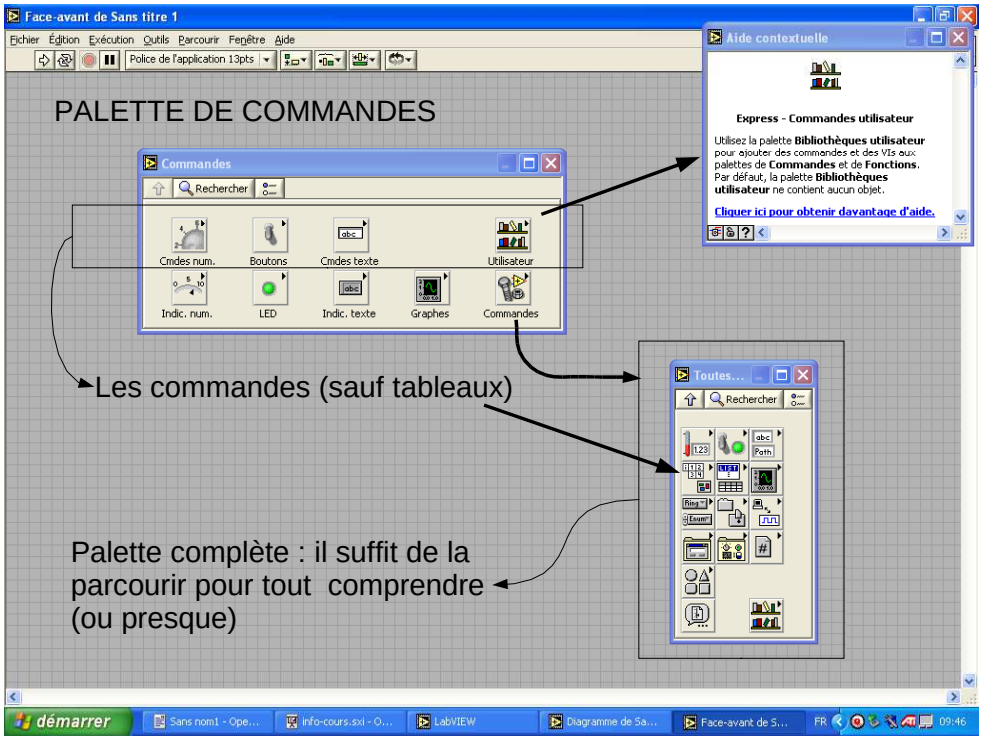
Un programme manipule des valeurs contenues dans des variables.
A chaque variable correspond un emplacement mémoire repéré par une adresse précise, et dont le contenu est la valeur prise par la variable au cours de l'exécution du programme.

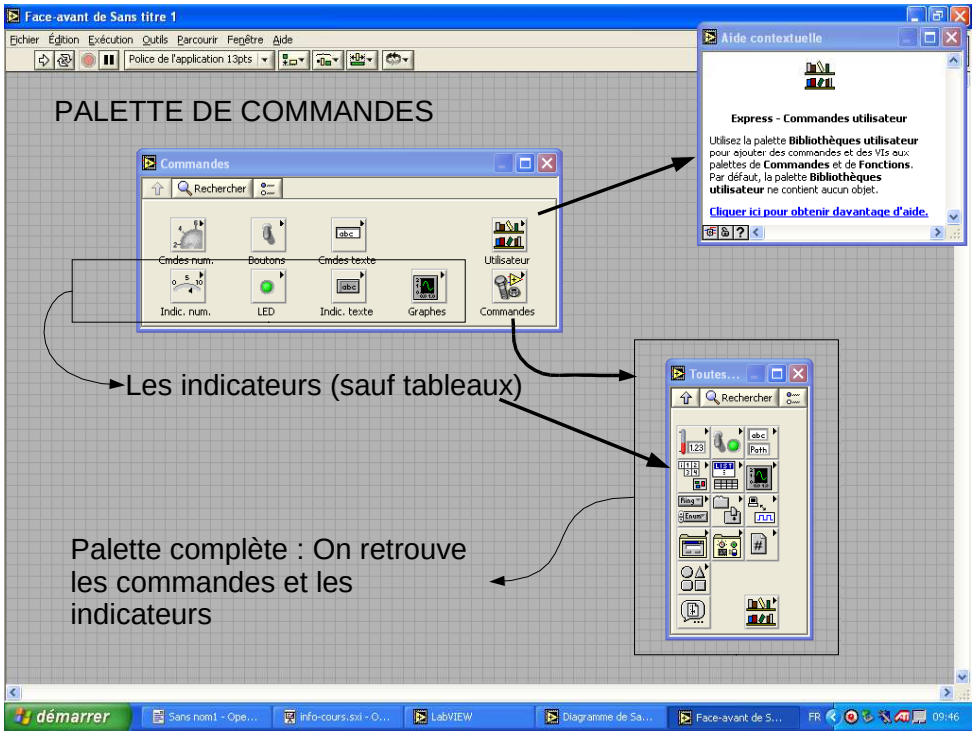


Lorsque l'on dépose une variable d'entrée ou sortie sur la face avant d'un VI, on définit :

- Le nom de la variable : son étiquette
- La représentation ou le type de la variable : la place en octet occupée par la variable et le mode de codage.

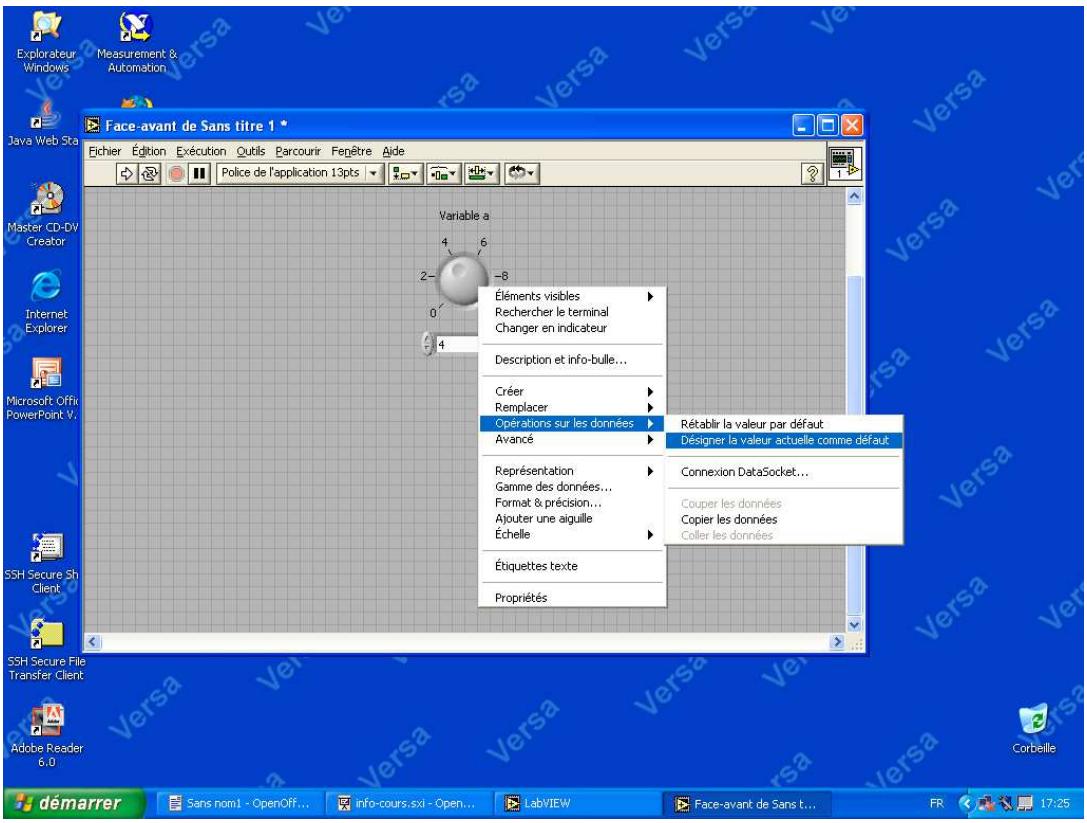
II – ENTREES SORTIES DANS LabVIEW





III – INITIALISATION

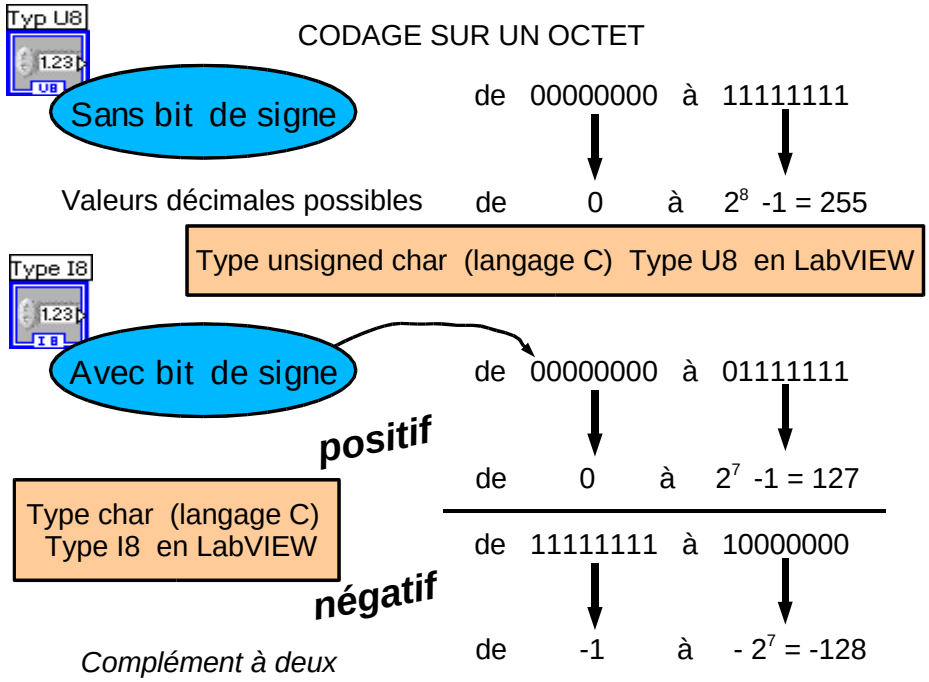
Initialiser la variable, c'est lui donner une valeur qu'elle prendra dès le début du programme.
Pour cela un clic droit sur l'objet, puis opérations sur les données puis valeur actuelle par défaut.



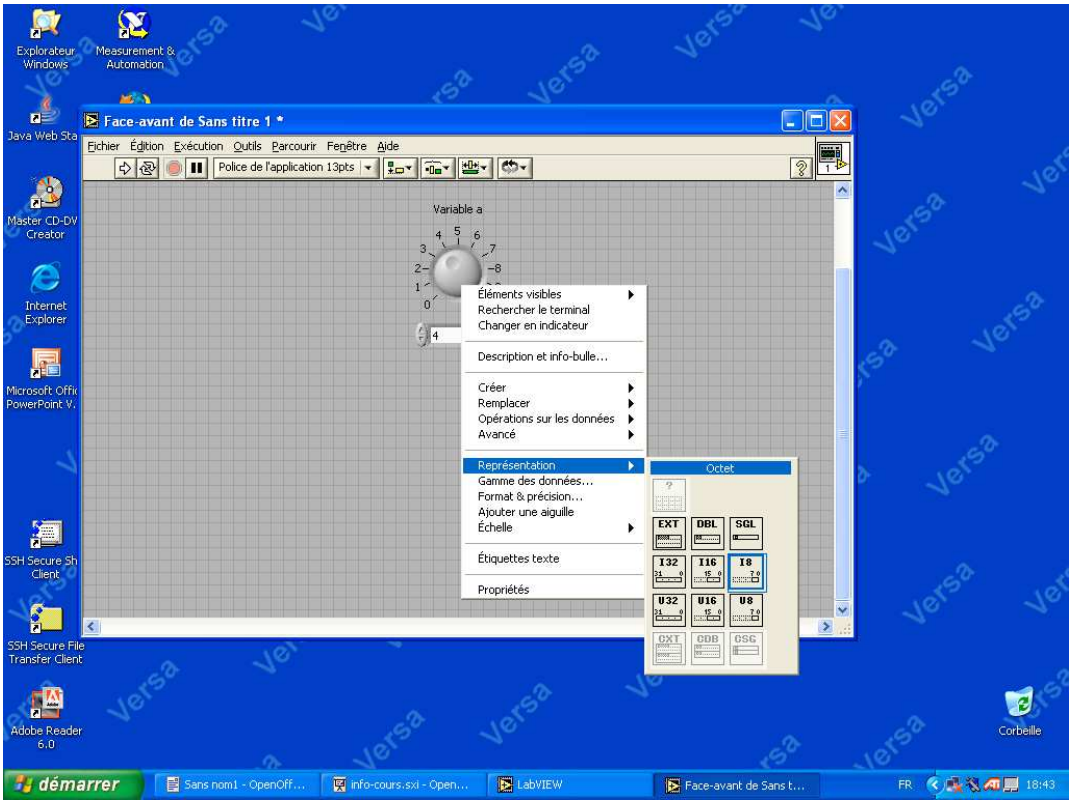
IV – TYPE DE VARIABLE

1°/ CODAGE DES NOMBRES ENTIERS

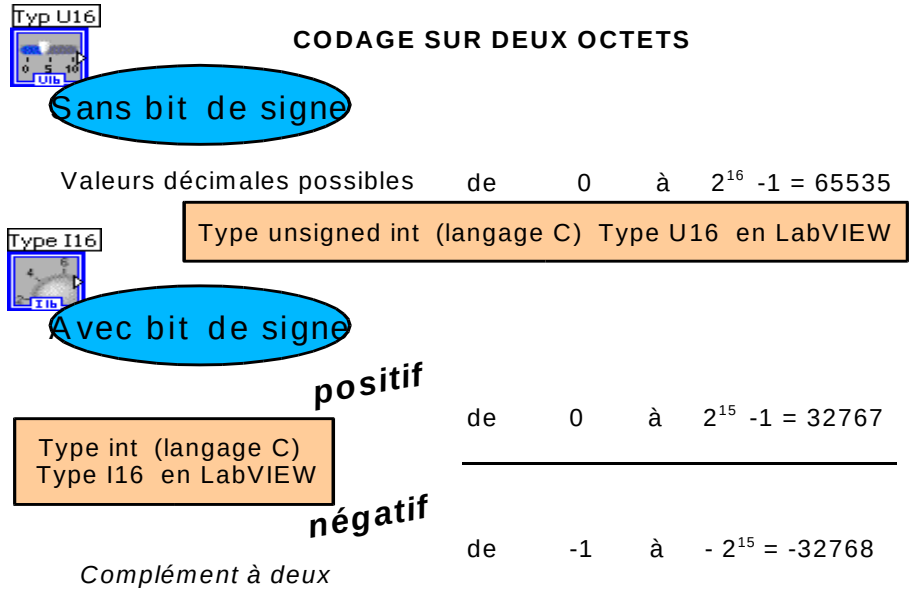
a – Codage sur un octet



b - Comment changer la représentation dans LabVIEW ?

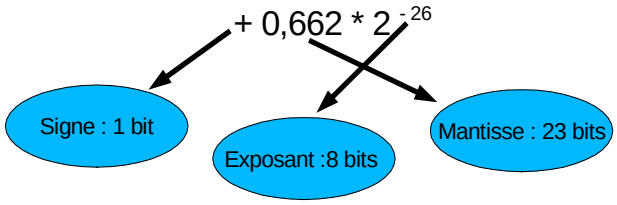


c – Codage sur 2 octets



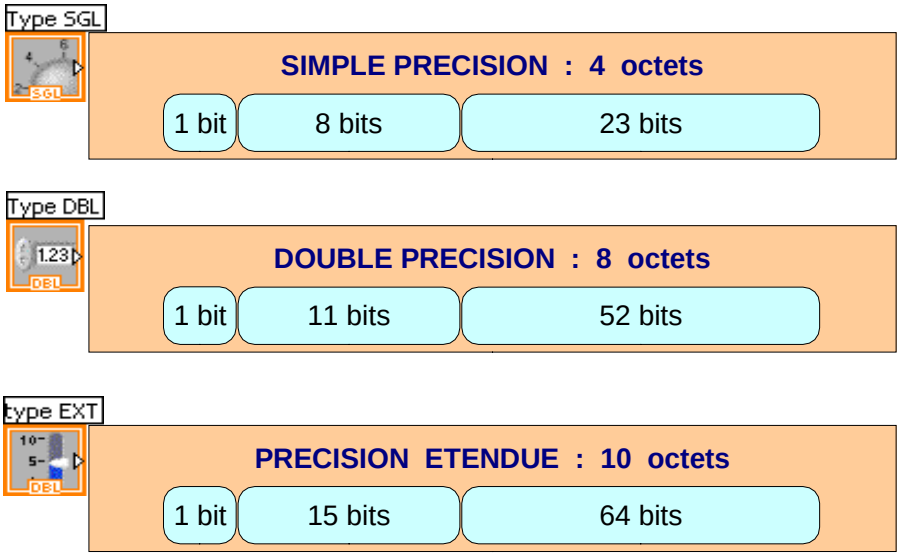
2°/ NOMBRES A VIRGULE FLOTTANTE

Un nombre à virgule flottante peut toujours s'écrire :

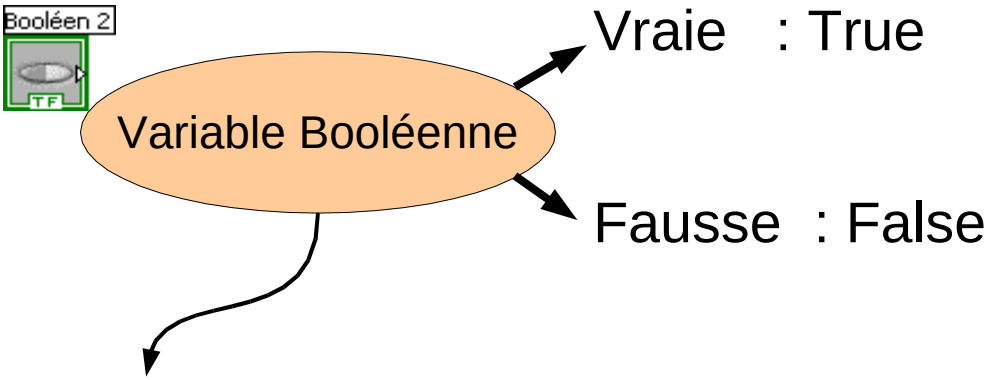


Valeurs extrêmes possibles :
- Nombre nul : 0
- Nombre positifs de : 1,21 10^{-38} à 3,40 10^{38}
- Nombres négatifs de : - 1,21 10^{-38} à - 3,40 10^{38}

Type : Float en C, Type SGL en LabVIEW



3°/ VARIABLES BOOLEENNES

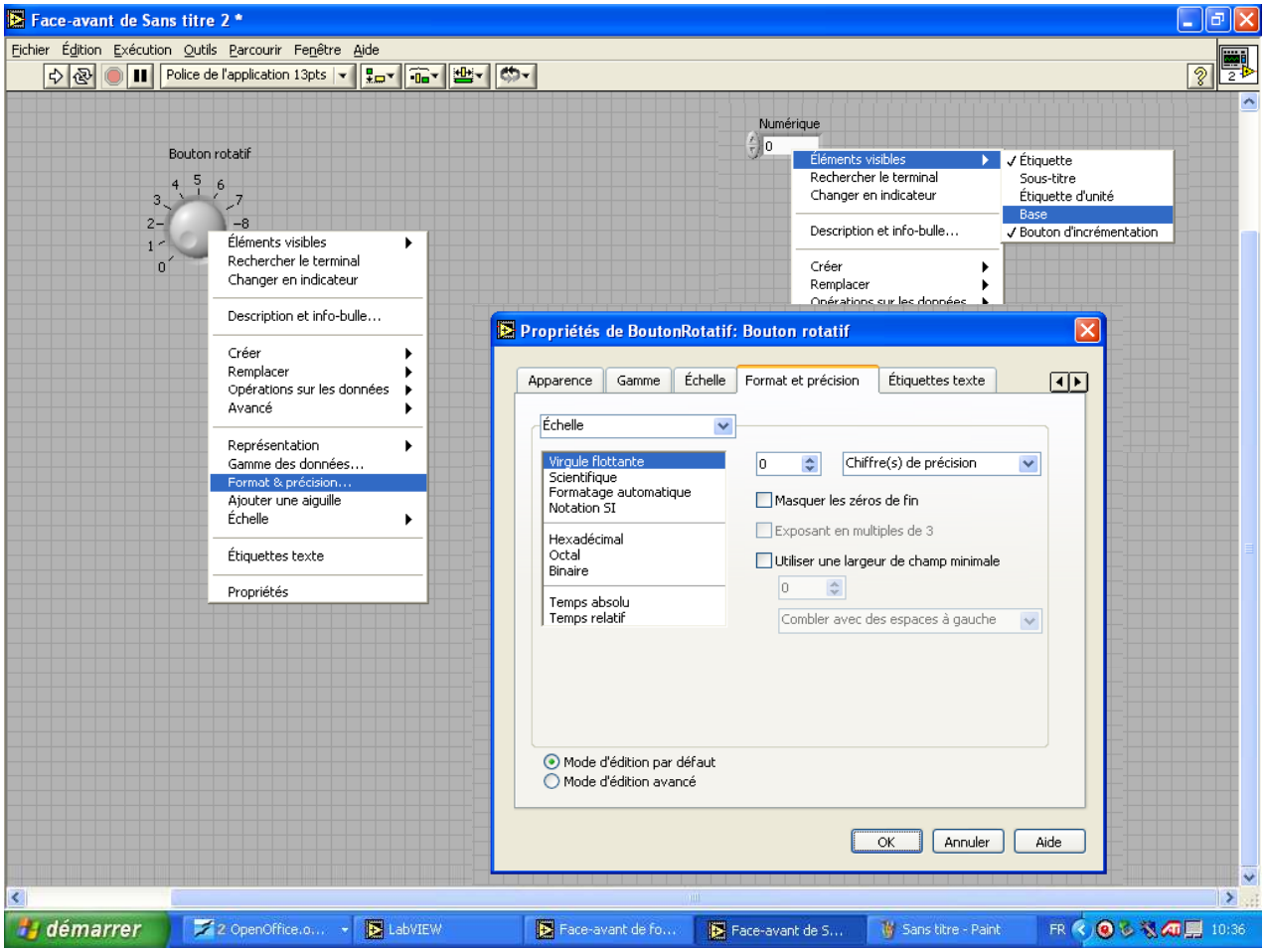


Codage sur 1 seul bit

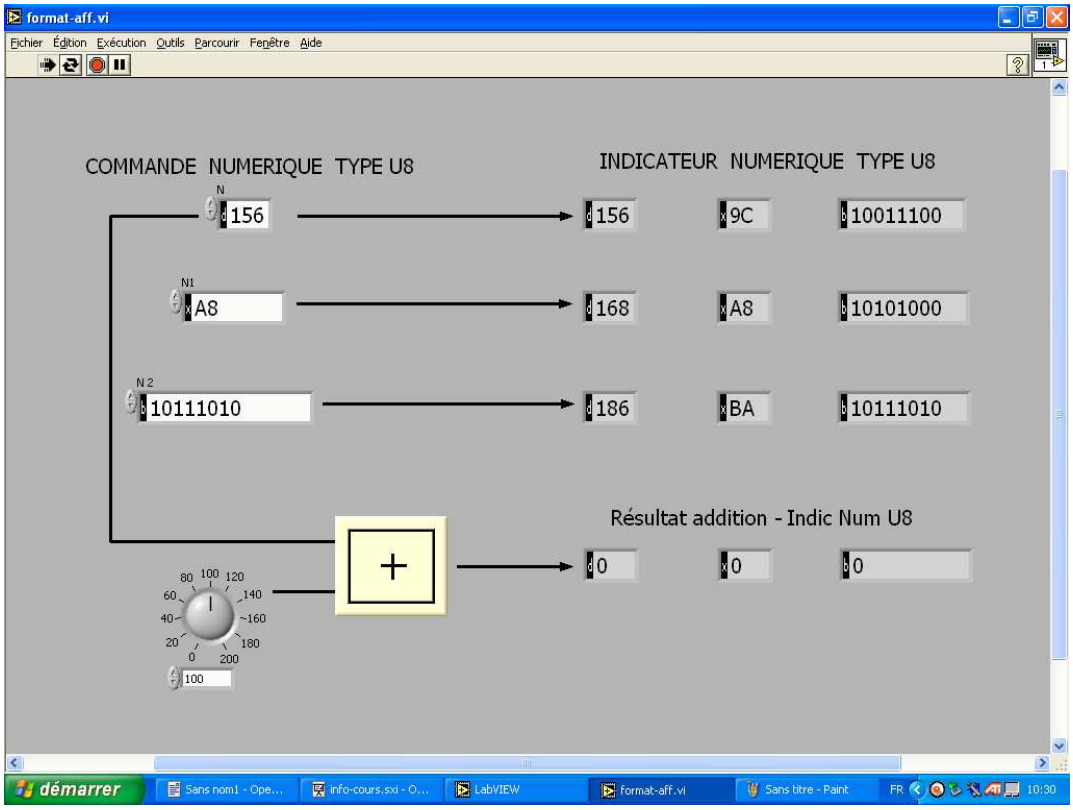
N'existe pas dans tous les langages :

OK en labVIEW mais pas en C !

V – FORMATS D'AFFICHAGE



Exemples de formats d'affichages :

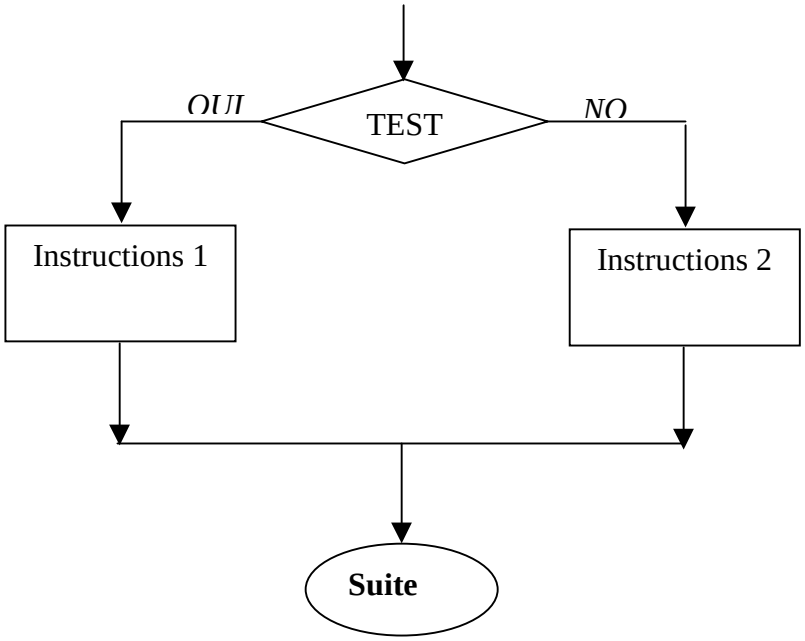


Chapitre 3 – LES TESTS

I – DEFINITION

Suivant le résultat d'un test, on réalise une série d'instructions ou une autre

II – REPRÉSENTATION



REMARQUES :

- Pas de règle sur le choix gauche ou droite du côté OUI et du coté NON.
- Un des blocs Instructions 1 ou 2 peut être vide (ne rien faire). Il suffit de ne pas faire apparaître le cadre.

III – LE TEST

Le test est la relation permettant de réaliser l'aiguillage.

Le test met en œuvre des opérateurs de comparaison $<$, $>$, $\leq$, $\geq$, $=$, $\neq$.
Il peut être aussi l'évaluation d'une simple variable booléenne (test de l'appui sur un bouton poussoir)

Le résultat du test est un booléen : il ne peut prendre que 2 valeurs : OUI ou NON.
Le test peut être simple du type $a < b$ ou plus complexe : $(a < b)$ ET $(a < c)$

Les relations de comparaisons sont alors reliées entre elles par des opérateurs logiques : ET, OU, NON ET, NON OU, OU Exclusif, etc …

La relation correspondant au test est écrite dans le losange.

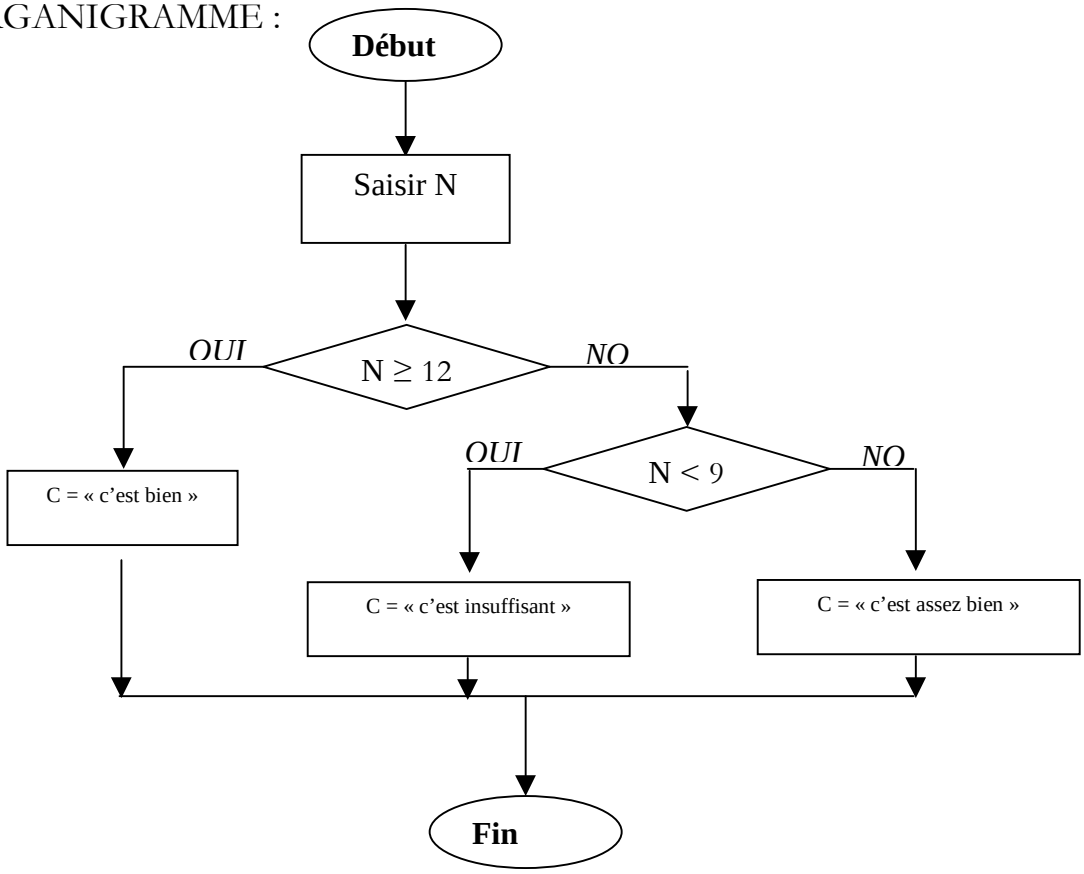
IV – EXEMPLE

Saisir une note N au clavier.
Afficher sur une chaîne de caractère le message «c'est bien » si $N \geq 12$
Afficher «c'est assez bien » si $9 \leq N < 12$
Afficher «c'est insuffisant » si $N < 9$

ENTREES SORTIES :

- Une entrée nommée N de type commande numérique.
- Une sortie nommée C afficheur chaîne de caractère.

ORGANIGRAMME :



Cet organigramme montre que les structures conditionnelles peuvent être imbriquées.

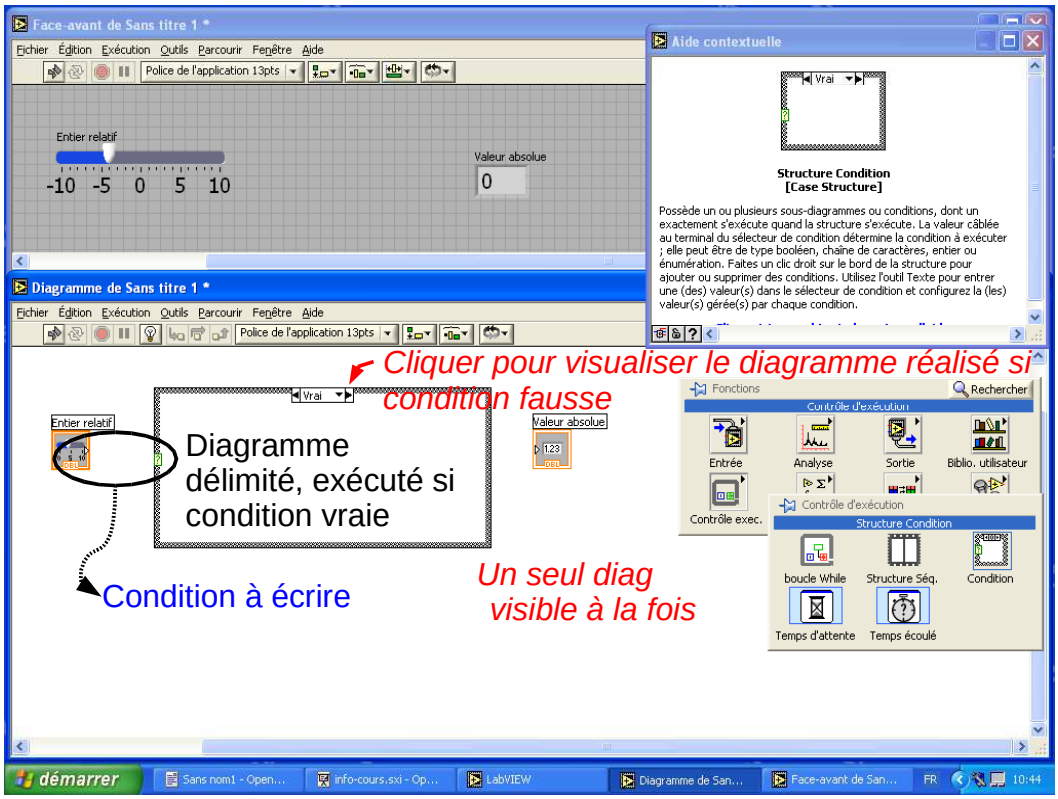
V – STRUCTURE CONDITIONNELLE DANS LabVIEW

Un clic droit dans le diagramme permet d'ouvrir la palette de fonctions. Cliquer sur l'icône contrôle d'exécution. On trouve alors la structure condition.

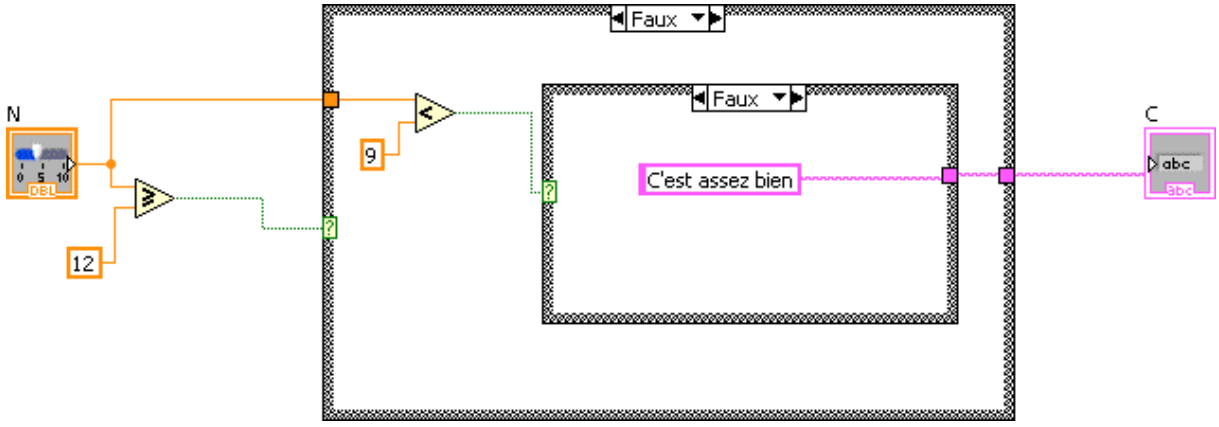
Le résultat booléen du test est connecté sur le point d'interrogation vert.

Les instructions réalisées si le résultat du test est OUI sont écrites à l'intérieur de la structure lorsque VRAI apparaît en haut du rectangle délimité.

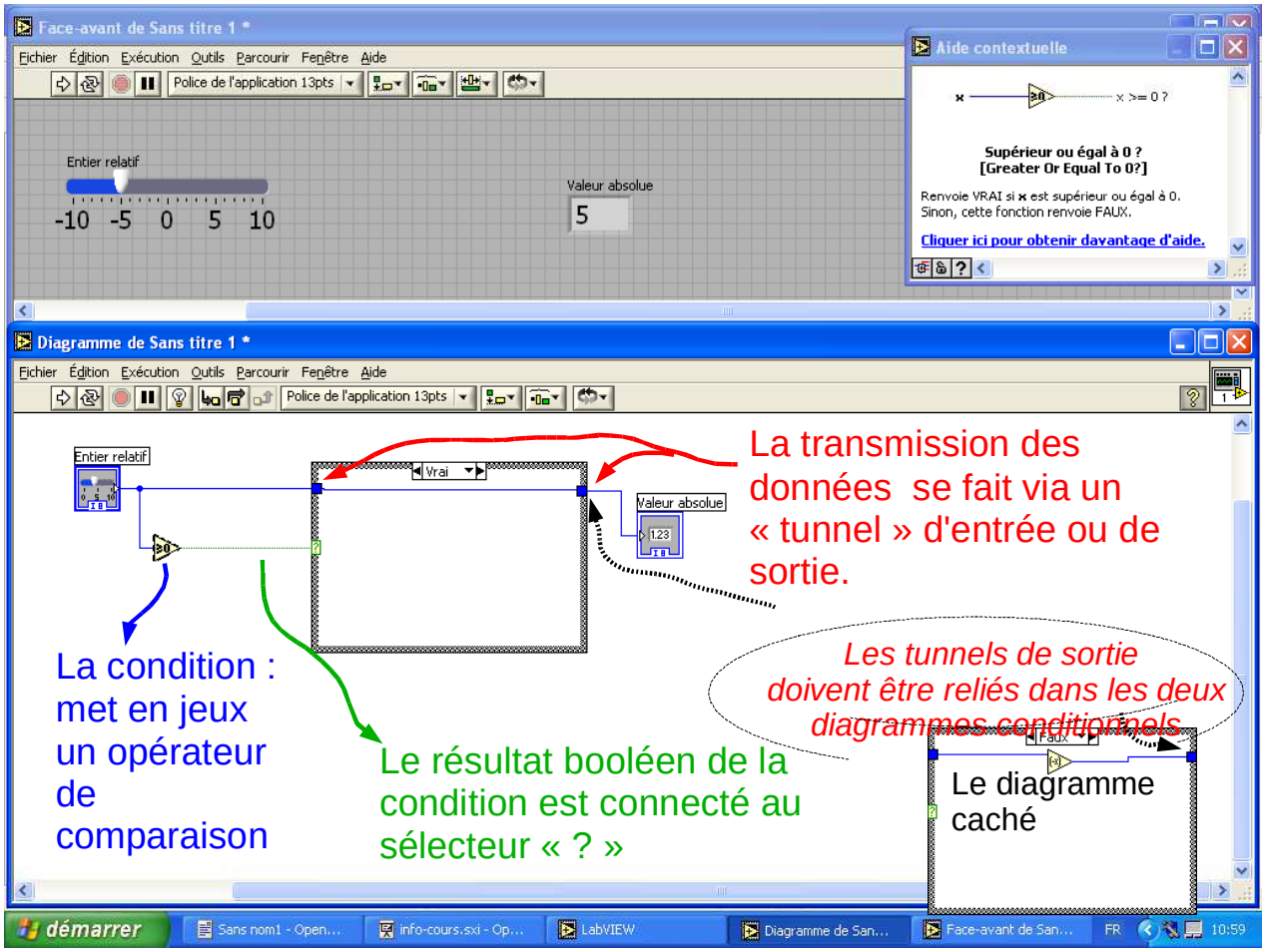
Les instructions réalisées si le résultat du test est NON sont écrites à l'intérieur de la structure lorsque FAUX apparaît en haut du rectangle délimité.
Un seul cas est visible simultanément, cliquer sur les triangles pour passer de l'un à l'autre.

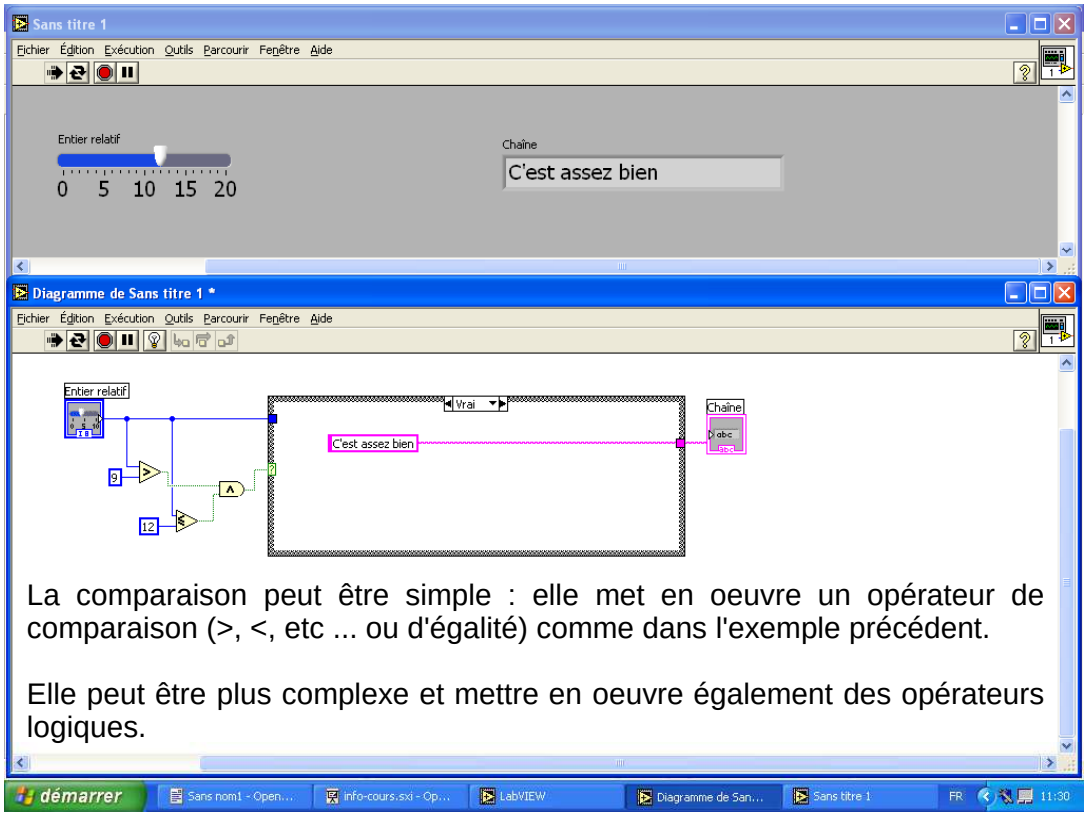


EXEMPLE :



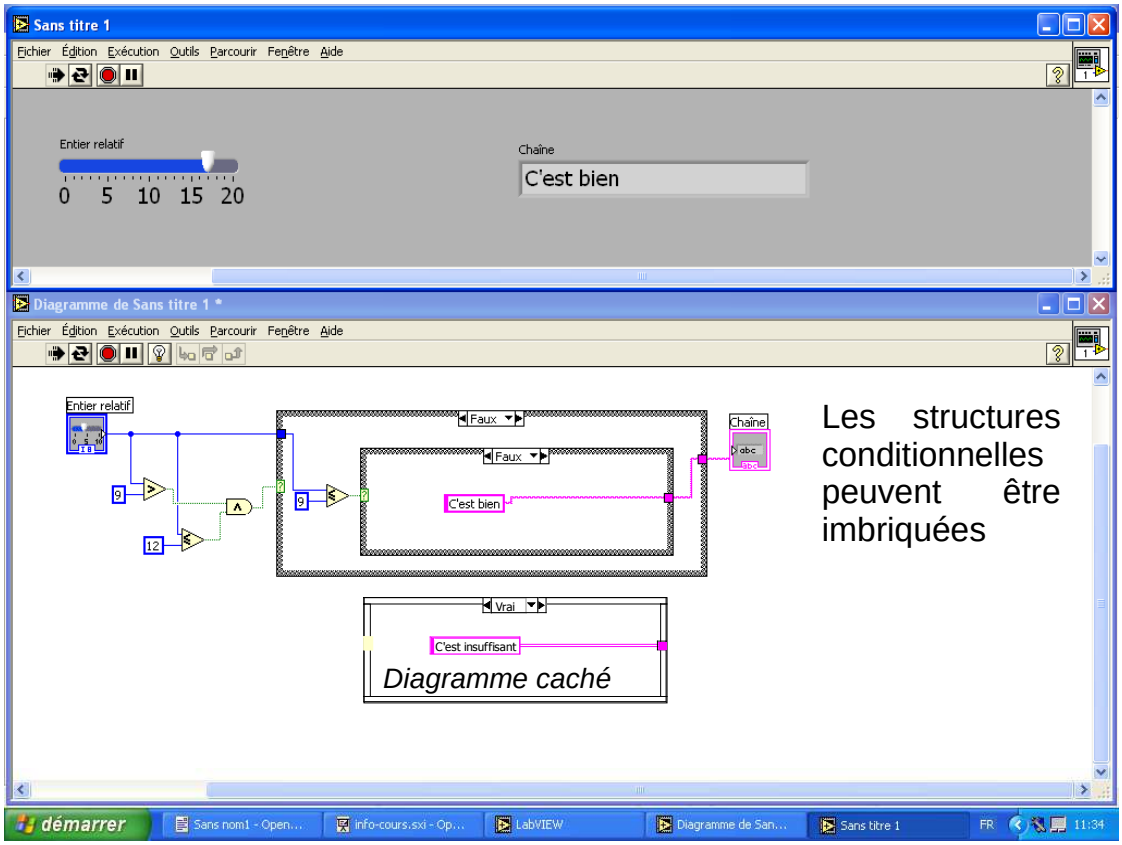
VI - REMARQUES





La comparaison peut être simple : elle met en oeuvre un opérateur de comparaison (>, <, etc ... ou d'égalité) comme dans l'exemple précédent.

Elle peut être plus complexe et mettre en oeuvre également des opérateurs logiques.



Les structures conditionnelles peuvent être imbriquées

Chapitre 4 – LA BOUCLE WHILE

I – DEFINITION

La boucle While permet de répéter une suite d'instructions en fonction du résultat d'un test.

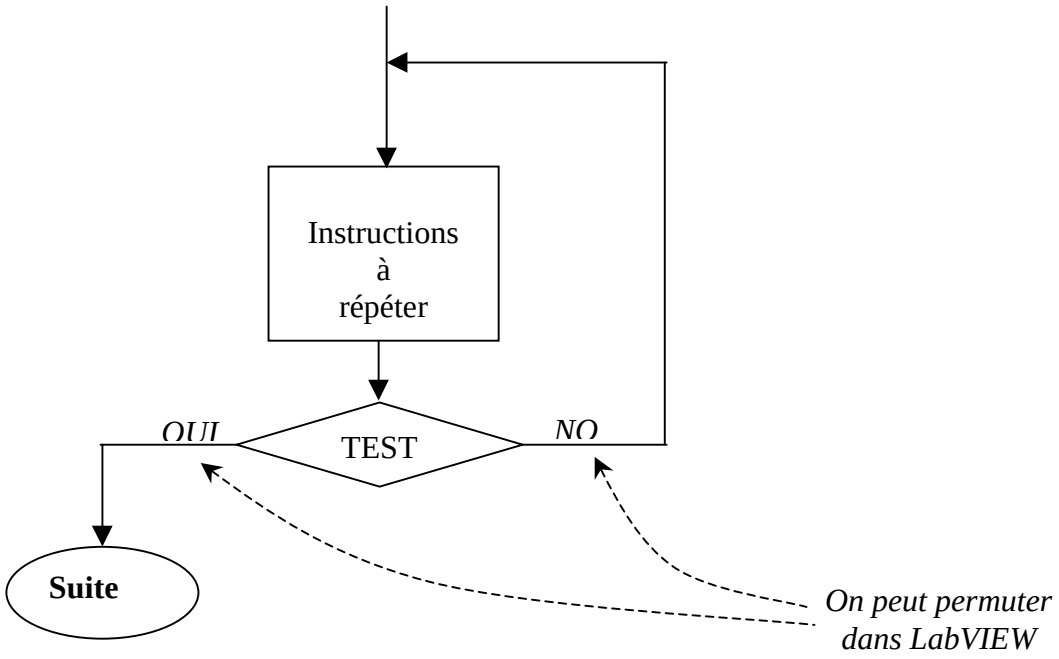
Un tour de boucle est appelé «itération".

Les instructions se répètent à chaque itération c'est à dire à chaque tour de boucle.

Dans LabVIEW, on peut régler la répétition en fonction du résultat VRAI ou FAUX du test.

Par défaut c'est le résultat FAUX qui permet la répétition.

II – ORGANIGRAMME



III– LE TEST

Comme pour la structure conditionnelle :

Le test est la condition permettant de réaliser la répétition. Il est évalué après chaque itération.

Le test met en œuvre des opérateurs de comparaison <, >, ≤, ≥, =, ≠.
Il peut être aussi l'évaluation d'une simple variable booléenne (test de l'appui sur un bouton poussoir)

Le résultat du test est un booléen : il ne peut prendre que 2 valeurs : OUI ou NON.

Le test peut être simple du type a < b ou plus complexe : (a < b) ET (a < c)

Les relations de comparaisons sont alors reliées entre elles par des opérateurs logiques : ET, OU, NON ET, NON OU, OU Exclusif, etc …

La relation correspondant au test est écrite dans le losange.

REMARQUES :

Le test est répété à la fin de chaque tour de boucle. Il est donc nécessairement placé à l'intérieur de la boucle.

La boucle réalise nécessairement au minimum un tour de boucle : Le bloc d'instructions à répéter est réalisé au moins une fois.

IV – EXEMPLE

1°/ LA BOUCLE D'ATTENTE

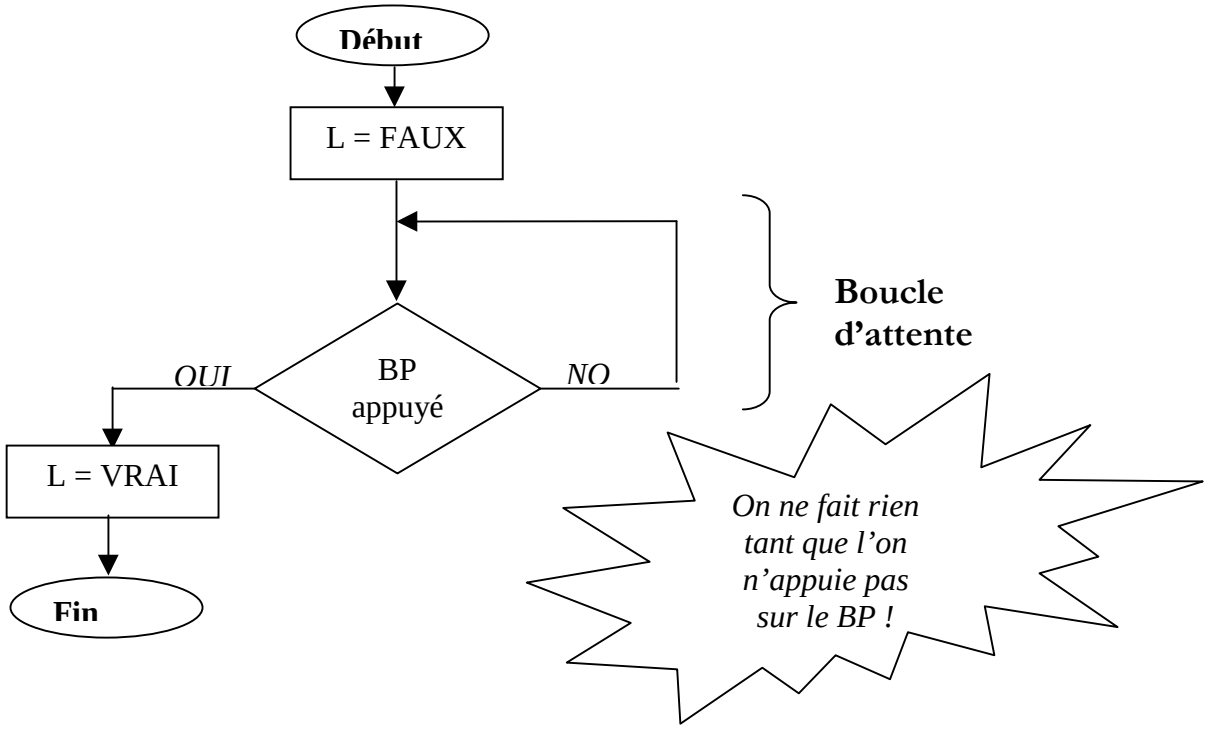
Attendre l'appui sur un bouton poussoir BP pour allumer une led L.

ENTREES SORTIES :

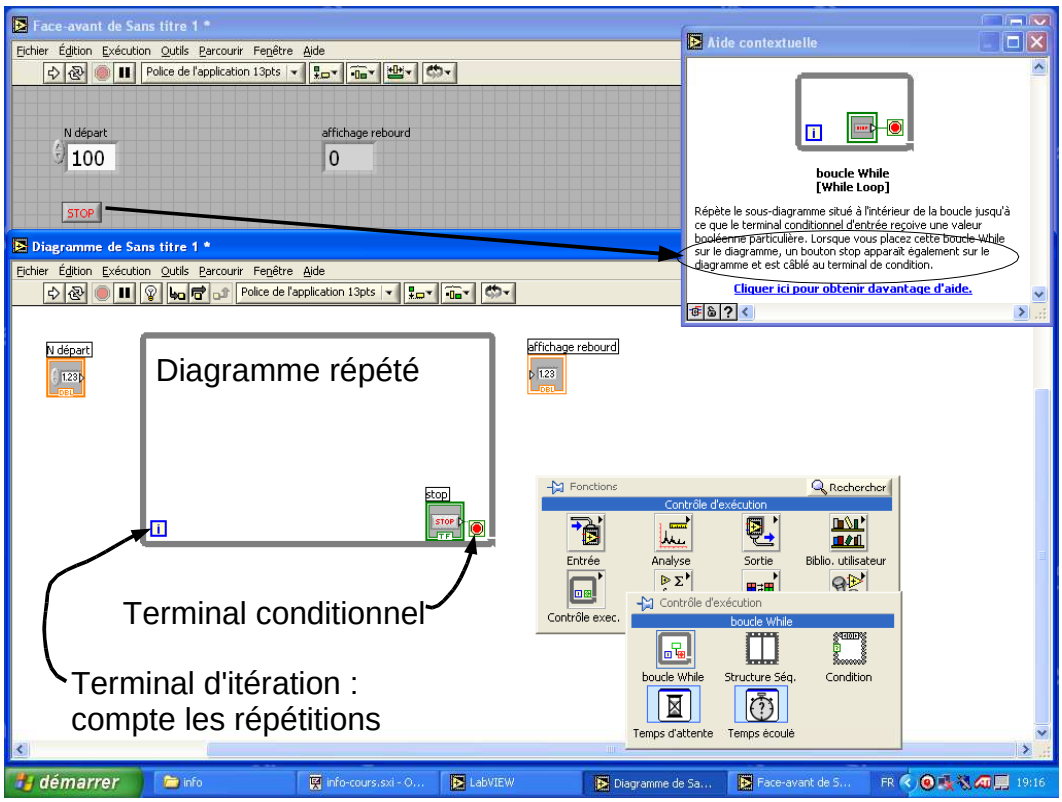
- Une Entrée booléenne : bouton poussoir nommé BP. Action mécanique : BP sans verrouillage.

- Une sortie booléenne : Led L.

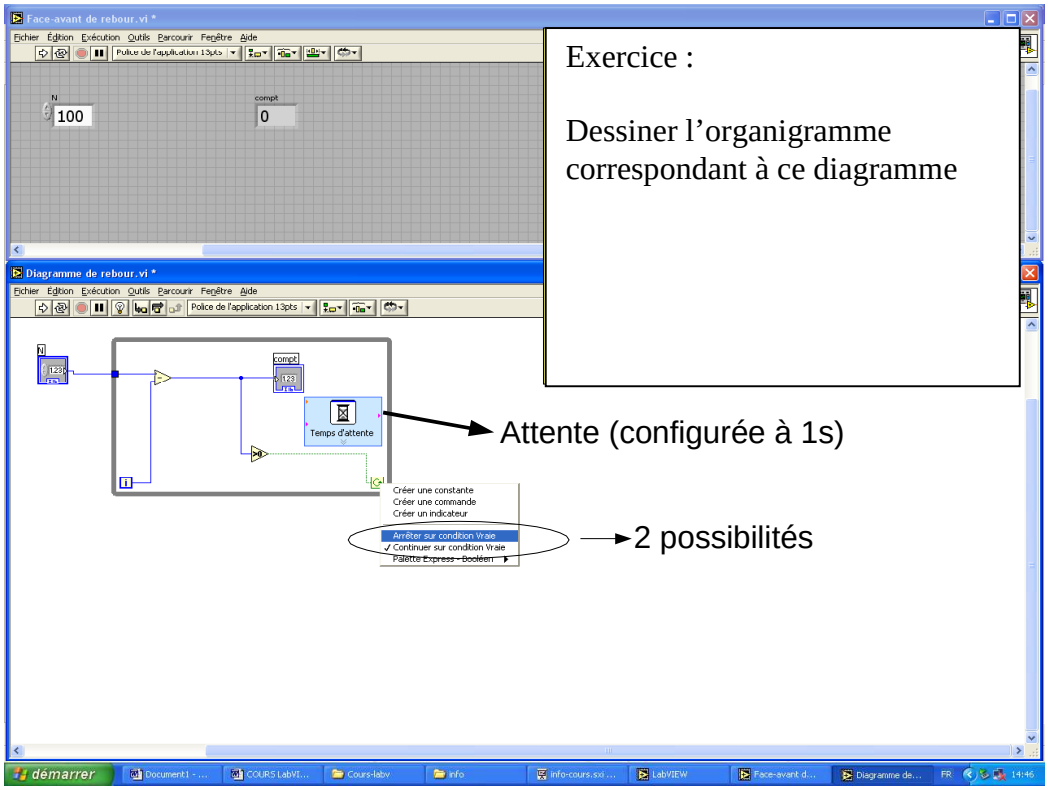
ORGANIGRAMME :



V – BOUCLE WHILE SOUS LabVIEW



Exemple : compte à rebours



Chapitre 5 - LA BOUCLE FOR

I – DEFINITION

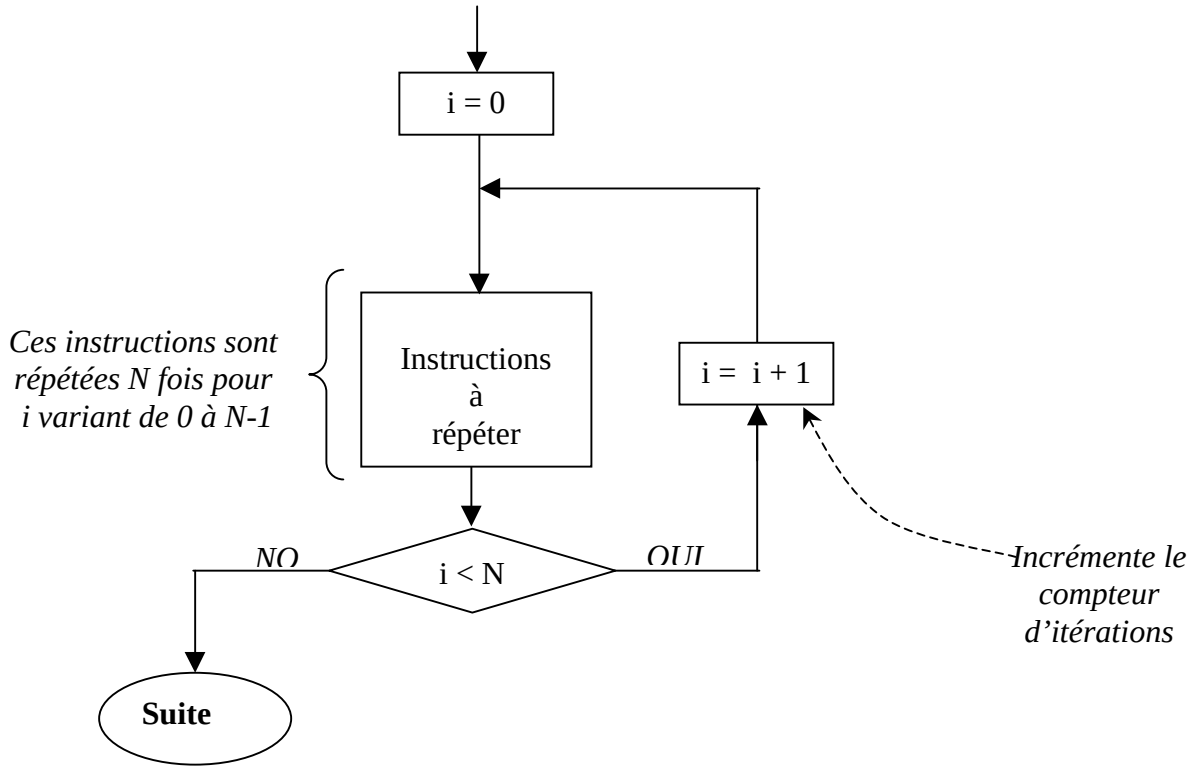
La boucle for permet de répéter une suite d'instructions **un nombre prédéterminé de fois.**

Avec la boucle While, on ne connaît pas à l'avance le nombre d'itérations qui seront réalisées.

Pour compter le nombre de répétitions, la boucle for utilise un compteur d'itérations. C'est une variable entière i variant de 0 à N-1 lorsque la boucle se répète N fois.

La boucle for utilise donc aussi une variable N entière donnant le nombre de tour de boucle.

II – ORGANIGRAMME



III - AFFECTATION

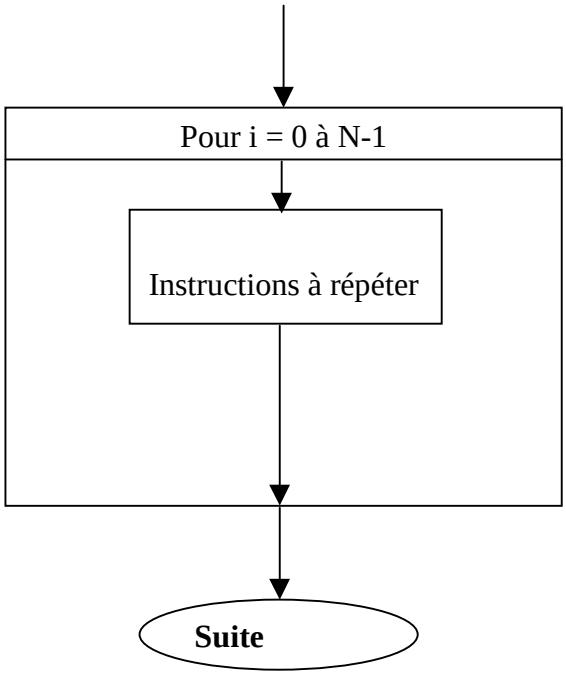
L'écriture i = 0 ne peut pas s'écrire 0 = i.
En effet il ne s'agit pas d'une égalité au sens mathématique.
Cette instruction est une affectation : On affecte la valeur 0 à la variable i.
Une affectation en informatique s'effectue toujours de la droite vers la gauche.

De même l'écriture i = i + 1 n'est pas non plus une égalité au sens mathématique.
En effet une telle écriture en mathématique conduit à 0 = 1 ! ! ! !

C'est une également une affectation : On prend la valeur de la variable i, on lui ajoute 1. La nouvelle valeur obtenue est alors redonnée à la même variable i.

IV – AUTRE ORGANIGRAMME POUR LA BOUCLE FOR

Pour simplifier, on peut représenter la boucle for ainsi :



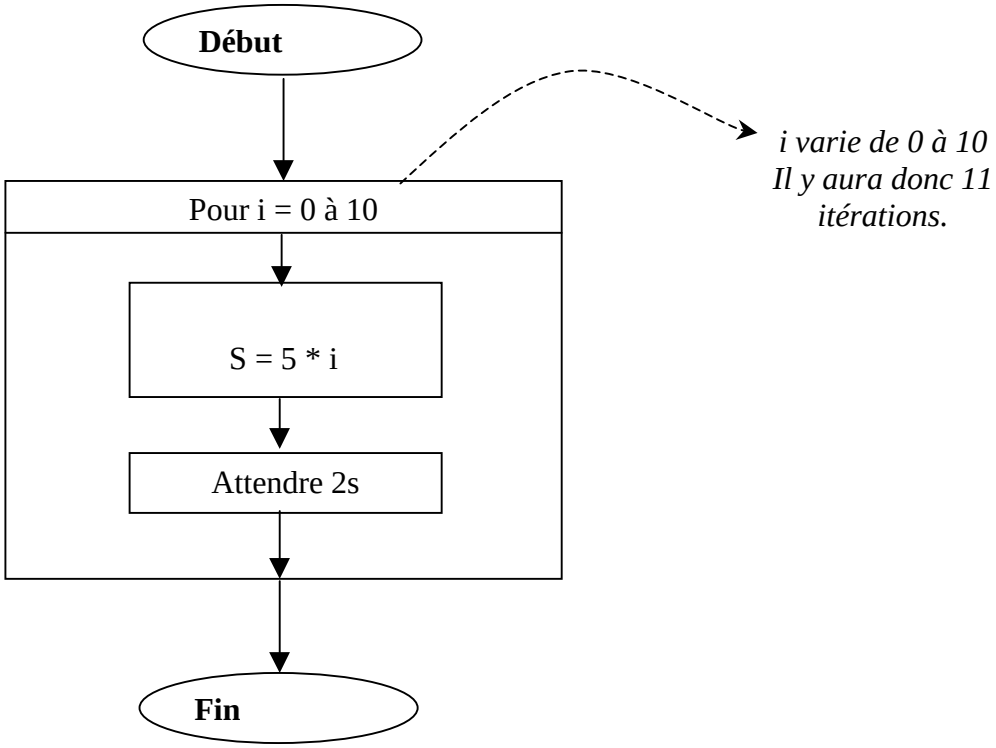
VI - EXEMPLE

Compter toutes les deux secondes de 5 en 5 jusqu'à 50, le résultat étant affiché sur un afficheur numérique.

ENTREES SORTIES :

- Pas d'entrée
- Une sortie S, afficheur numérique.

ORGANIGRAMME



VII – BOUCLE FOR EN LabVIEW

Clic droit dans le diagramme, puis étendre la palette. Aller sur programmation puis structures.

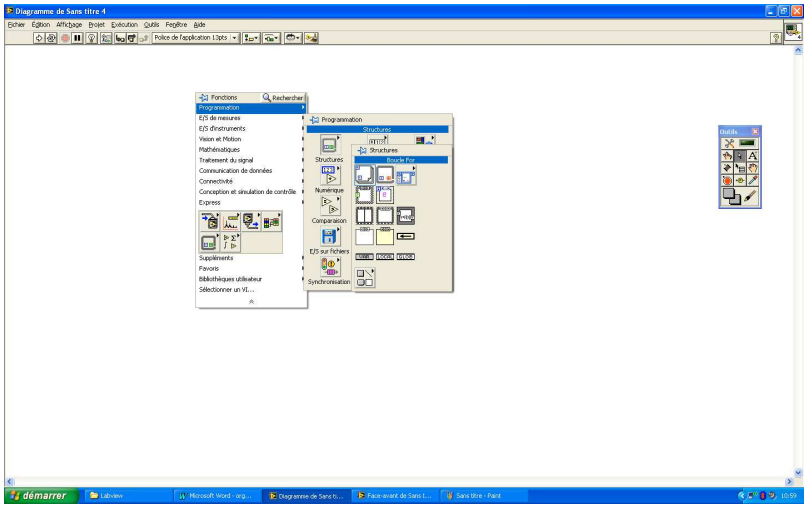
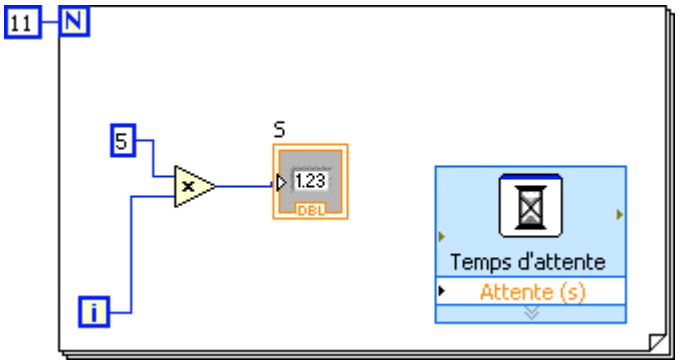
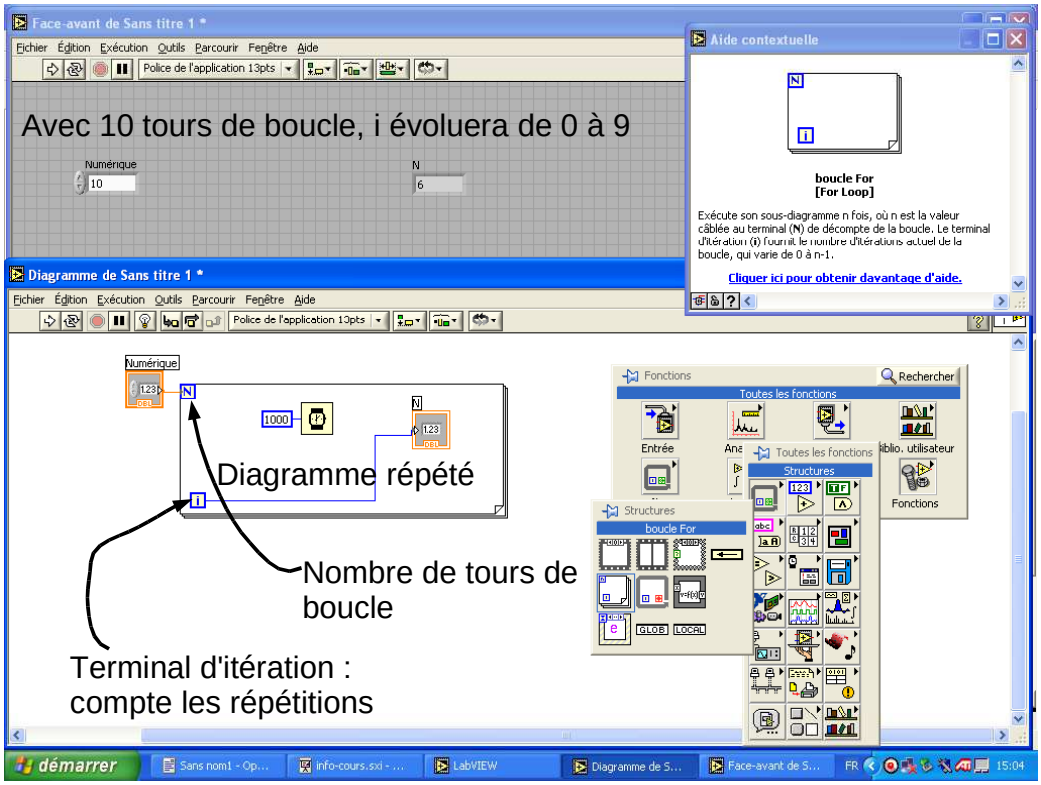


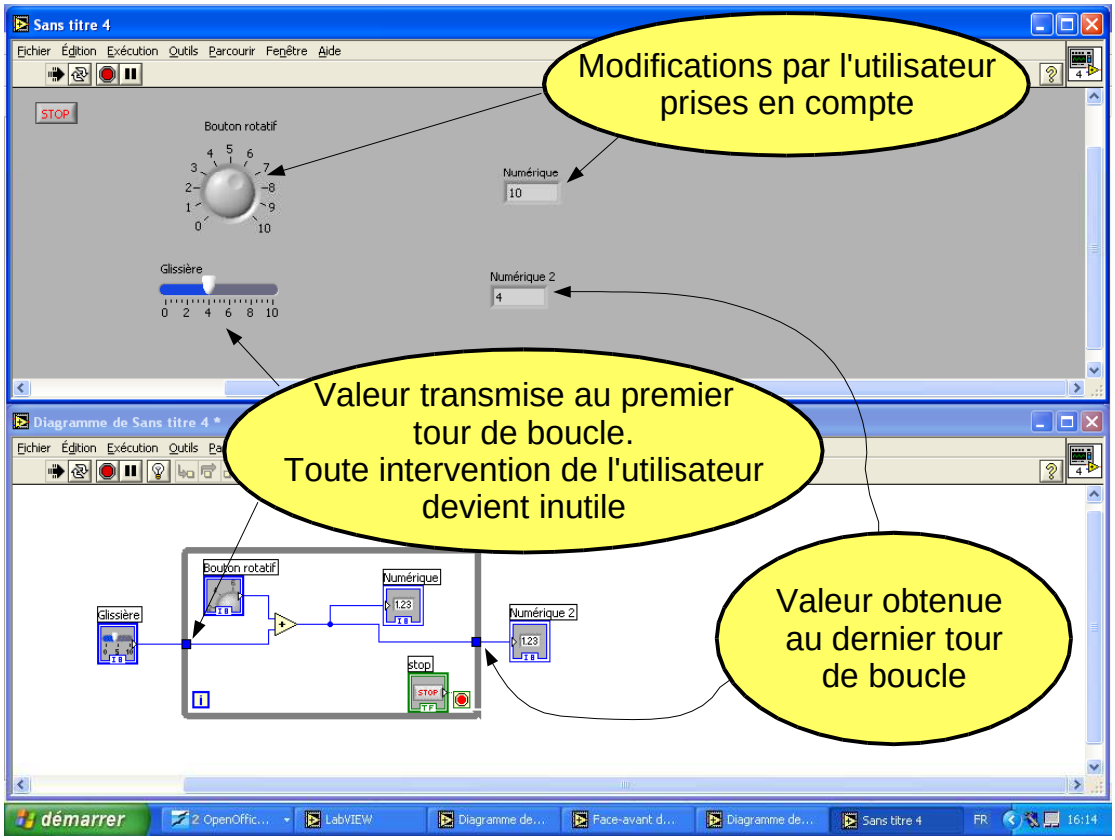
Diagramme :



REMARQUE :



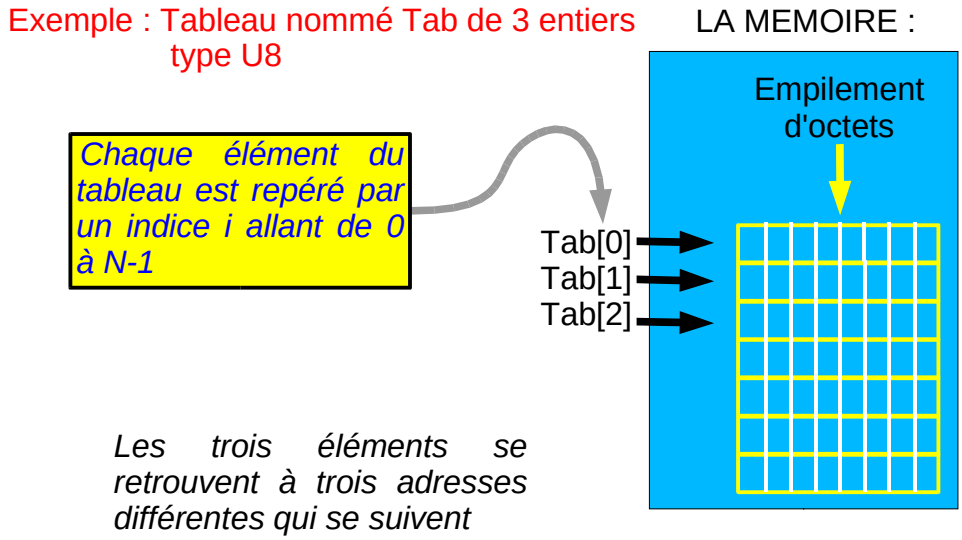
VIII – COMMANDES ET INDICATEUR DANS OU HORS DE LA BOUCLE



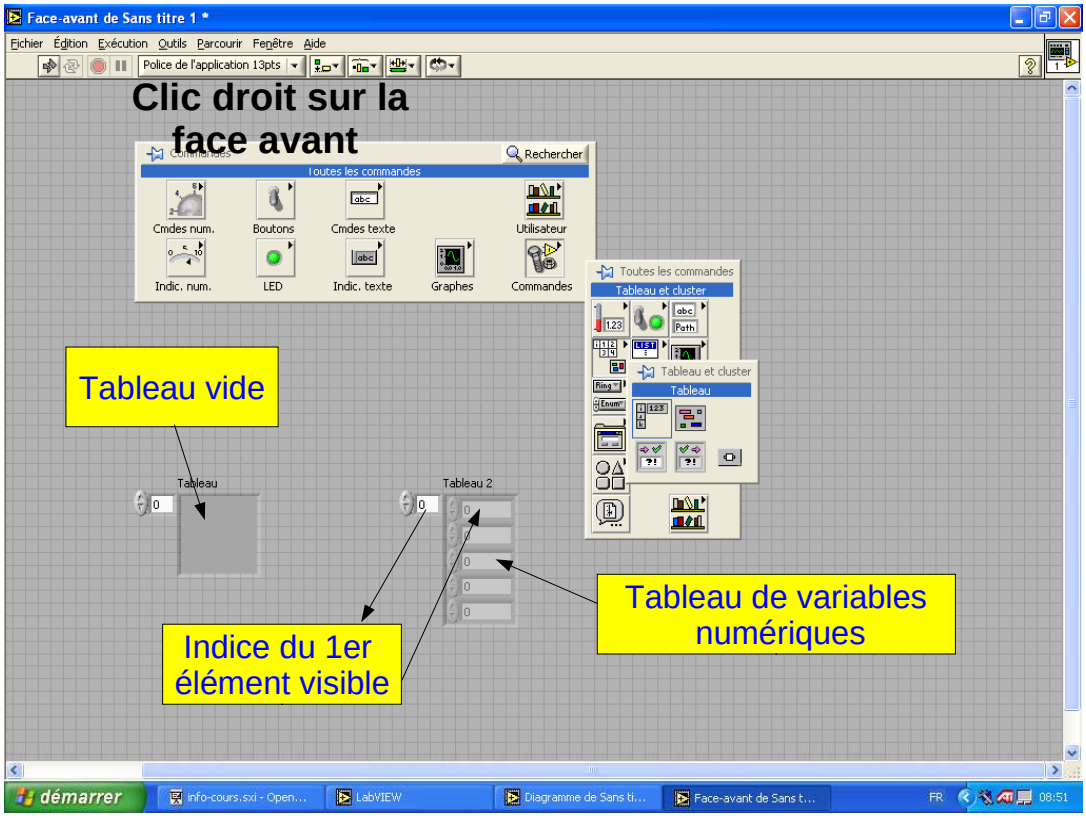
Chapitre 6 – LES TABLEAUX

I – DEFINITION

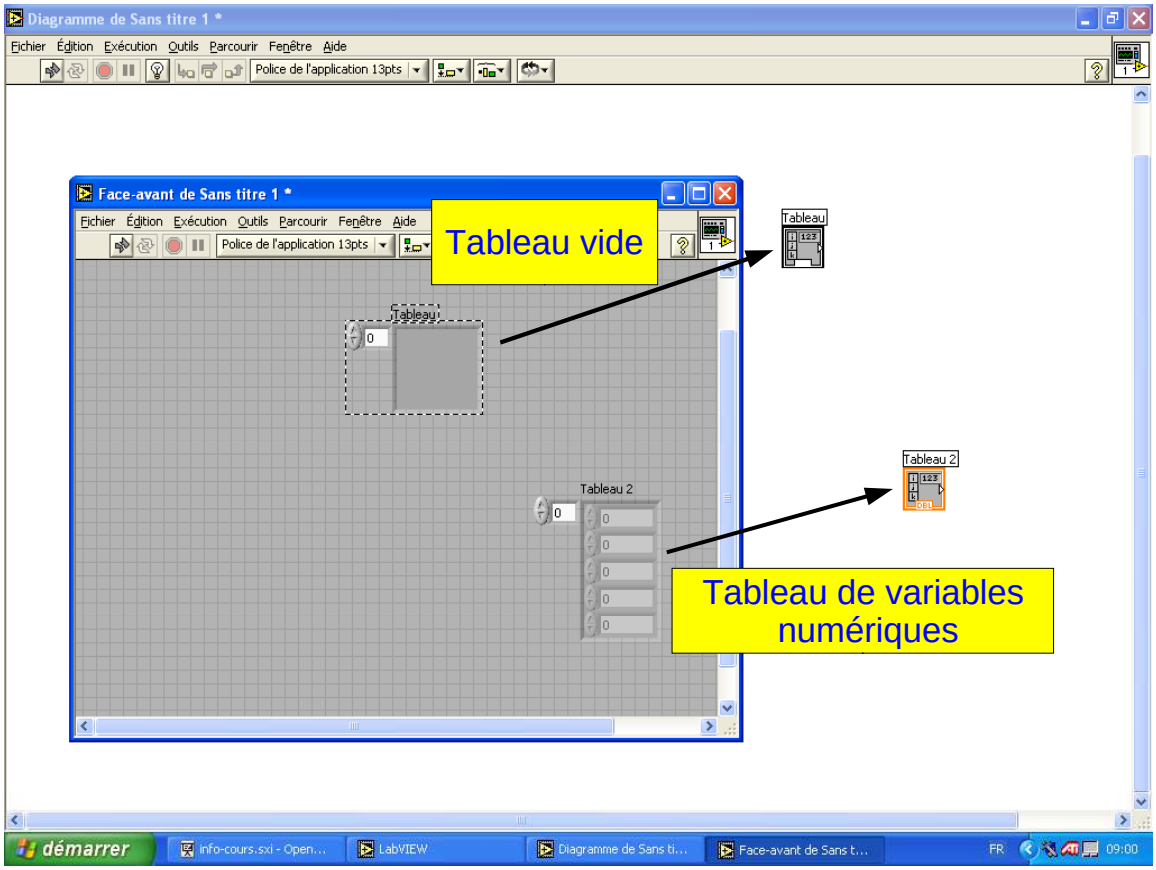
Un tableau permet de regrouper plusieurs variables numériques de même type sous une même appellation.
Les différents éléments du tableau (array) sont alors repérés par un indice (index).
Le nombre N d'éléments du tableau constitue sa taille (array size)



II – TABLEAUX DANS LabVIEW

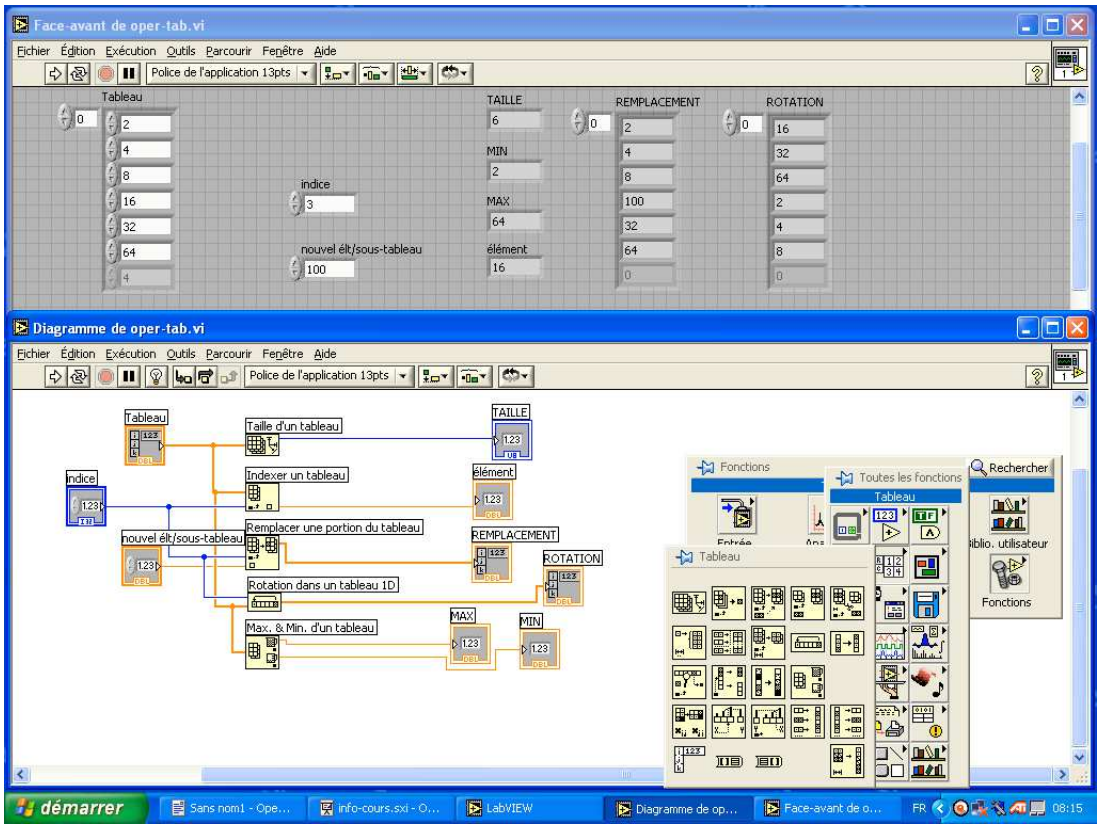


Quand on dépose le tableau sur la face avant, il est vide.
Il convient donc de le remplir à l'aide du type de variable choisi :
tableau de variables numériques,
tableau de booléens,
tableaux de chaînes de caractères
etc…



III – OPERATIONS SUR LES TABLEAUX

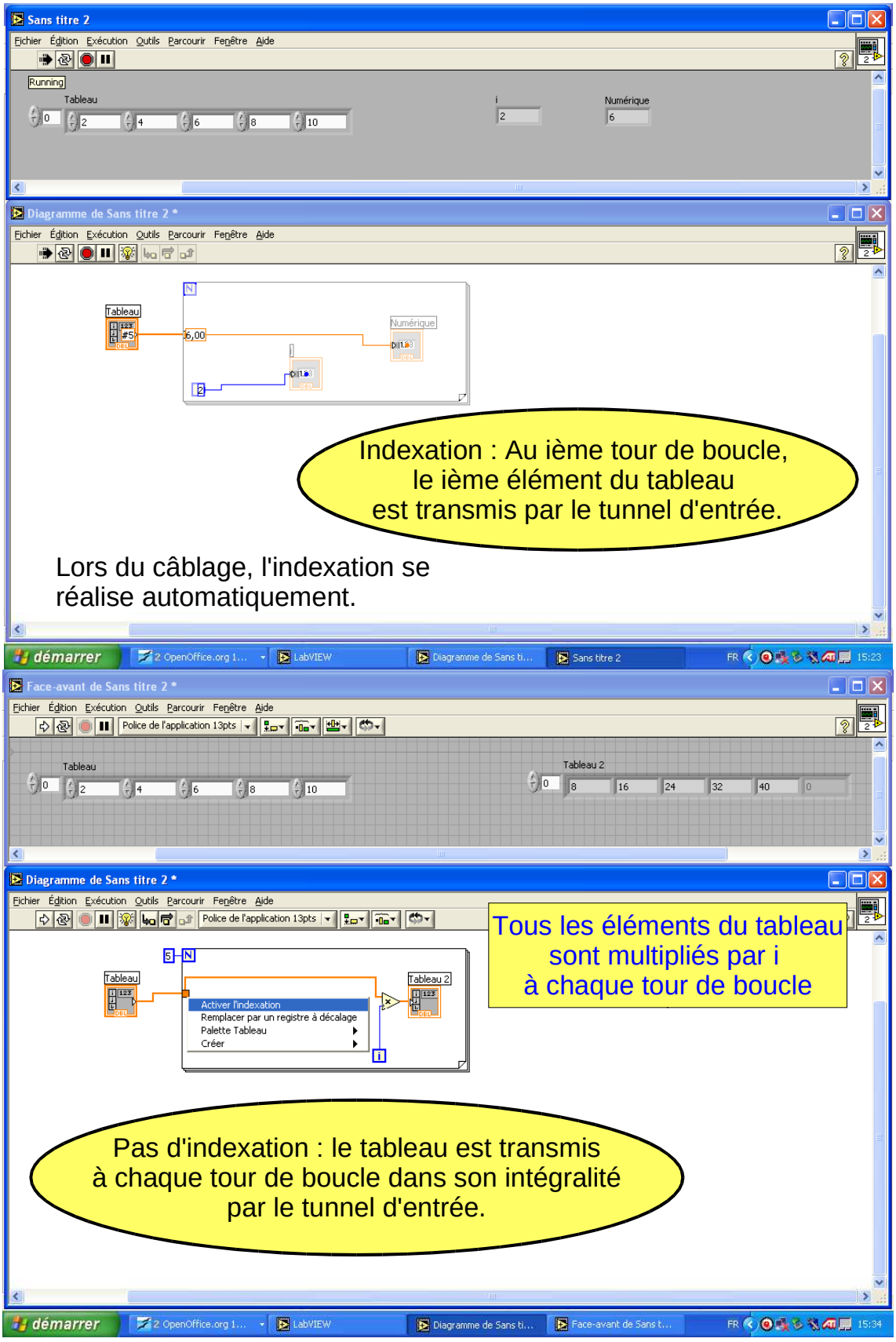
En plus des opérations arithmétiques classiques, il existe des fonctions spéciales :



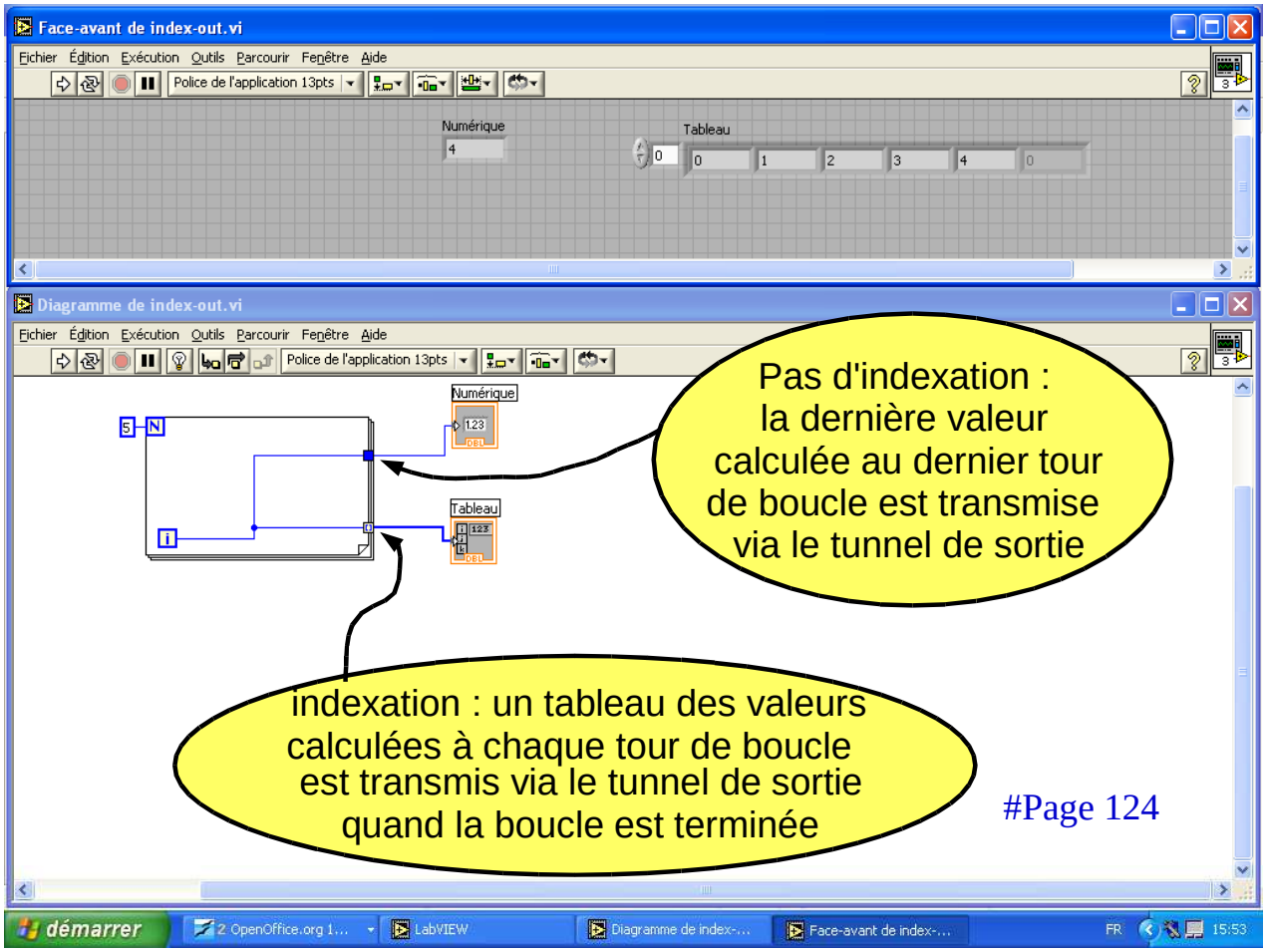
Les fils de liaisons des données tableaux sont en traits épais de la couleur du type de la variable : ici, on a des tableaux de variables numériques à virgule flottante, les connexions sont donc de couleur orange. Indice et taille du tableau sont des entiers donc couleur bleue.

IV – INDEXATION

1°/ En entrée



2°/ En sortie

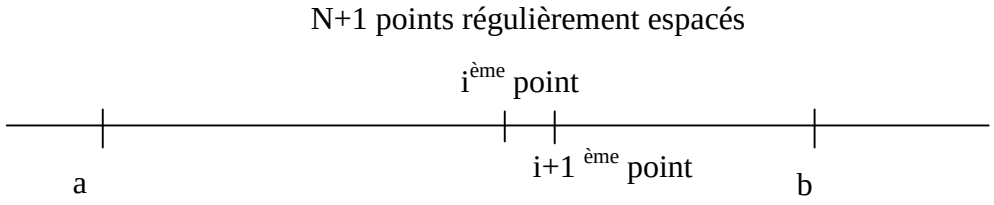


#Page 124

Chapitre 7 - LES GRAPHES

I – LES ECHELLES

1°/ ECHELLE LINEAIRE



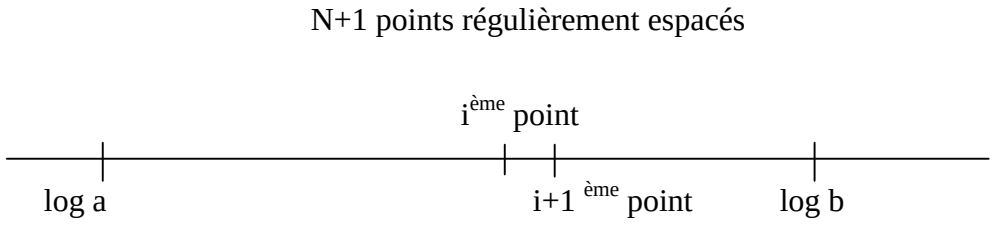
Pas : distance entre deux points. On a : **pas = (b – a) / N**

Passage d'un point à l'autre : $x_i = x_{i-1} + pas$

Ainsi : $x_i = a + i \,.\, pas$

On a une suite arithmétique de premier terme $x_0 = a$ et de raison q = pas

2°/ ECHELLE LOGARITHMIQUE



Pas : distance entre deux points. On a : $pas = (\log b - \log a) / N = \log (b/a)^{1/N}$

Passage d'un point à l'autre : $\log x_i = \log x_{i-1} + pas$

Ainsi : $\log x_i = \log a + i \,.\, pas = \log a + i \log (b/a)^{1/N} = \log a + \log (b/a)^{i/N}$

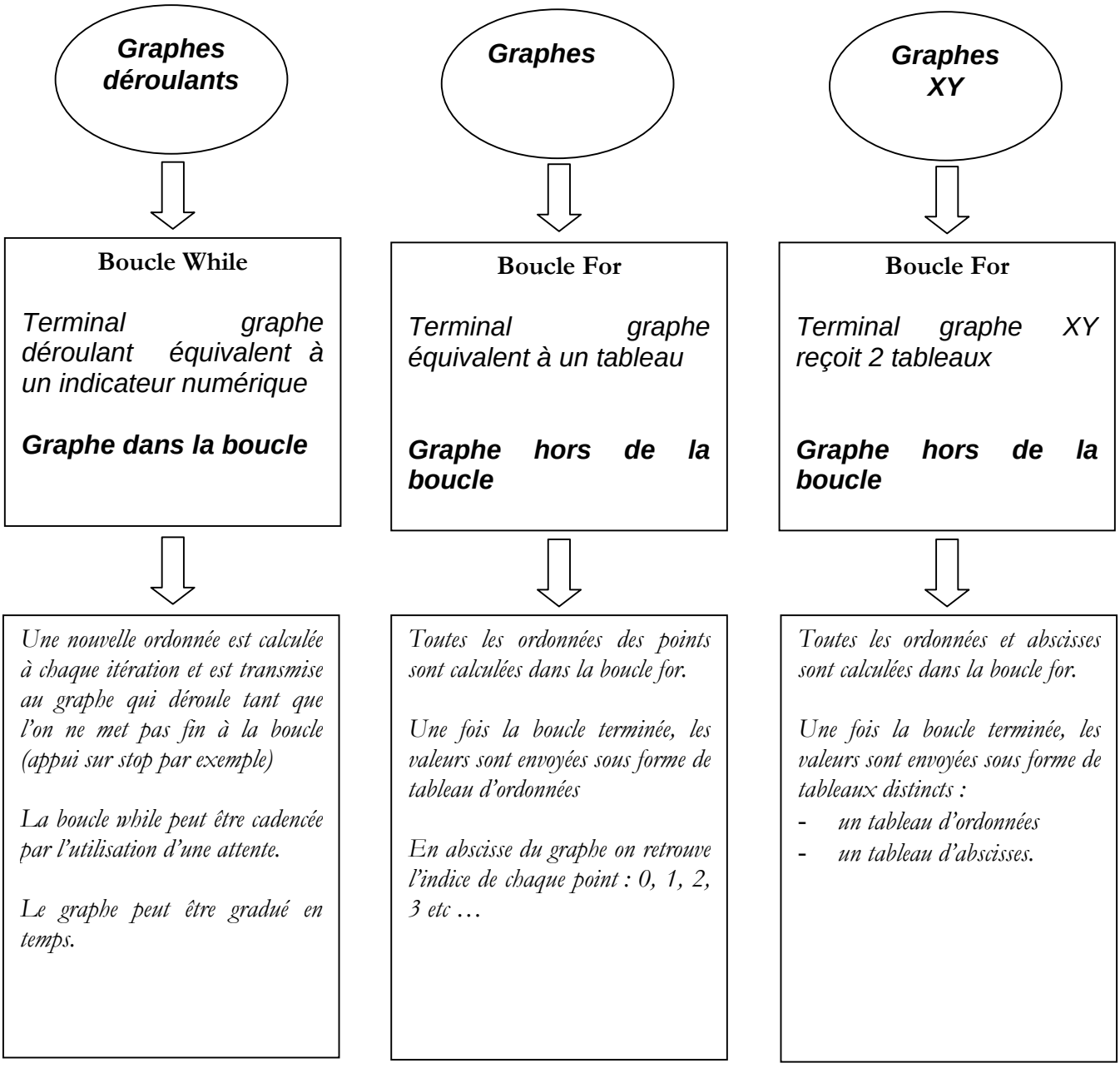
$= \log [\, a \,.\, (b/a)^{i/N} \,]$

Donc : $x_i = a \,.\, (b/a)^{i/N} \iff x_i = a \,.\, (b/a)^{i-1/N} \,.\, (b/a)^{1/N}$

$$x_i = x_{i-1} \,.\, (b/a)^{1/N} \text{ ou } x_i = a \,.\, [(b/a)^{1/N}]^{\,i}$$

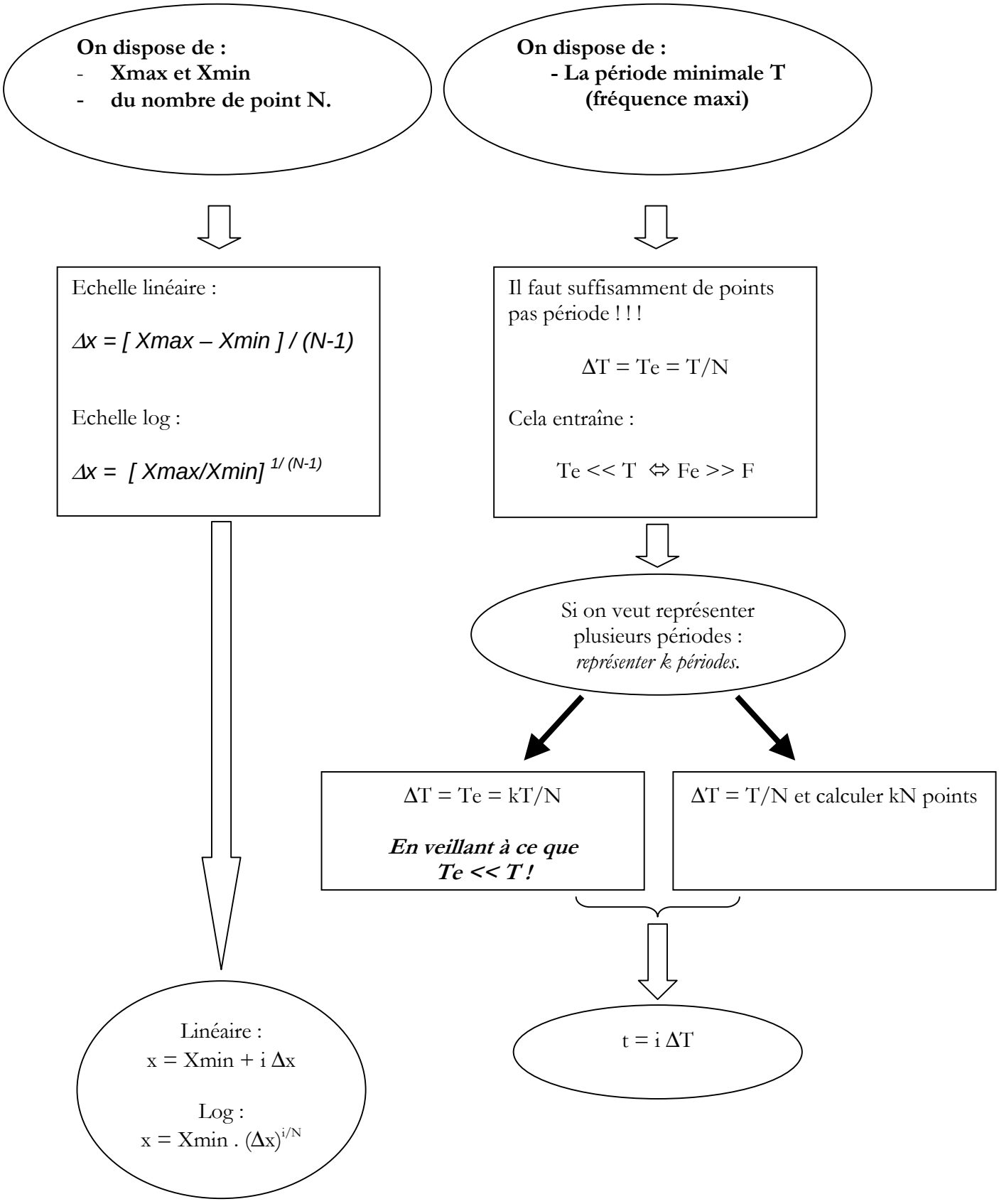
On a une suite géométrique de premier terme $x_0 = a$ et de raison $q = (b/a)^{1/N}$

II – LES DIFFERENCES



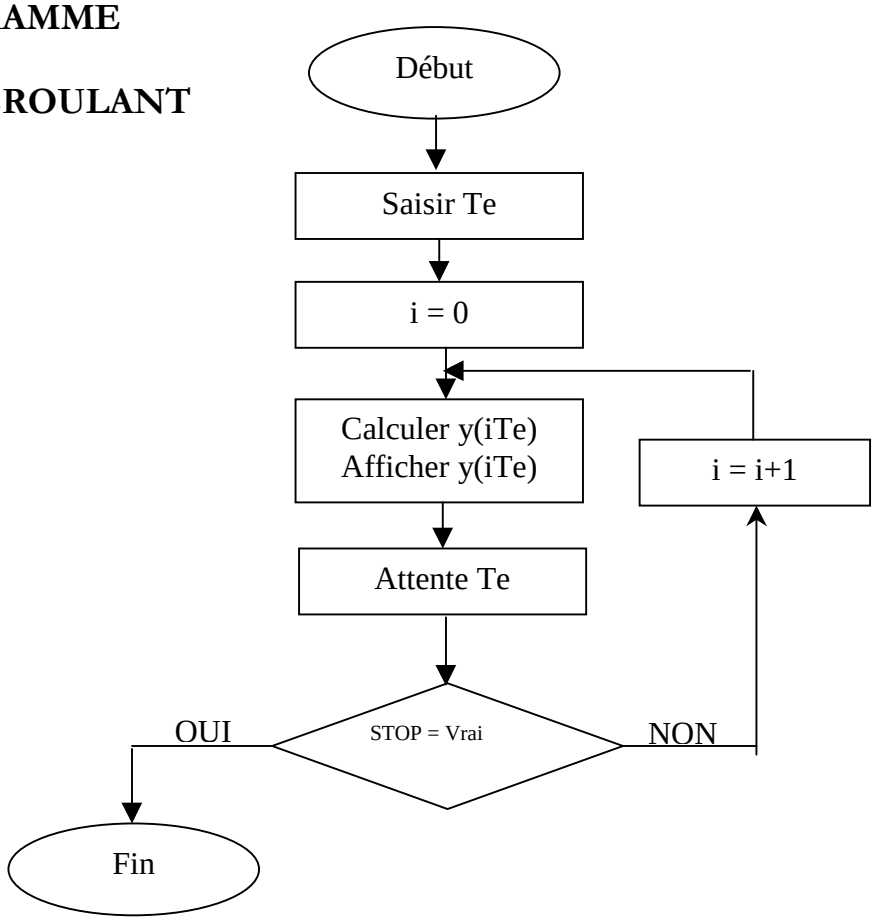
III - PERIODE D'ECHANTILLONNAGE

Quelle que soit la représentation graphique, il faut définir un pas entre deux points de calcul ou d'acquisition. Dans le cas d'une représentation temporelle, on parle de période d'échantillonnage.

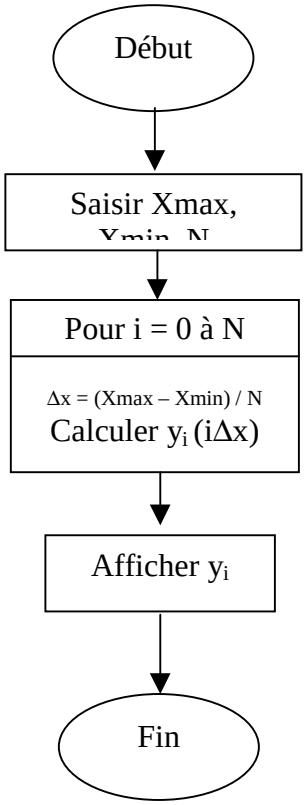


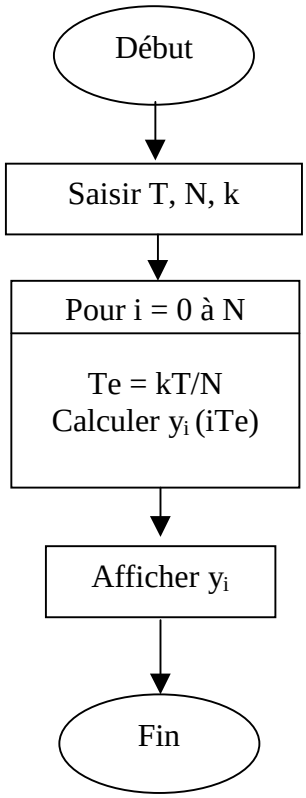
IV – ORGANIGRAMME

1°/ GRAPHE DEROULANT

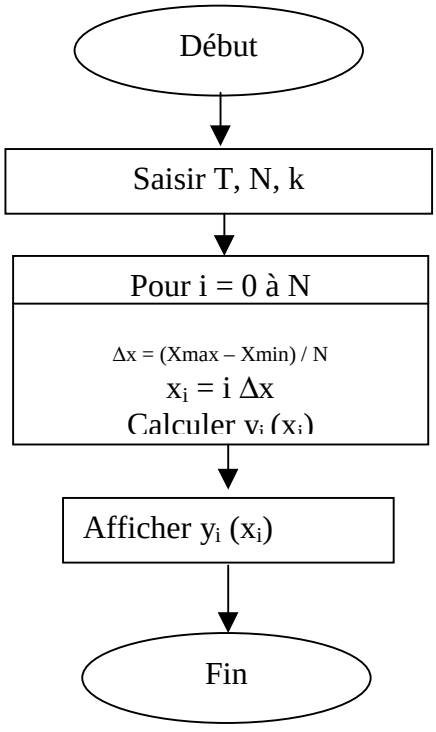


2°/ GRAPHE





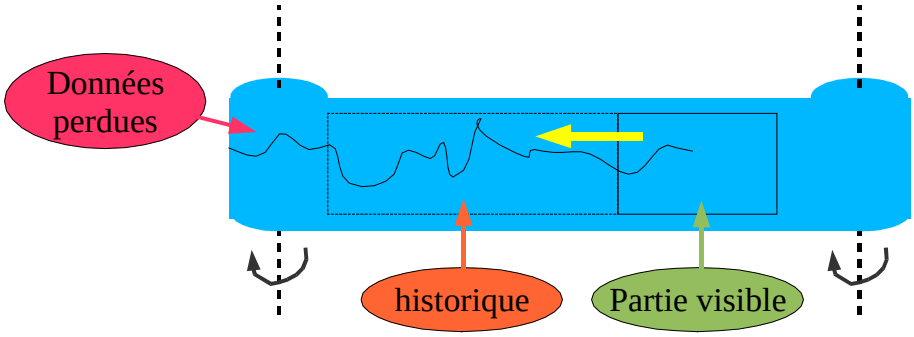
3°/ GRAPHE XY



Chapitre 8 – GRAPHES DANS LABVIEW

I – GRAPHE DEROULANT : Waveform Chart

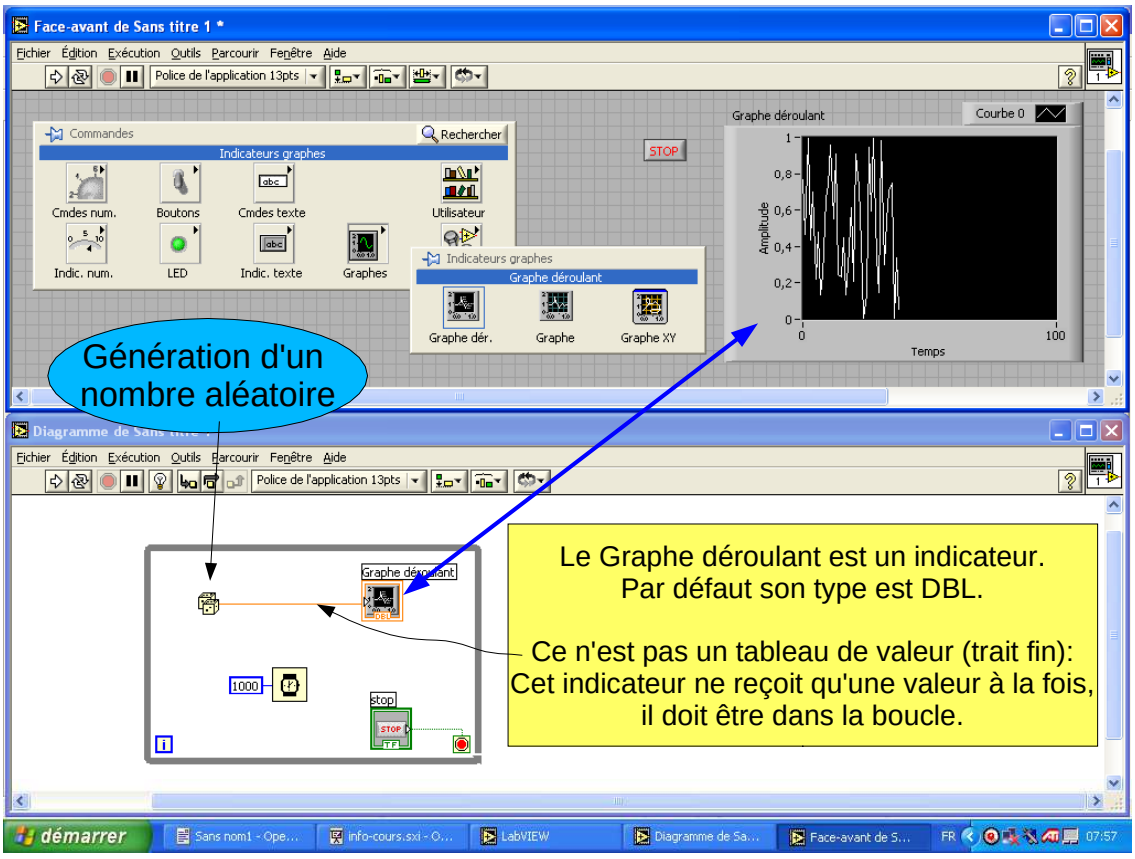
Le rôle du graphe déroulant : Waveform Chart est d'afficher l'évolution temporelle d'une donnée variable dans le temps.



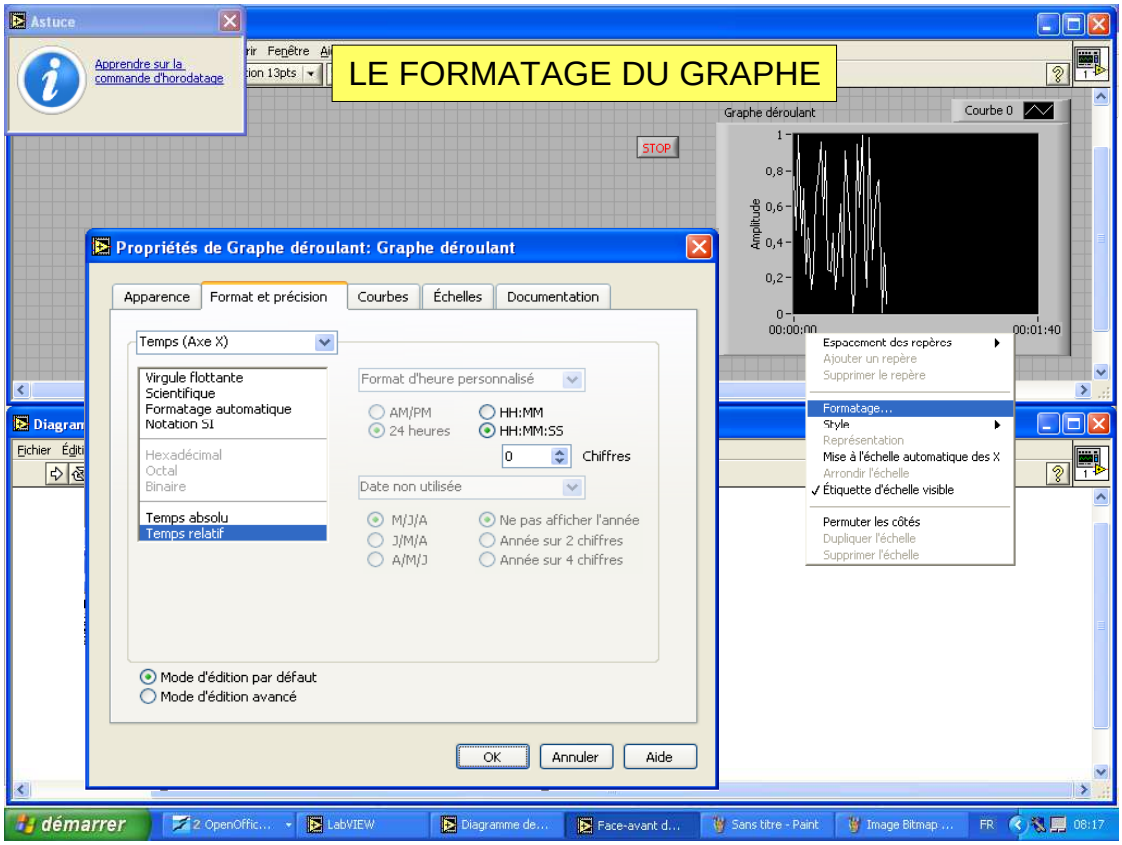
A chaque instant le graphe ne reçoit qu'une donnée à la fois :
Ce n'est pas un tableau de valeurs.

Par programmation, il faut donc répéter l'envoi des données au graphe, le terminal graphe doit donc être dans une boucle while.

Exemple dans LabVIEW :



Certains paramètres du graphe peuvent être ajustés : clic droit sur l'objet graphe dans la face avant.



II – GRAPHES : Waveform Graph

Le rôle du graphe est de représenter graphiquement un tableau de données.

	donnée
0	5
1	10
2	15
3	20
4	25
5	30
6	35
7	40

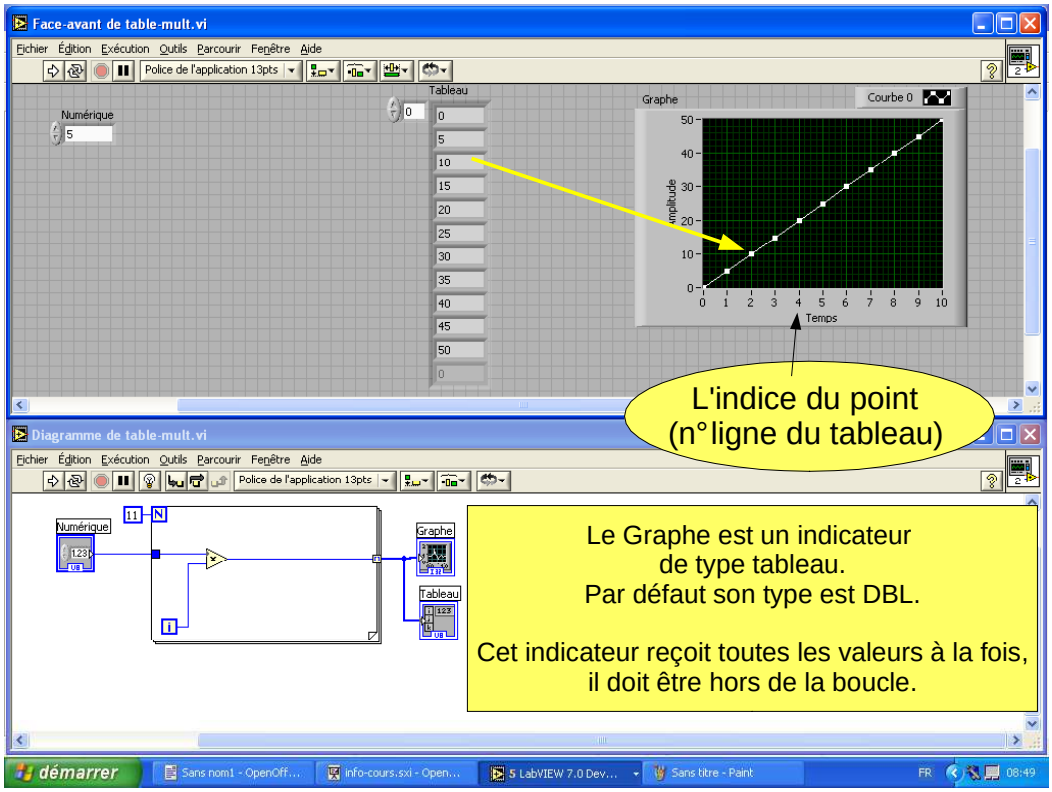
Le graphe reçoit simultanément
l'ensemble des données
sous forme d'un tableau de valeurs.

Les données sont généralement
calculées dans une boucle

Par programmation, il ne faut envoyer
les données au graphe qu'une seule fois,
le terminal graphe doit donc être hors de la boucle.

Le tableau peut être obtenu par indexation
du tunnel de sortie de la boucle

Exemple dans LabVIEW :



III – GRAPHE XY

Le rôle du graphe XY est de représenter graphiquement un tableau de données Y en fonction d'un tableau de données X

Donnée X	Donnée Y
0,1	5
0,2	10
0,3	15
0,4	20
0,5	25
0,6	30
0,7	35
0,8	40

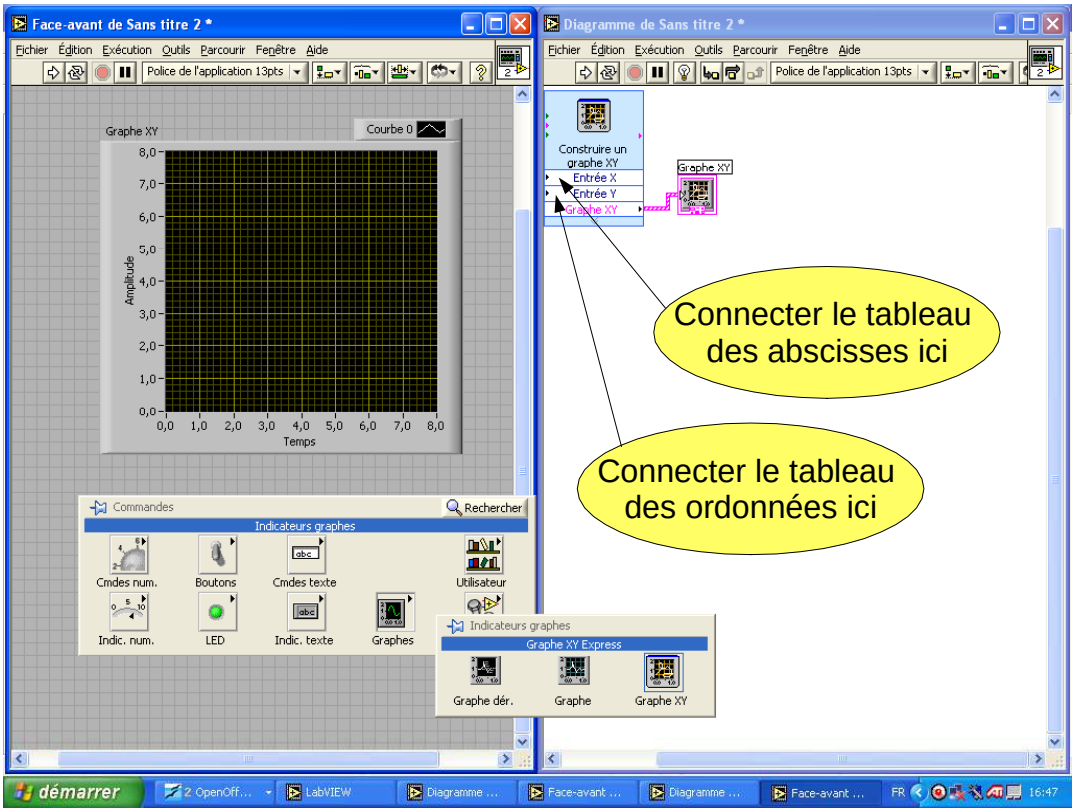
Le graphe reçoit simultanément
l'ensemble des données
sous forme de 2 tableaux de valeurs.

Les données sont généralement
calculées dans une boucle

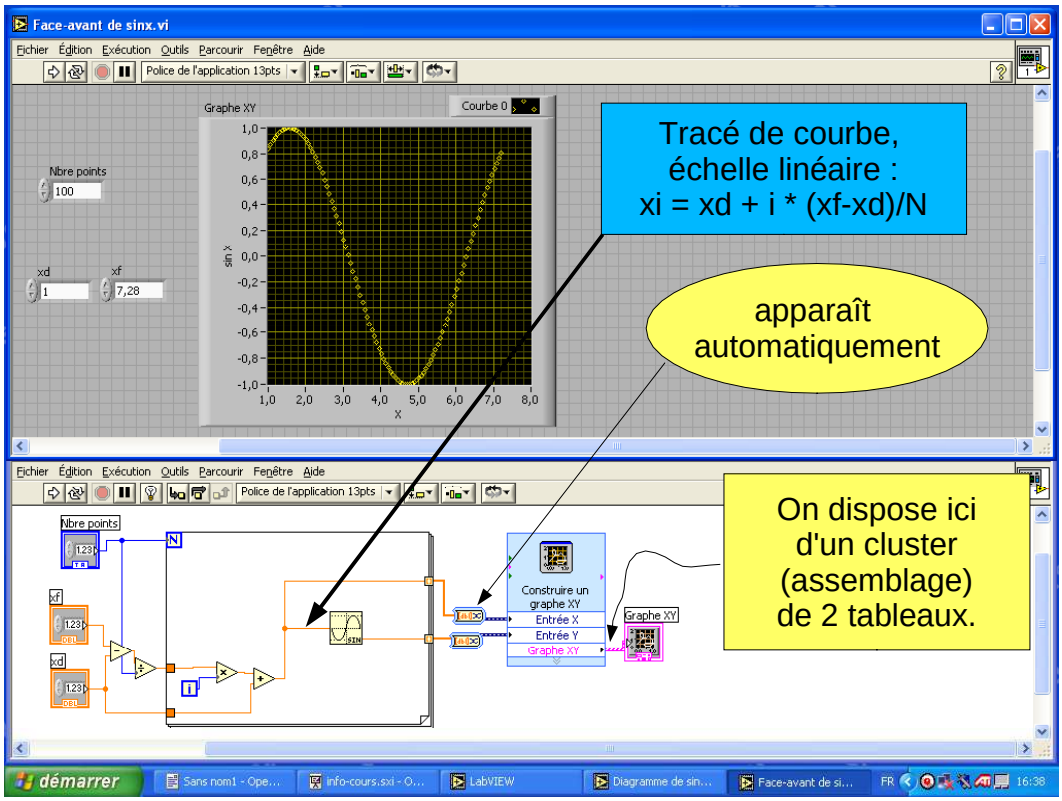
Par programmation, il ne faut envoyer
les données au graphe qu'une seule fois,
le terminal graphe XY doit donc être hors de la boucle.

Les 2 tableaux sont obtenus directement
au niveau des 2 tunnels de sortie de la boucle.
Ils sont assemblés pour former un cluster.

Utilisation dans LabVIEW :



Exemple dans LabVIEW :

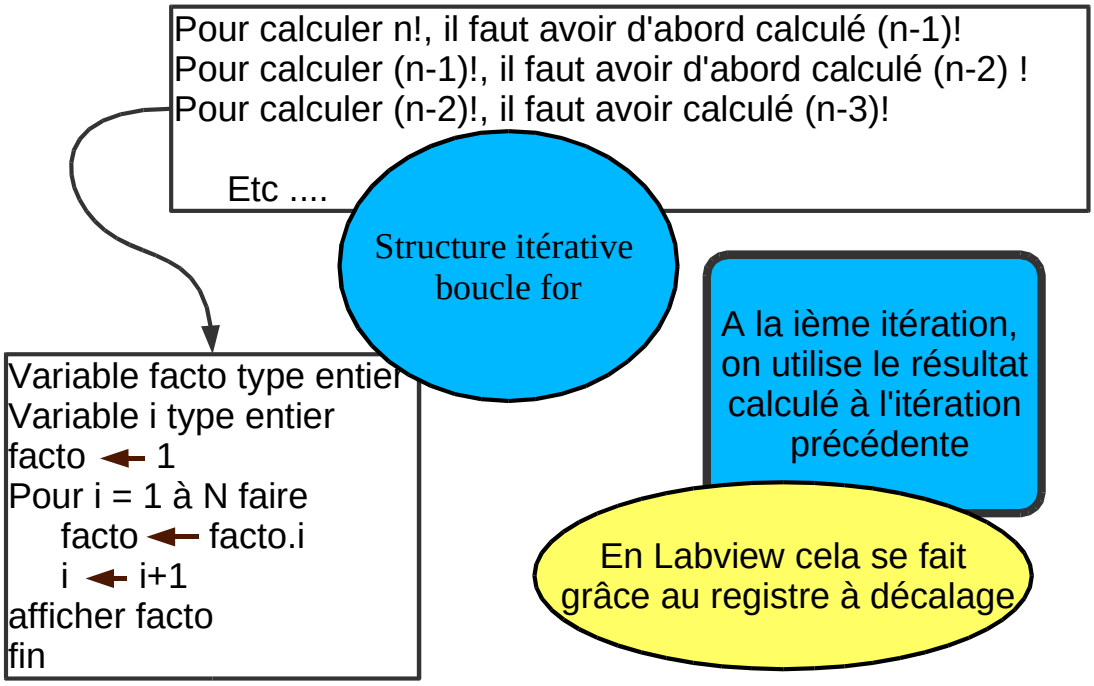


Chapitre 9 – LES REGISTRES A DECALAGE OU NŒUDS DE RETROACTION

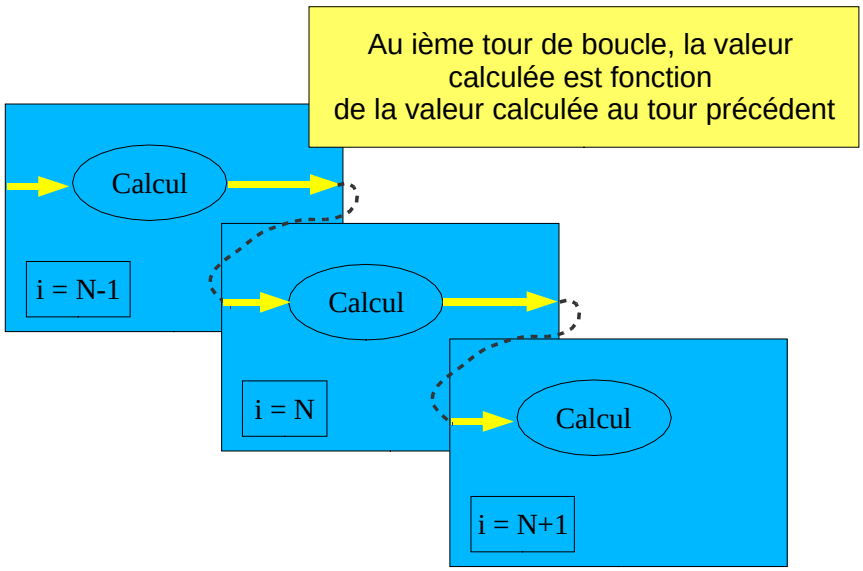
I – EXEMPLE D'UTILISATION

EXEMPLE : Programmer n!

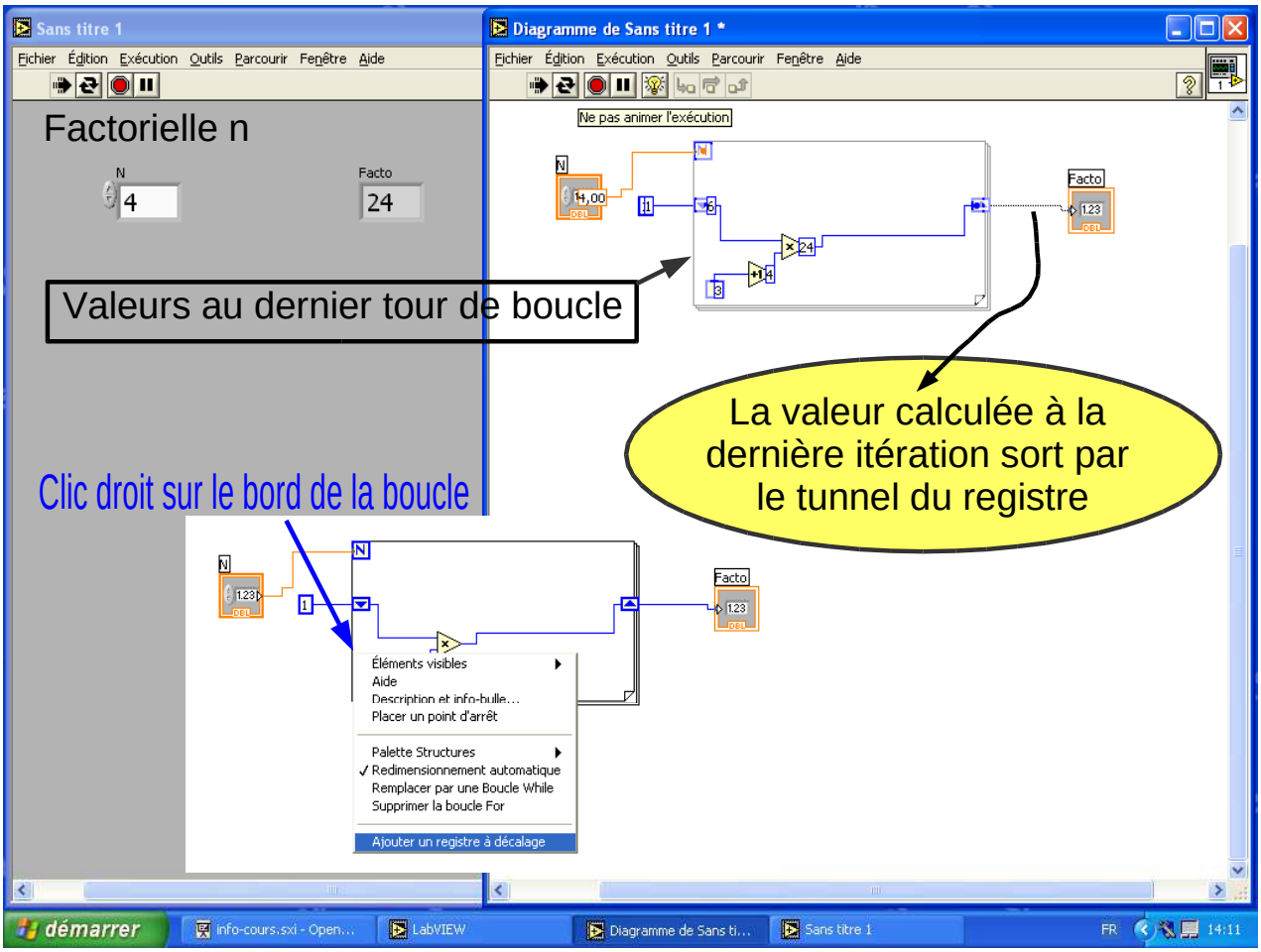
On sait que n! = (n-1)! . n



II – REGISTRE A DECALAGE DANS LabVIEW



Exemple : factorielle n avec registre à décalage :



III – DECALAGES MULTIPLES

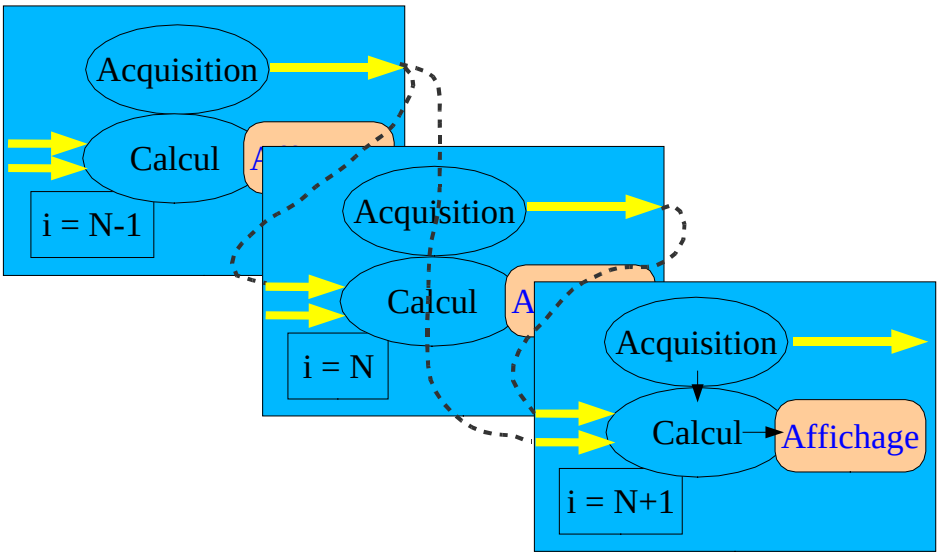
1°/ EXEMPLE

Exemple : Calculer et mettre à jour la moyenne M des 3 dernières mesures m_i

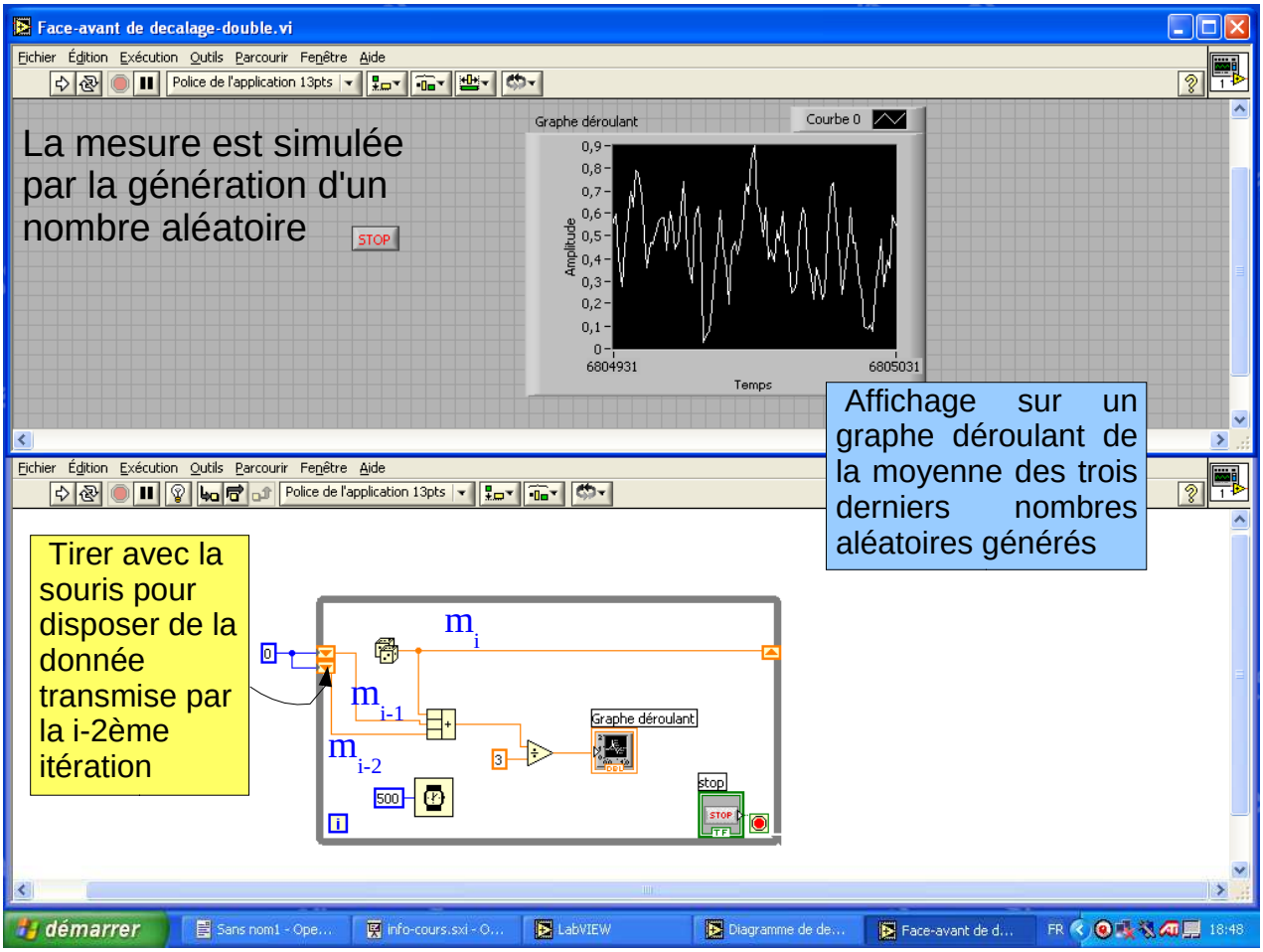
$$\text{On a } M_i = (m_i + m_{i-1} + m_{i-2}) / 3$$

On répète la mesure périodiquement => boucle while

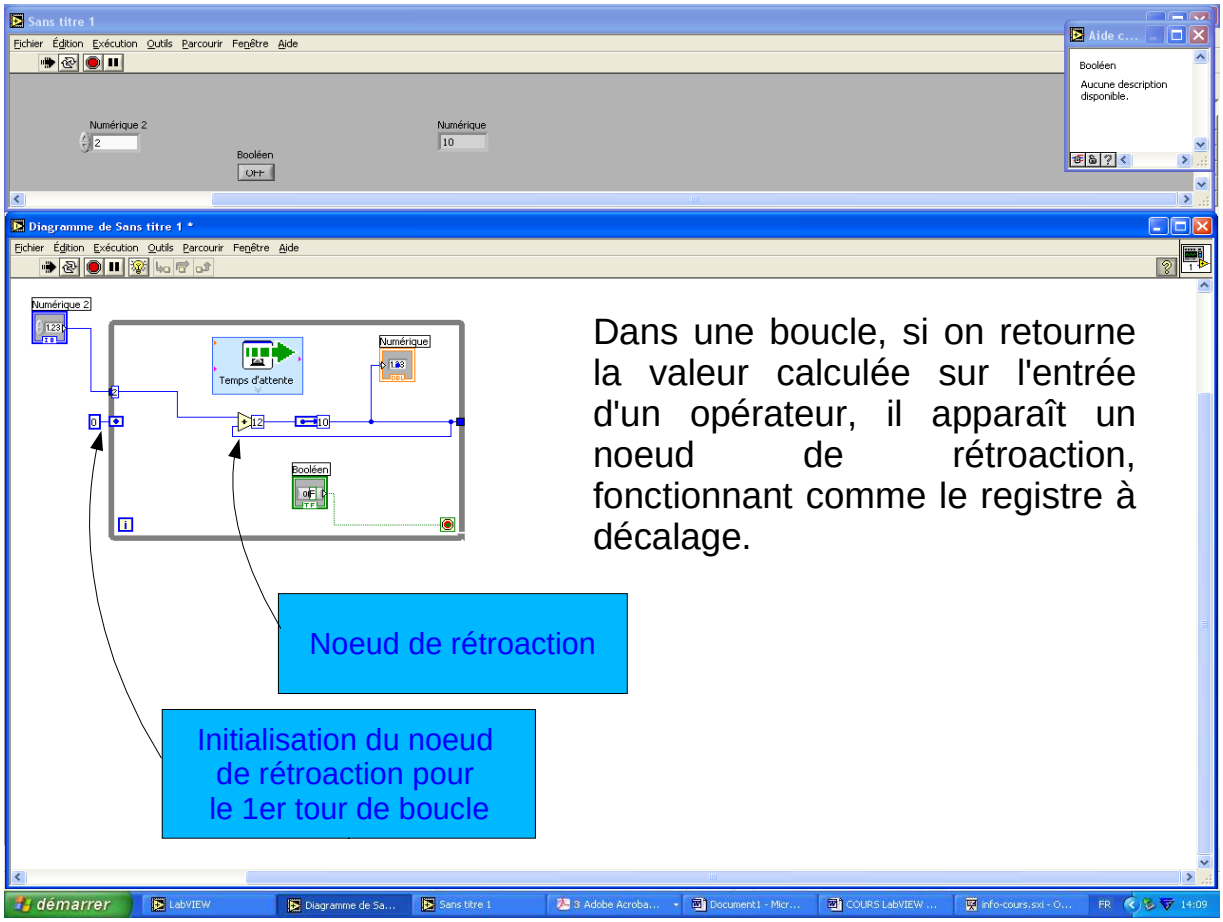
Pour calculer Mi, on a besoin des valeurs mesurées dans les deux itérations précédentes.



Ce qui donne dans LabVIEW :



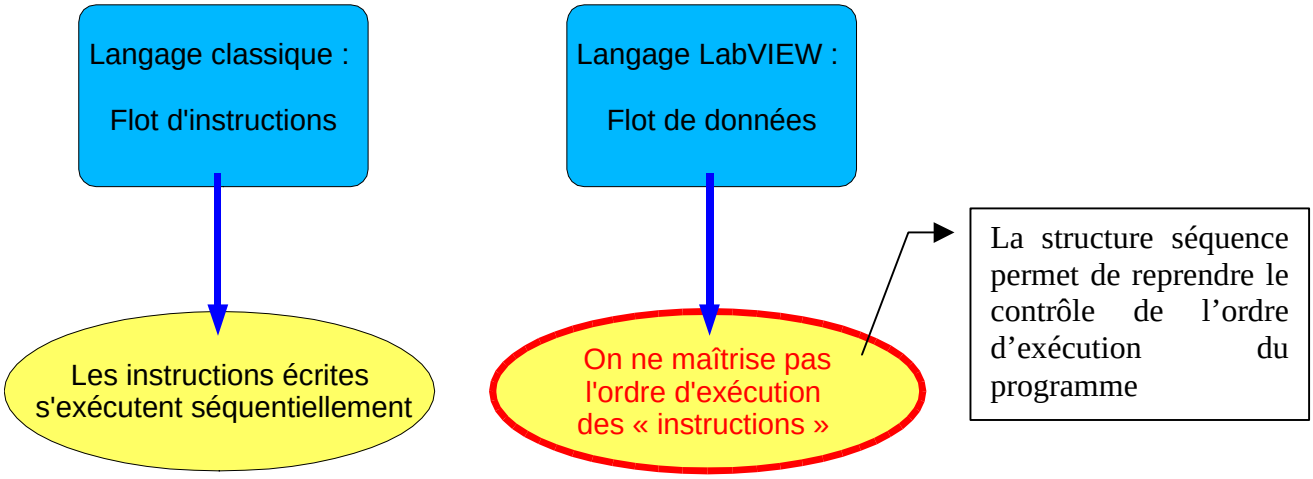
IV – NOEUDS DE RETROACTION



Par rapport au registre à décalage, le nœud de rétroaction simplifie le câblage et fait apparaître clairement l'appel du résultat du calcul du tour de boucle précédent.

Chapitre 10 – LES SEQUENCES

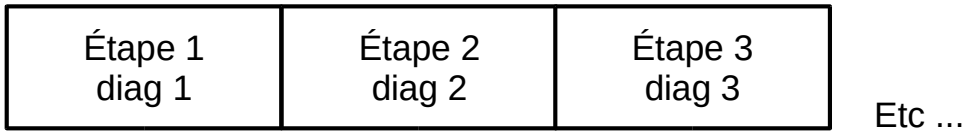
I – INTRODUCTION



II – LA STRUCTURE SEQUENCE DANS LabVIEW

La structure séquence se présente comme une succession d'étapes semblable à la pellicule d'un film

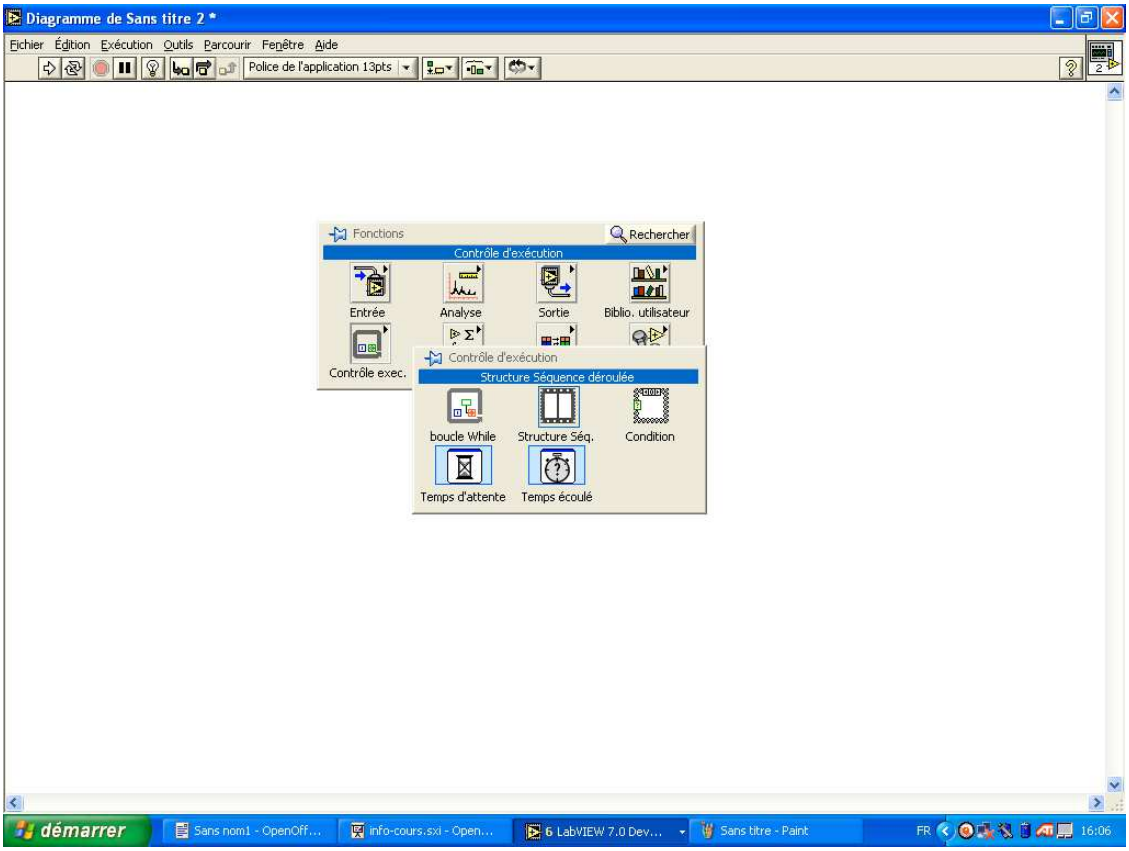
Dans chaque étape (frame) : un diagramme.
Les étapes successives sont exécutées dans l'ordre.



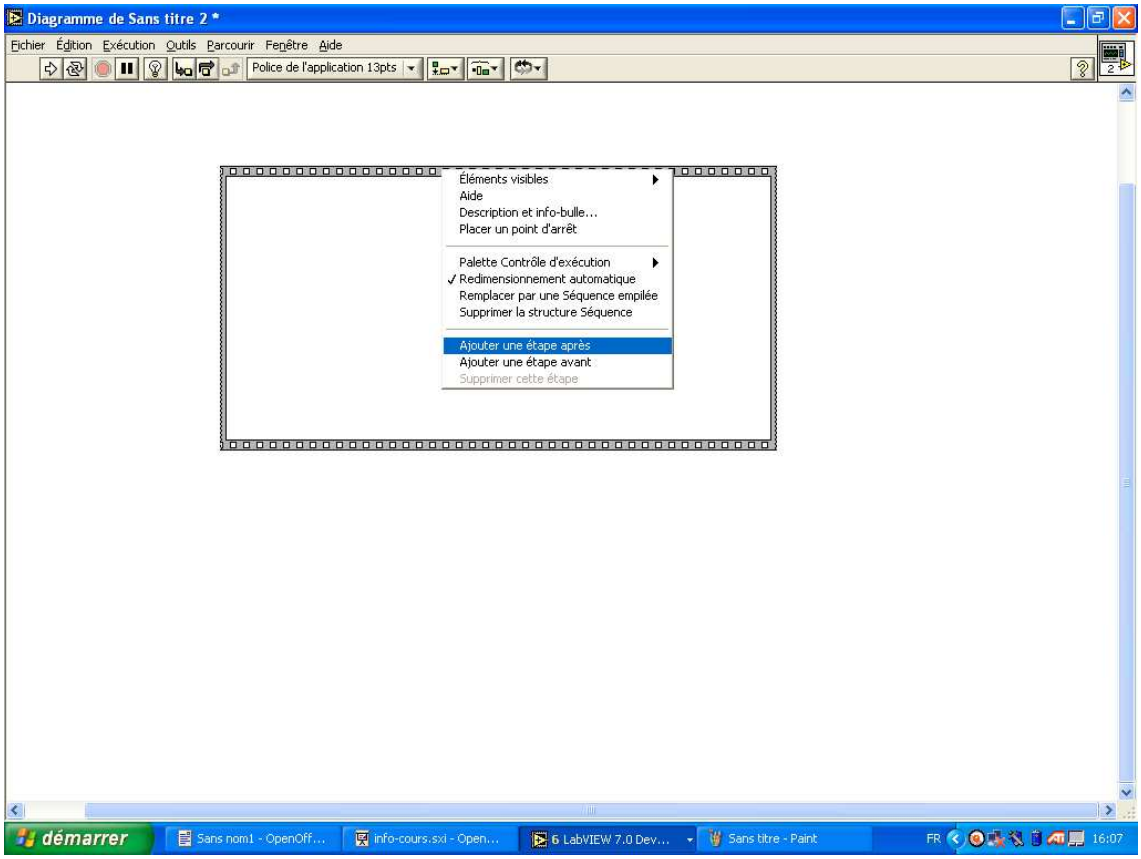
Etc ...

Les différentes étapes peuvent être empilées (une seule visible à la fois) ou déroulées.

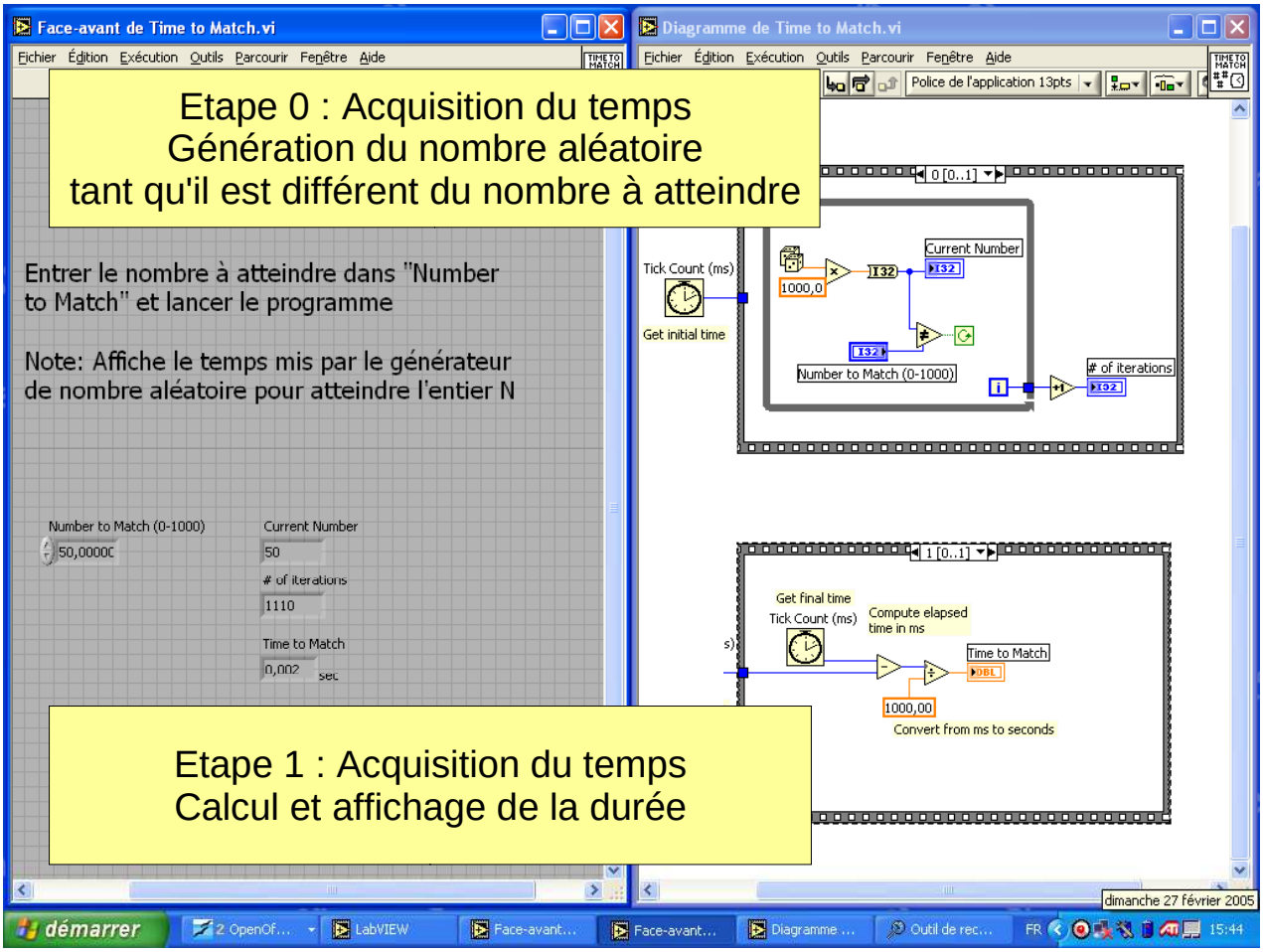
On retrouve cette structure dans la palette de fonctions, contrôles d'exécution.



On ajoute les étapes grâce au menu local apparaissant par clic droit sur le bord de la séquence.



III – EXEMPLE



IV – UTILISATION EN INFORMATIQUE D'INSTRUMENTATION

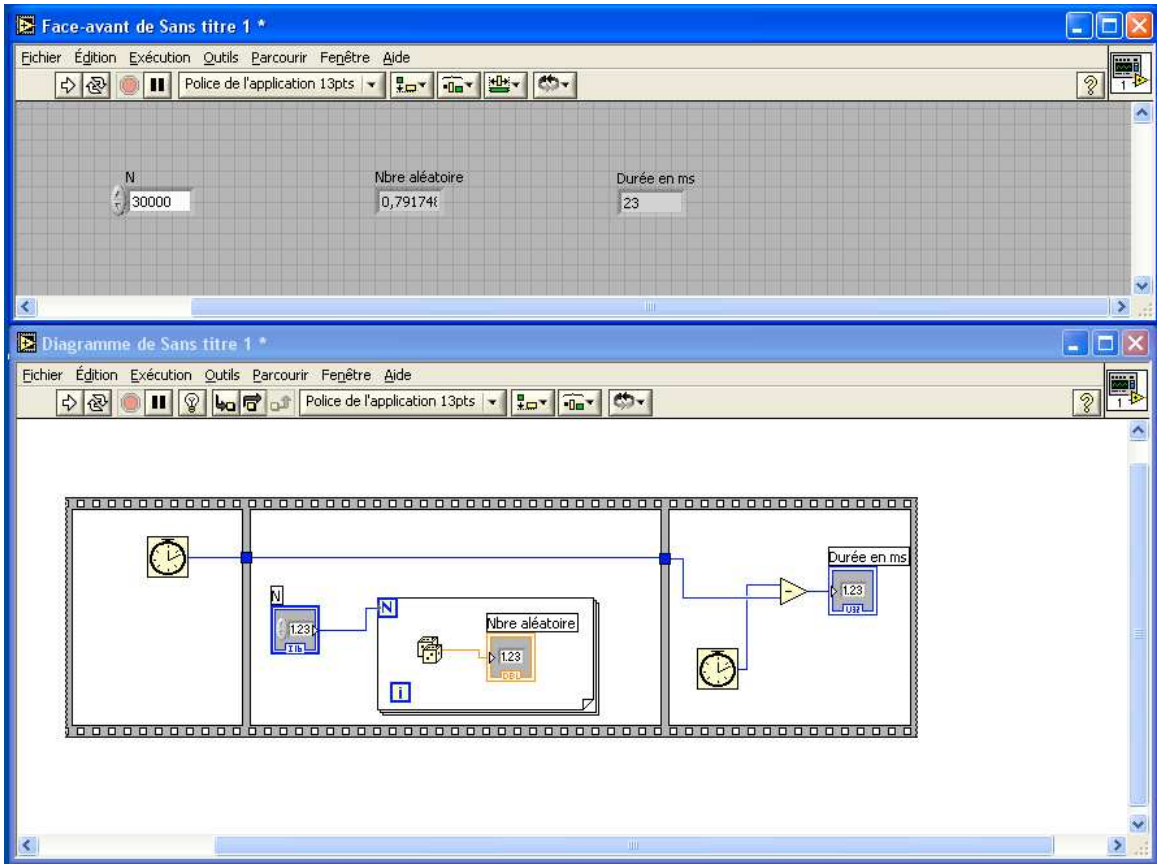
Cette structure est souvent utilisée
en informatique d'instrumentation

Étape 1 Configuration carte d'acquisition	Étape 2 Acquisition Sortie Commande	Étape 3 RAZ carte daq

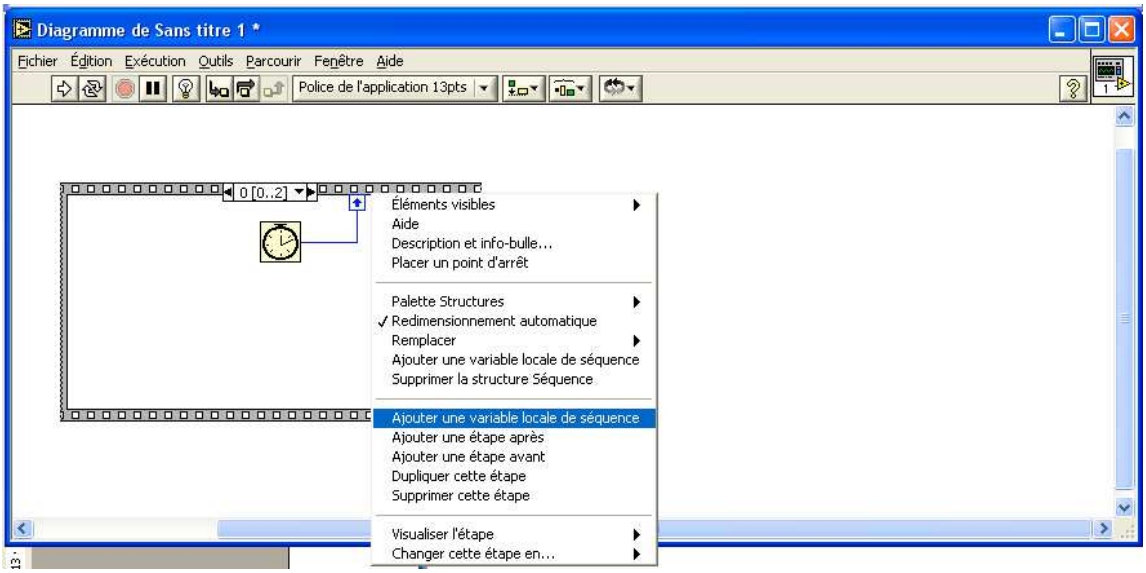
Cette structure peut également être utilisée également pour commander un système séquentiel (feux de carrefour)

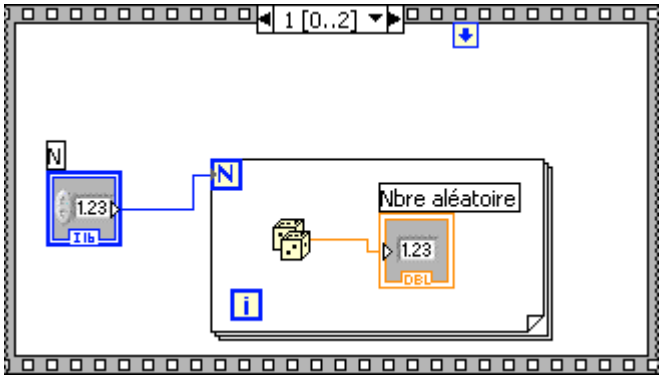
V – PASSAGE DE VALEURS DANS LA SEQUENCE

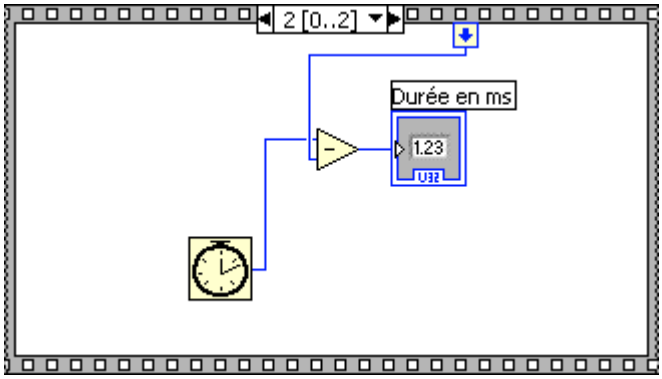
Le passage de valeurs d'une étape à l'autre d'une séquence déroulée se fait naturellement au moyen d'un tunnel.



Par contre dans une séquence empilée, il faut utiliser variable locale de séquence : clic droit sur le bord de la séquence. On dispose alors d'une flèche entrante distribuant la donnée dans les autres étapes de la séquence.



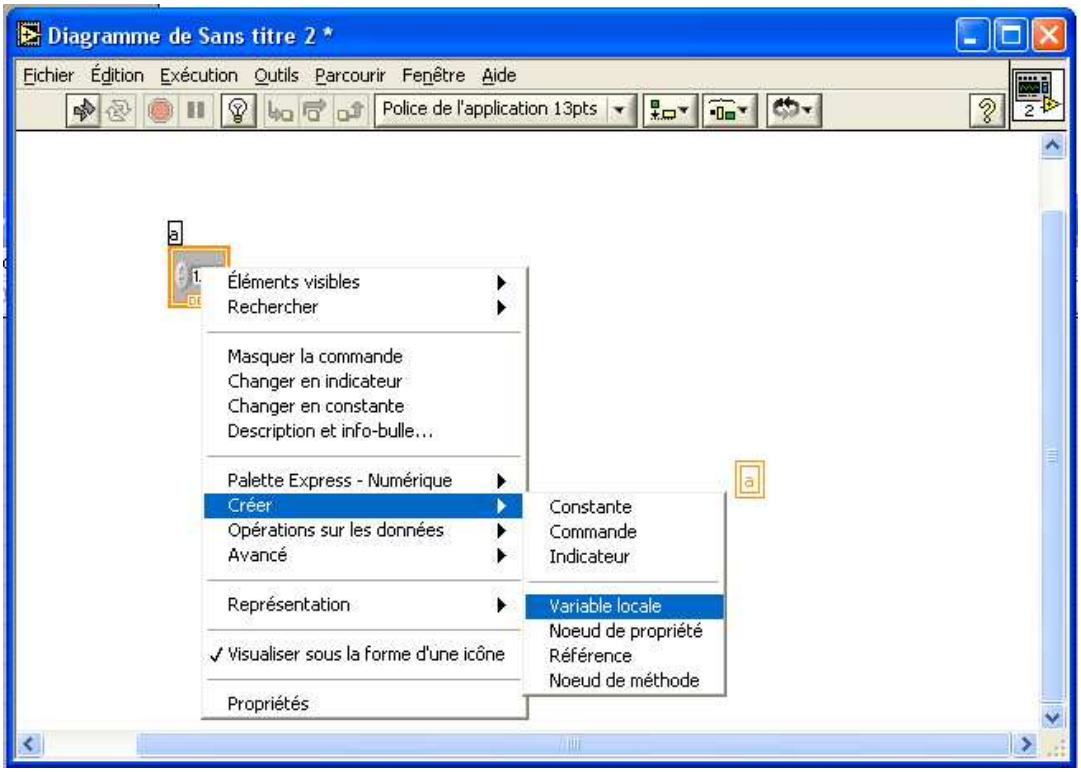

1 [0..2]
N
1.23
I16
N
Nbre aléatoire
1.23
DBL
i


2 [0..2]
Durée en ms
1.23
U32

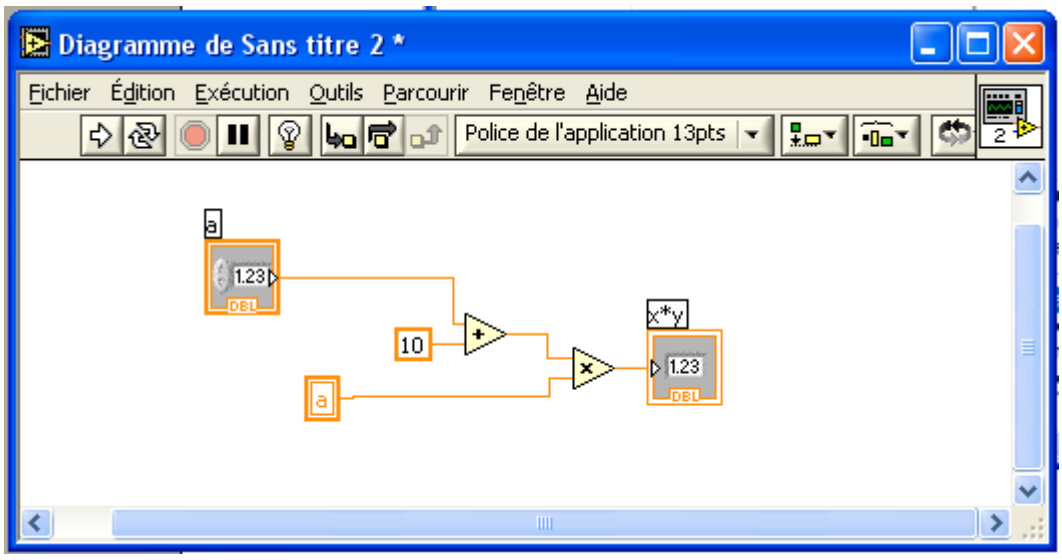
Chapitre 10 – COMPLEMENTS

I – VARIABLES LOCALES

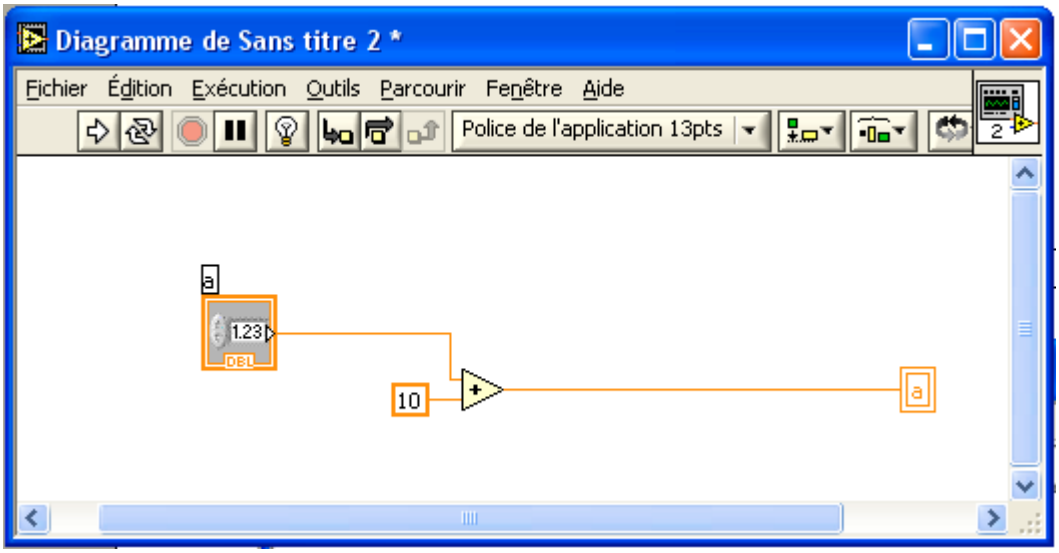
Pour simplifier le câblage (éliminer des fils de trop grande longueur), on peut avoir recours à une variable locale. La variable locale créée prend le nom de la variable d'origine, et peut être utilisée en lecture ou en écriture, forçant ainsi la donnée à changer de valeur.



Dans l'exemple ci-dessous, la variable locale est en lecture (variable d'entrée) et apparaît comme une copie de a. Le terminal de sortie affichera donc (10+a)*a.

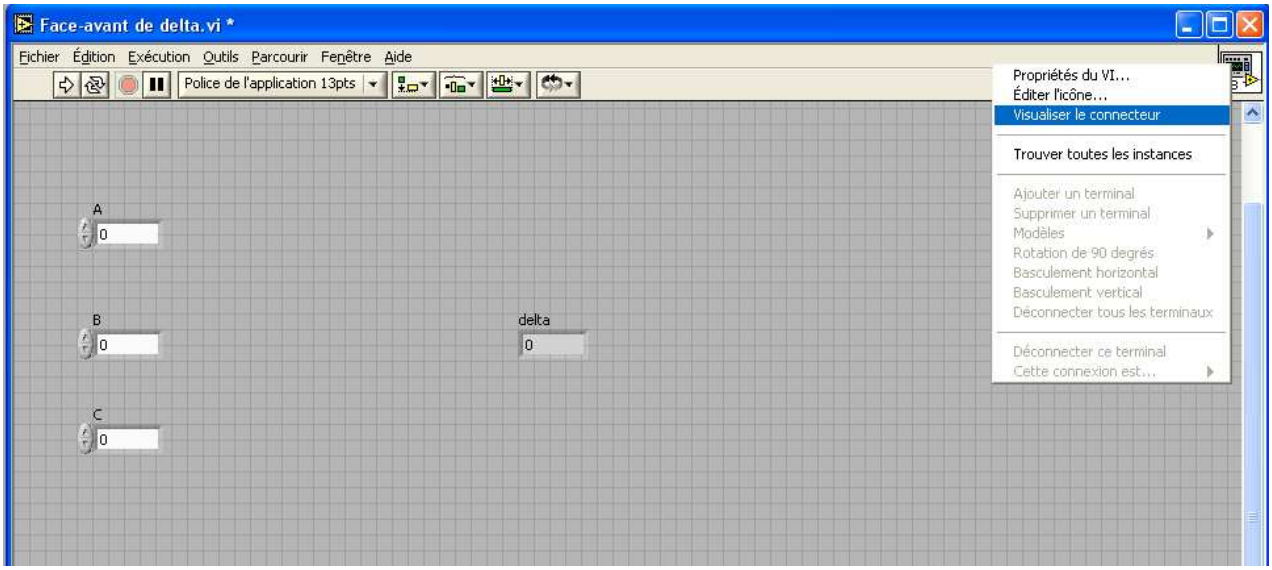


Dans l'exemple ci dessous, la variable locale est en écriture. Si la valeur de a était 1 au lancement du programme, elle se transforme en 11 lors de son arrêt.

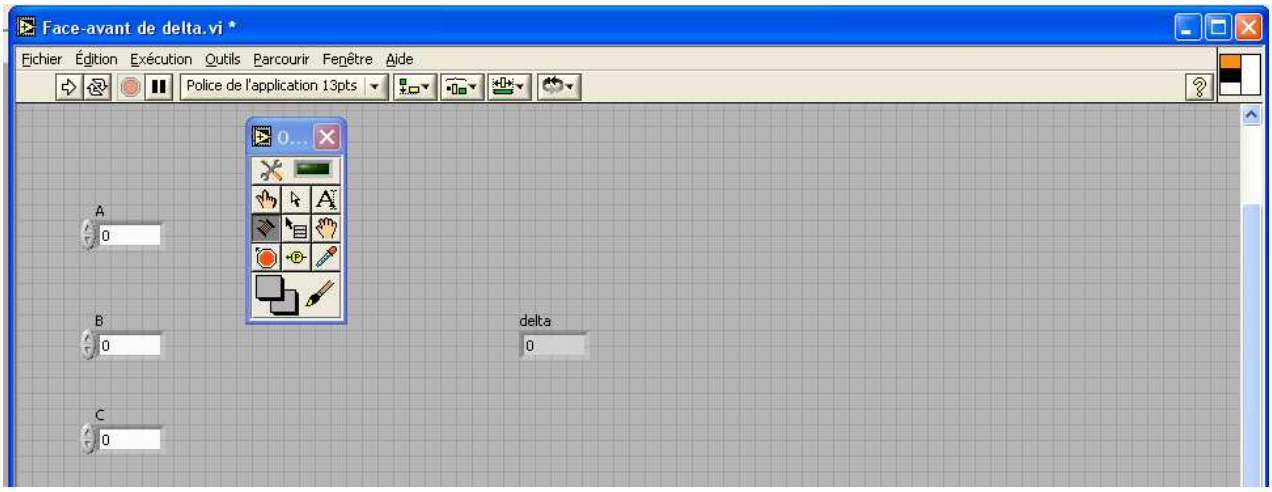


II – SOUS PROGRAMMES : ENCAPSULATION

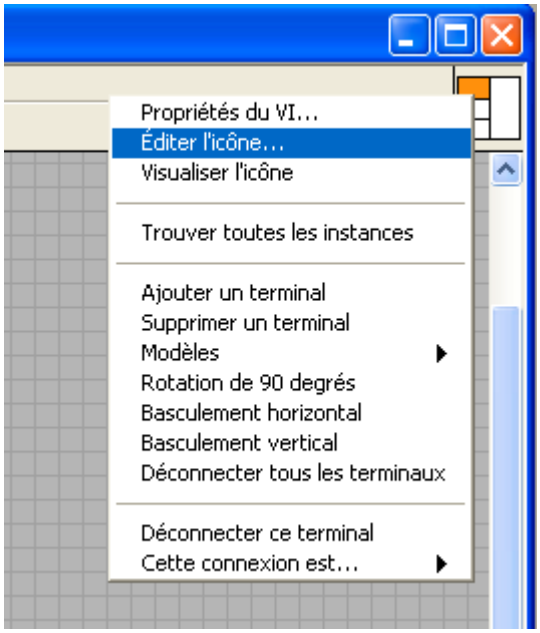
Nous souhaitons créer un sous programme encore qualifié de sous VI, de calcul du discriminant de l'équation du second degré.
Un Clic droit en haut à gauche sur l'icône LabVIEW permet d'ouvrir le menu local donné ci-dessous. Choisir Visualiser le connecteur.



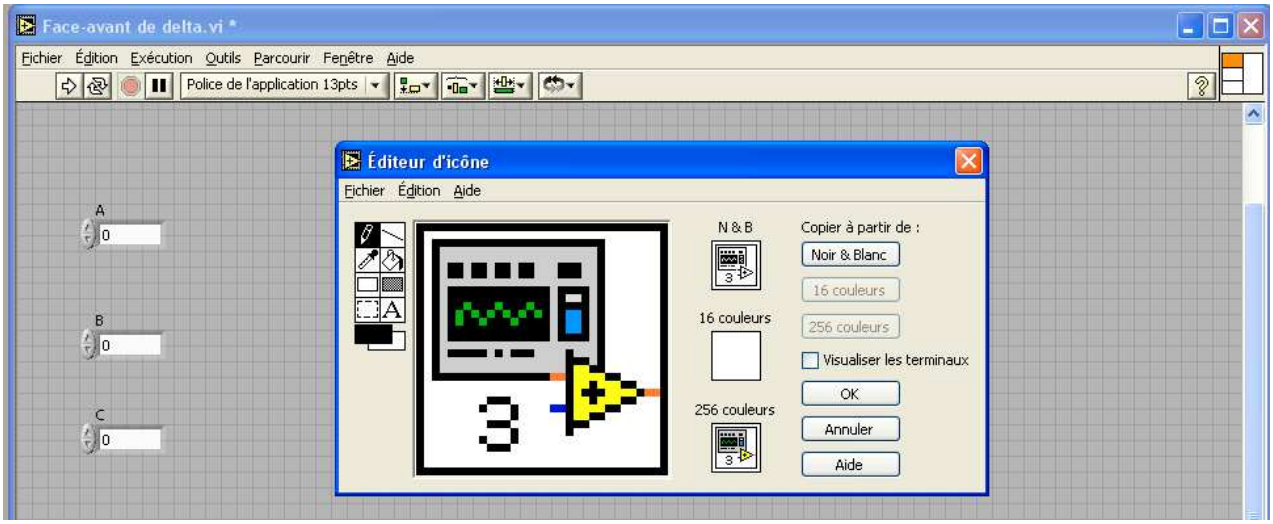
LabVIEW sait que le programme manipule trois variables d'entrée et une sortie, et propose le connecteur correspondant. Il peut cependant être changé. Il convient ensuite d'associer à chaque rectangle du connecteur une variable d'entrée ou sortie. Pour cela il faut cliquer à l'aide de l'outil bobine, d'abord sur l'objet de la face avant correspondant à la variable, puis sur le rectangle de l'icône.



Ensuite, on peut modifier l'icône correspondant à ce sous programme : Choisir modifier l'icône dans le menu local obtenu par clic droit sur le connecteur.

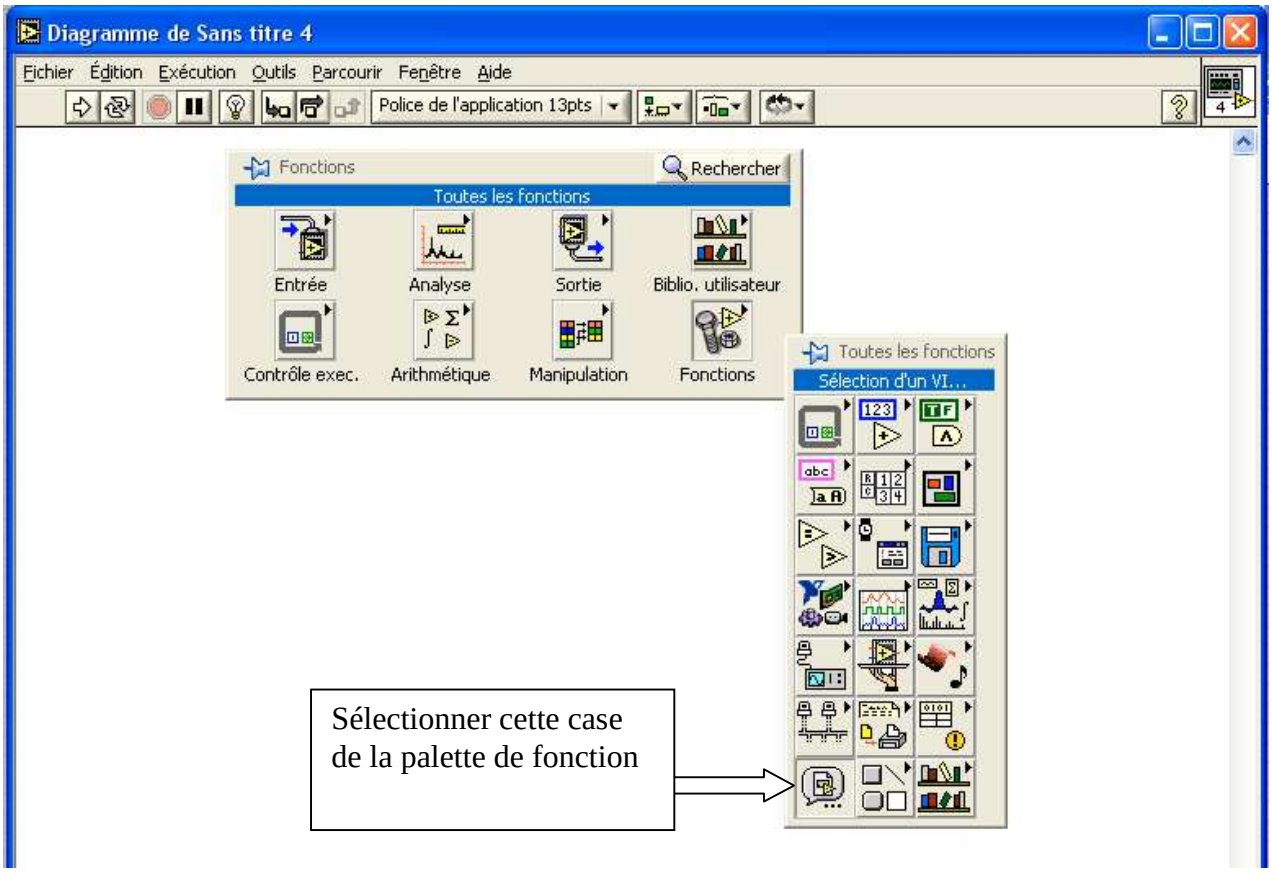


Il apparaît alors la fenêtre ci-dessous permettant de modifier, grâce à une palette de dessin, l'apparence de l'icône associée au sous VI.



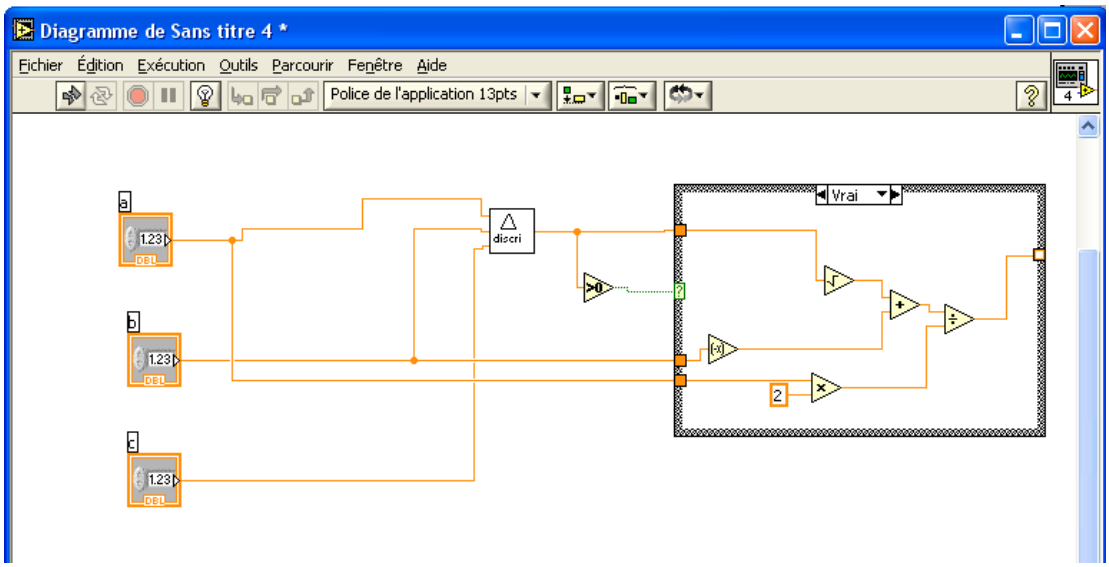
Il convient ensuite de sauvegarder ce sous VI à l'emplacement désiré.

Appel au sous dans un programme principal :
Dans le diagramme du programme principal, on utilise la palette de fonction :



Une boîte de dialogue invite à ouvrir le sous VI souhaité.

On dépose ainsi l'icône correspondante dans le diagramme et elle peut être utilisée comme n'importe quel VI LabVIEW. (Voir ci-dessous).



TP n° 1 et 2 – ENTREES SORTIES, OPERATEURS (Ch1 et Ch2)

EXERCICE 1.1

Soient 2 variables a et b numériques placées en face avant.
Afficher sur 4 sorties différentes : s1 = a+b, s2 = a – b, s3 = a*b et s4 = a / b

EXERCICE 1.2

Soit a une variable d'entrée de type entier.
Sur un afficheur numérique, afficher a sous le format binaire, hexadécimal et décimal. (Cf page 12)

Changer la représentation de a et s : U8, I8, U16, I16 et observer les valeurs extrêmes prises en charges par ces différentes représentations. (Cf page 10)

EXERCICE 1.3

Dans l'exercice précédent, ajouter une deuxième variable entière d'entrée b et afficher a+b.
Donner les valeurs limites des différentes représentations U8, I8, U16, I16.
Que se passe t-il quand le résultat dépasse les valeurs extrêmes ?

EXERCICE 1.4 – Réalisation d'un calcul

1°/ Utilisation d'opérateurs

Le gain en dB d'un circuit RC est donné par $G = 20 \log [1/\sqrt{(1 + RC2\pi f)^2}]$
Afficher sur un indicateur numérique le résultat de calcul de G pour les valeurs de R, C et f affichées sur des contrôleurs numériques en face avant.

Exemple de valeurs réalistes : R = 4,7 kΩ, C = 10 nF, f ≈ 1000 Hz.

2°/ Utilisation d'une boite de formule

Même question en utilisant la boite de formule (express/arithmétique)

3°/ Utilisation d'une boite de calcul

Même question en utilisant la boite de calcul (programmation/boîte de calcul)

EXERCICE 1.5

On veut également afficher le déphasage : φ = - arctan (RC2πf)

1°/ Modifier le programme de l'exo 4 1°/ pour afficher le gain et le déphasage

2°/ Idem avec les boites de formules

3°/ Idem avec une boite de calcul unique.

EXERCICE 1.6 – Opérateurs booléens

Réaliser un programme qui :

- allume un afficheur booléen L1 si deux interrupteurs K1 et K2 sont enfoncés.
- allume un afficheur booléen L2 si deux interrupteurs K1 ou K2 sont enfoncés.
- allume un afficheur booléen L3 si deux interrupteurs K1 ou K2 sont enfoncés, mais pas les deux.

EXERCICE 1.7 – Nombres complexes

Réaliser un programme qui calcule les racines d'une équation du second degré ($ax^2+bx+c = 0$) en utilisant deux afficheurs numériques de représentation complexe double précision (CDB).
Les coefficients a, b et c sont des contrôleurs numériques sur la face avant, représentation nombre à virgule flottante double précision (DBL).

EXERCICE 1.8 – Opérateur division euclidienne

1°/ Réaliser un programme qui convertit un nombre de mois, de jours, d'heures, de minutes et de secondes en secondes.

2°/ Réaliser un programme qui convertit un nombre de secondes en mois, jours, heures, minutes et secondes.

EXERCICE 1.9 – Chaînes de caractères

Réaliser un programme qui affiche sur un indicateur chaîne de caractère unique le contenu deux commandes de chaînes de caractère.

TPn°3 – TESTS ET STRCTURE CONDITIONNELLE (Ch3)

EXERCICE 3.1

On place 2 potentiomètres à glissière a et b sur la face avant.
Si a > b allumer une led, si a ≤ b éteindre la led

EXERCICE 3.2

On place 2 potentiomètres à glissière a et b sur la face avant et un afficheur numérique s

Si a > b alors s = a + (b-a)/2
Si a ≤ b alors s = b + (a – b)/2

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

EXERCICE 3.3 - Testeur de batterie

La tension d'une batterie 12 V est simulée par un potentiomètre U à glissière en face avant. Le résultat du test de la batterie est affiché sur 3 leds différentes.

Si 10 < U < 12 => Led Verte allumée
Si 8 < U < 10 => Led Orange allumée
Si 0 < U => Led Rouge allumée

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

TPn°4 – BOUCLE WHILE (Ch4)

EXERCICE 4.1 – Boucle d'attente

Réaliser un programme qui attend l'appui sur un bouton poussoir avant d'afficher le résultat de a+b.
N. B : On ne lancera pas l'exécution du vi par l'exécution récurrente !

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

EXERCICE 4.2 – Lecture un point

Ecrire un programme qui lit en permanence un potentiomètre placé en face avant et affiche la partie entière de la valeur prise. L'arrêt du programme est réalisé par appui sur un bouton poussoir stop placé en face avant. On ne lancera pas l'exécution du vi par l'exécution récurrente !
La valeur lue sera actualisée toutes les N s (N est une commande numérique en face avant)

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

EXERCICE 4.3 - Suite arithmétique

Réaliser un programme qui affiche successivement toutes les secondes les valeurs de la suite géométrique définie par son premier terme u_0 et sa raison r. Le programme s'arrête par appui sur un bouton stop. *Rappel :* $u_n = u_0 + n\,r$
U_0 et n sont des contrôleurs numériques sur la face avant.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI
4°/ Examiner l'influence de la position des terminaux de U_0 et q dans la boucle ou hors de la boucle. Quelle est la bonne solution ?

EXERCICE 4.4 – Nombre aléatoire

Réaliser un programme qui affiche sur un indicateur numérique un nombre aléatoire compris entre 0 et 20 généré périodiquement toutes les 0,5 s. Le programme s'arrêtera par appui sur un bouton stop.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

4°/ Compléter le vi de l'exo précédent de façon à afficher :

a) Le n° de l'itération (n° du tour de boucle en cours de réalisation).
b) Le nombre total d'itérations réalisées après appui sur le bouton stop.

TPn°5 – BOUCLE FOR (Ch5)

EXERCICE 5.1 – Nombre aléatoire

Réaliser un programme qui affiche sur un indicateur numérique trente nombres aléatoires compris entre 0 et 20 générés toutes les 0,5 s.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

EXERCICE 5.2 - Suite géométrique

Réaliser un programme qui affiche successivement toutes les secondes les 20 premières valeurs de la suite géométrique définie par son premier terme u_0 et sa raison q. *Rappel :* $u_n = u_0 * q^n$
U_0 et q sont des contrôleurs numériques sur la face avant.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI
4°/ Examiner l'influence de la position des terminaux de U_0 et q dans la boucle ou hors de la boucle. Quelle est la bonne solution ?

TPn°6 – TABLEAUX (Ch6)

EXERCICE 6.1

Réaliser un programme qui génère automatiquement un tableau de 10 nombres entiers compris entre 0 et 20.

Modifier ce programme pour afficher successivement toutes les secondes les éléments du tableau sur un afficheur numérique.

EXERCICE 6.2

Réaliser un programme qui génère automatiquement un tableau de 10 nombres entiers compris entre 0 et 20. (faire un copier coller de la 1ère partie de l'exercice précédent).

Une fois le tableau généré, on souhaite éliminer les 5 éléments les plus petits sans les remplacer, et afficher sur un afficheur numérique la moyenne des éléments du tableau.

a) Après avoir réaliser le tri du tableau
b) Sans trier préalablement le tableau et sans changer l'ordre des éléments

EXERCICE 6.3 – Indexation en sortie

Ecrire un programme qui génère un nombre aléatoire entier compris entre 0 et 100 toutes les 0,5 s et qui s'arrête par appui sur un bouton stop.

Afficher alors dans 2 tableaux distincts les nombres pairs et impairs générés.

EXERCICE 6.4 – Tableaux et boucles imbriquées

Générer un tableau de 15 lignes et 10 colonnes à l'aide de nombres entiers aléatoires compris entre – 10 et +10.
Modifier le programme pour remplacer les nombres pairs par 0.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

TPn°7 – TABLEAUX GRAPHES (Ch6, 7, 8)

EXERCICE 7.1 – Abscisses d'une échelle linéaire (Cf p 32-35)

Générer un tableau de N+1 points d'abscisse x compris entre a et b d'une graduation linéaire.
N, a et b sont des commandes numériques saisies par l'utilisateur du programme.

EXERCICE 7.2 – Graphe déroulant

Ecrire un programme qui génère et visualise sur un graphe déroulant une sinusoïde d'amplitude 10 V modifiable en face avant, de fréquence 1 Hz modifiable par une commande en face avant.
La période d'échantillonnage (intervalle de temps servant à cadencer la boucle) sera compatible avec la période de la sinusoïde et modifiable en face avant.
On affichera également la période de la sinusoïde sur un indicateur numérique.
La fonction sinus est disponible dans la palette de fonctions mathématiques.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI
4°/ Est-il préférable de placer les commandes d'amplitude et de fréquence dans ou hors de la boucle ?

EXERCICE 7.3 - Graphe

Ecrire un programme qui génère un tableau de N valeurs de la fonction $y(t) = 10 \sin (2 \pi f t)$.
f sera réglable par un terminal en face avant mais ne doit pas être modifiable pendant le calcul des points.
On représentera deux périodes de cette fonction avec N points pour la représentation.

N sera réglable en face avant mais doit être tel qu'il y ait suffisamment de points calculés dans une période !

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI
4°/ Doit-on placer la commande de fréquence dans ou hors de la boucle ?

EXERCICE 7.4 – Graphe XY

Modifier le programme précédent pour générer également un tableau des valeurs d'abscisse t de façon à afficher la courbe y(t) sur un graphe XY gradué en temps.

EXERCICE 7.5 - Abscisses d'une échelle logarithmique (Cf p 32-35)

Générer un tableau de N+1 points d'abscisse x compris entre a et b d'une échelle logarithmique.
N, a et b sont des commandes numériques saisies par l'utilisateur du programme.

EXERCICE 7.6 - Graphe XY échelle logarithmique

Ecrire un programme permettant d'afficher sur un graphe la courbe de réponse en fréquence d'un circuit RC :

$$G(f) = 20 \log [1 / \sqrt{ (1 + (RC.2\pi f)^2) }]$$

R et C sont modifiables grâce à une commande numérique.
Le nombre de points N est modifiable également sur la face avant.
La courbe sera tracée de f_a = 10 Hz à f_f = 100000 Hz

On prendra R = 4700 **Ω**, C = 10 nF.

Le graphe utilisera en abscisses une échelle logarithmique (points régulièrement espacés) qui sera graduée en Hz.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

TP n°8 et 9 – BOUCLES ET REGISTRES A DECALAGE (Ch 8, Ch 9)

EXERCICE 8.1

Réaliser un programme qui affiche successivement toutes les secondes les 20 premières valeurs de la suite géométrique définie par son premier terme u_0 et sa raison q entrés au clavier sur des afficheurs numériques. Rappel : $u_n = u_{n-1} * q$

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

EXERCICE 8.2

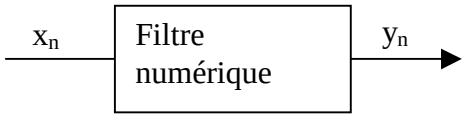
Tracer sur un graphe la courbe représentant les 10 premières valeurs de la suite récurrente définie par :

$$u_0 = -0,5 \quad \text{et } u_{n+1} = \sqrt{(u_n + 1)}$$

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

EXERCICE 8.3

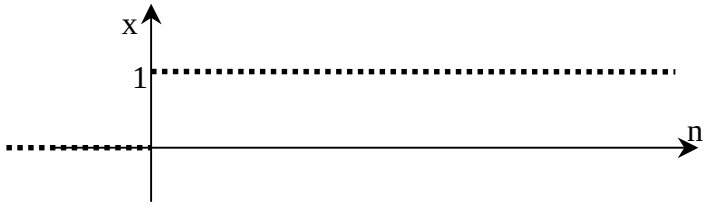
Soit un filtre numérique défini par son équation de récurrence :



$$y(n) = x(n) - 1,6180x(n- 1) + x(n- 2) + 1,5161y(n - 1) - 0,8780y(n - 2)$$

Initialement on a : $y_{-1} = 0$ et $y_{-2} = 0$

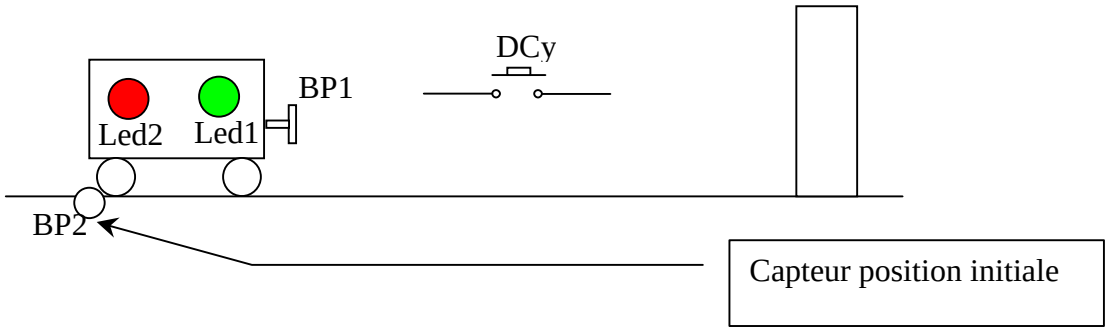
Tracer sur un graphe la courbe représentant la suite des valeurs de yn en réponse à une séquence d'entrée échelon : $x_n = \{1,1,1 \ldots.\}$ pour $n > 0$. $x_n = 0$ pour $n < 0$.



1°/ Donner l'organigramme.
2°/ Dessiner le diagramme correspondant sur feuille
3°/ Réaliser le VI

TP n° 10 – LES SEQUENCES (Ch 10)

EXZECICE 10.1



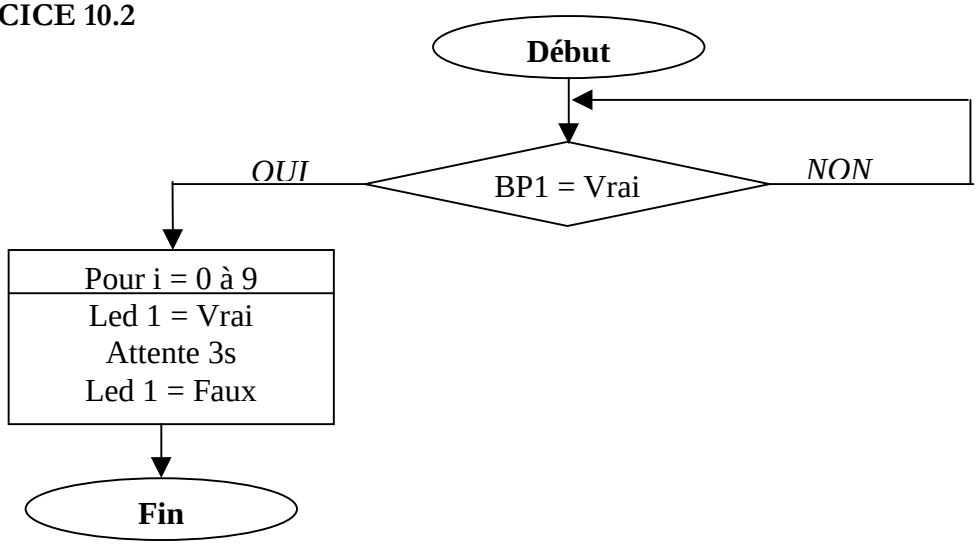
La marche avant est simulée par l'allumage d'une led verte (led1) La marche arrière est simulée par l'allumage d'une led rouge (led2).
Le wagon étant en position initiale, l'appui sur le bouton DCy (départ cycle) lancera la marche avant.
Le contact avec le mur est simulé par appui sur un bouton poussoir BP1

Simuler le fonctionnement du chariot qui attend l'appui sur DCy pour avancer et qui doit faire marche arrière quand le bouton poussoir BP1 est enfoncé. Le retour en position initiale est détecté par le contact simulé par un bouton poussoir BP2.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme sur feuille
3°/ Réaliser le vi en utilisant la structure séquence.

N. B : On ne lancera pas l'exécution du vi par l'exécution récurrente !

EXERCICE 10.2



1°/ Quelles sont les variables d'entrées et sorties apparaissant dans cet organigramme ?
2°/ Expliquer la fonction réalisée par cet organigramme
3°/ Transcrire cet organigramme en diagramme LabVIEW.

EXERCICE 10.3

Même dispositif. Simuler le fonctionnement du chariot qui attend l'appui sur DCy pour avancer et qui effectue 10 aller-retour détectés par appui sur le bouton poussoir BP. Le retour en position initiale est signalé par le capteur simulé par un bouton poussoir BP2 en face avant.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme sur feuille
3°/ Réaliser le vi en utilisant la structure séquence.

EXERCICE 10. 4 – Jeu de lumières

On utilise un chenillard constitué d'un tableau de 10 leds.

Dans un premier temps, les leds s'allument successivement. Ce cycle recommence trois fois consécutives.

Dans un deuxième temps les 10 leds clignotent 4 fois simultanément (allumage pendant ¼ seconde).

Dans un troisième temps, pendant ¼ s les leds de n° impair sont allumées pendants que les n° pairs sont éteintes puis inversement pendant ¼ s, ce cycle recommençant 3 fois.

Les trois phases doivent se succéder jusqu'à l'arrêt par appui sur un bouton Stop.

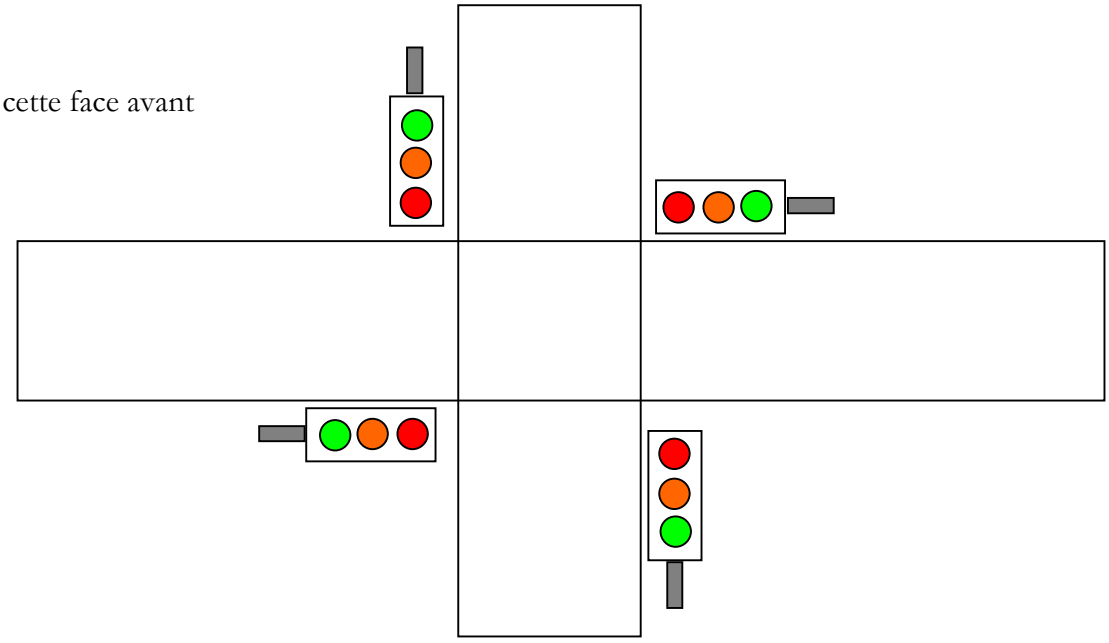
EXERCICE 10. 5 – Mesure d'un temps d'exécution

Réaliser un programme qui mesure le temps d'exécution d'une boucle for qui réalise 1000 fois la génération d'un nombre aléatoire.
On utilisera obligatoirement la structure séquence.

1°/ Donner l'organigramme.
2°/ Dessiner le diagramme sur feuille
3°/ Réaliser le vi en utilisant la structure séquence.

EXERCICE 10. 6 – Feux de carrefour

Réaliser cette face avant



Réaliser en utilisant la structure séquence, la programmation du fonctionnement normal des feux de circulation.

Les trois leds de chaque feu seront regroupées dans un cluster.

Pour utiliser la même variable dans différentes étapes d'une séquence, on crée une variable locale mise en écriture (Cf Chapitre 9 du fascicule).

TP n° 11 – EXPORT EN FICHIER TABLEUR

EXERCICE 11.1

Créer un tableau à 2 colonnes constitué de 500 points correspondant au tracé de la courbe représentative de la fonction f définie par :

$$f : [\,0\,,\,10\,\tau\,] \rightarrow R$$

$$t \rightarrow E\,(\,1 - \exp\,(\,-t\,/\,\tau\,)\,)$$

E et τ sont des commandes en face avant.

Ajouter du bruit sur la courbe : le bruit est simulé par la génération d'un nombre aléatoire. L'amplitude du bruit est une commande en face avant.

1°/ Ecrire le programme fabriquant le tableau et permettant le tracé de la courbe sur un graphe XY.

2°/ Par programmation, exporter le tableau en fichier tableur.

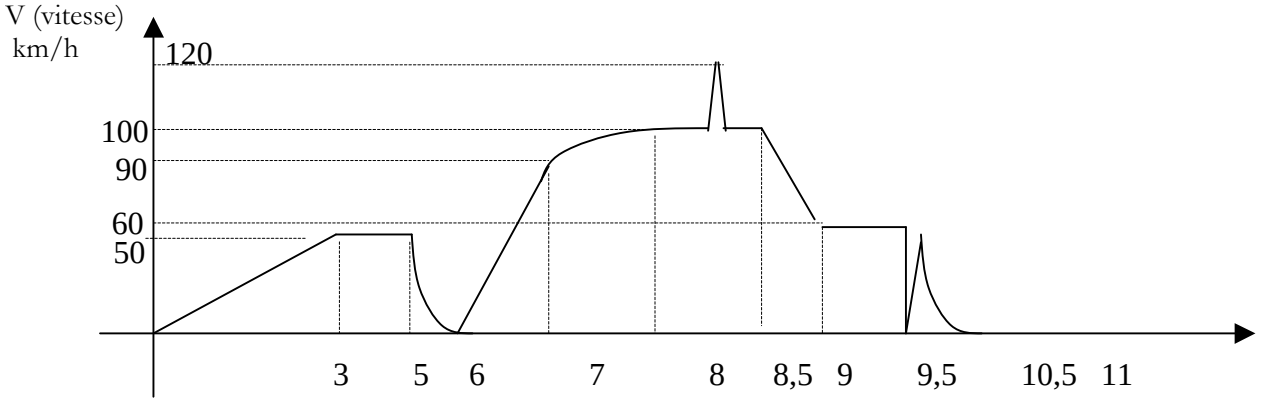
3°/ Ecrire un nouveau programme qui ouvre le fichier créé et qui trace la courbe sur un graphe XY.

4°/ Réaliser un traitement destiné à filtrer le bruit et afficher la courbe obtenue après traitement.

EXERCICE 11.2

L'enregistrement de la vitesse de rotation d'une roue de voiture a permis de représenter la variation de la vitesse du véhicule au cours du temps.

Créer un VI LabVIEW qui simule ce roulage. (utiliser la structure séquence et les boucles).



Tracer la courbe et exporter les données en fichier tableur.

EXERCICE 11.3 – Post traitement des données

Créer un VI qui ouvre le fichier tableur précédent et qui trace la courbe.

Compléter le VI de façon à traiter les données :

Déterminer l'accélération du véhicule supposé se déplacer en ligne droite.
Déterminer la distance parcourue
Tracer ces deux courbes.

Déterminer les instants où les roues ont patiné ou se sont bloquées.

Eliminer du tableau des vitesses les phénomènes de patinage ou de blocage des roues.

Exporter les données traitées dans un nouveau fichier tableur.

EXERCICE 11.4 –

Reprendre l'exercice 2 en ajoutant un bruit aléatoire d'amplitude réaliste sur la mesure de la vitesse.

Reprendre l'exercice 3 en réalisant en plus le filtrage de ce bruit.

EXERCICE 11.5 – Moyenne glissante

La moyenne glissante ou moyenne mobile est un type de moyenne statistique utilisée pour analyser des séries ordonnées de données, le plus souvent des séries temporelles, en supprimant les fluctuations transitoires de façon à en souligner les tendances à plus long terme. Cette moyenne est dite mobile parce qu'elle est recalculée de façon continue, en utilisant à chaque calcul un sous-ensemble d'éléments dans lequel un nouvel élément remplace le plus ancien ou s'ajoute au sous-ensemble.

Ce type de moyenne est utilisé généralement comme méthode de lissage de valeurs, en particulier dans le domaine financier pour l'analyse technique de cours boursiers.

Le site mentionné ci-dessous, donne les mesures de la pollution de l'air à Montbéliard concernant divers polluants. On s'intéresse au polluant PM10 (poussières microscopiques 10 µm) sur 48h pour Montbéliard centre.
http://www.atmo-franche-comte.org/pages/fr/menu1/les_mesures_en_direct_50.html

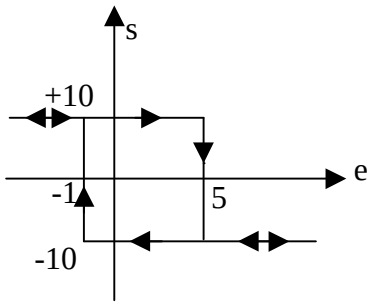
Ces données sont disponibles sur mpeea.free.fr page LabVIEW : fichier data à télécharger et à enregistrer.
Utiliser l'outil LabVIEW qui permet d'ouvrir un fichier tableur. Placer ces données dans un tableau.
Tracer le graphe correspondant.
Tracer également sur ce même graphe la moyenne glissante sur 8h.
Quel type de filtrage sur les données introduit la moyenne glissante ?

TP n° 12 – EXERCICES COMPLEMENTAIRES

EXERCICE 12.1 : Comparateur à hystérésis

Un comparateur a hystérésis comporte deux seuils différents selon le sens de variation de la grandeur d'entrée.

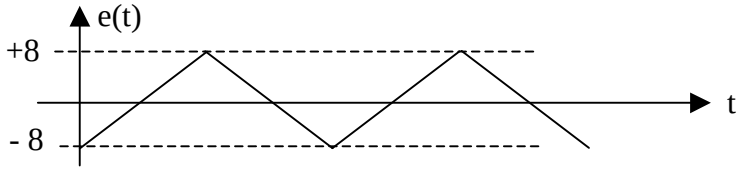
Le fonctionnement est décrit par la caractéristique de transfert s(e) donnée ci-dessous.



Quand e(t) augmente, s bascule à –10 quand e > 5.

Quand e(t) diminue, s bascule à +10 quand e < -1

1°/ Ecrire un programme affichant sur un graphe déroulant (waveform chart) la sortie s(t) pour une entrée triangulaire périodique (période T par exemple de 20 ms) tant que l'on n'a pas appuyé sur un bouton stop.



2°/ Faire apparaître la caractéristique de transfert s(e) sur un graphe XY.

EXERCICE 12.2 –

Représenter sur un graphe déroulant le signal modulé en amplitude :
$h(t) = \sin(2\pi f_m t) \sin(2\pi f_p t)$

f_p est la fréquence porteuse et f_m est la fréquence modulante :
fm << fp.

fm et fp sont des variables numériques disponibles sur la face avant. Elles seront initialisées à
fm = 0,1 Hz et fp = 1 Hz.

On souhaite disposer de 20 points par période de la porteuse tout en visualisant sur l'écran deux périodes du signal modulant.