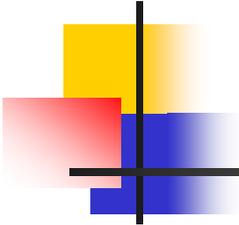


Introduction à LabVIEW



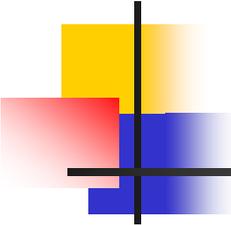
Introduction

LabVIEW est un langage de programmation graphique, particulièrement adapté aux montages expérimentaux.

Il est à la fois souple, puissant et rapide.

Atouts:

- programmation intuitive
- développement rapide
- interface utilisateur
- nombreuses bibliothèques disponibles
- drivers existants pour la plupart des instruments



Introduction

Interfaçage d'expériences

- Contrôle d'instruments
- acquisition des données

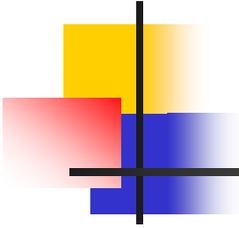
Analyse et traitement de données

- traitement du signal
- traitement statistique

LabVIEW

Présentation et stockage de données

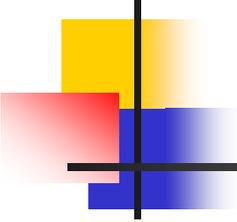
- Affichage courbes
- Stockage données



Vue d'ensemble

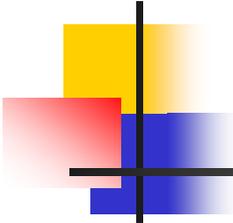
- Cours sur 2 jours
- présentation de LabVIEW.
- Les exercices et les exemples sont disponibles sur notre site, rubrique enseignement :

<http://lasim.univ-lyon1.fr/recherche/onlisno/Labview.html>

The logo graphic consists of a vertical black line and a horizontal black line intersecting at the center. To the left of the intersection, there are three overlapping squares: a yellow one at the top, a red one in the middle, and a blue one at the bottom. The word "LabView" is written in a blue, sans-serif font to the right of the vertical line.

LabView

- Langage de programmation graphique
- Développé par National Instruments
- Site web– www.ni.com/labview
- fondé sur le principe du flot de données
- programmation par câblage
- Très puissant



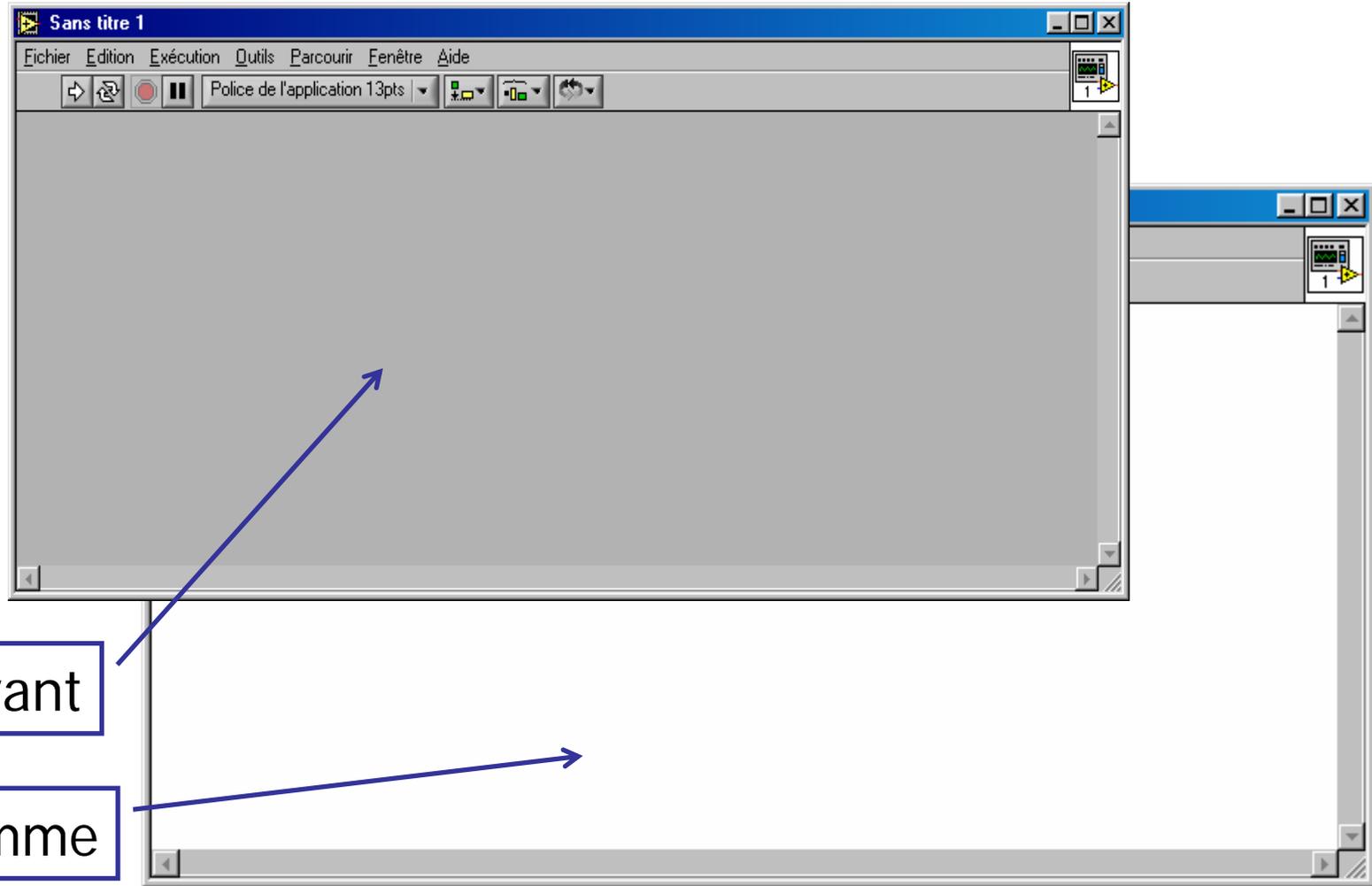
Pour commencer

- LabView 6i disponible sur vos machines
(version actuelle : 7.1)
- lancer LabVIEW :
**démarrer → programmes → National
Instruments → LabVIEW 6 → LabVIEW**

Boîte de dialogue au lancement



Ecrans à l'ouverture

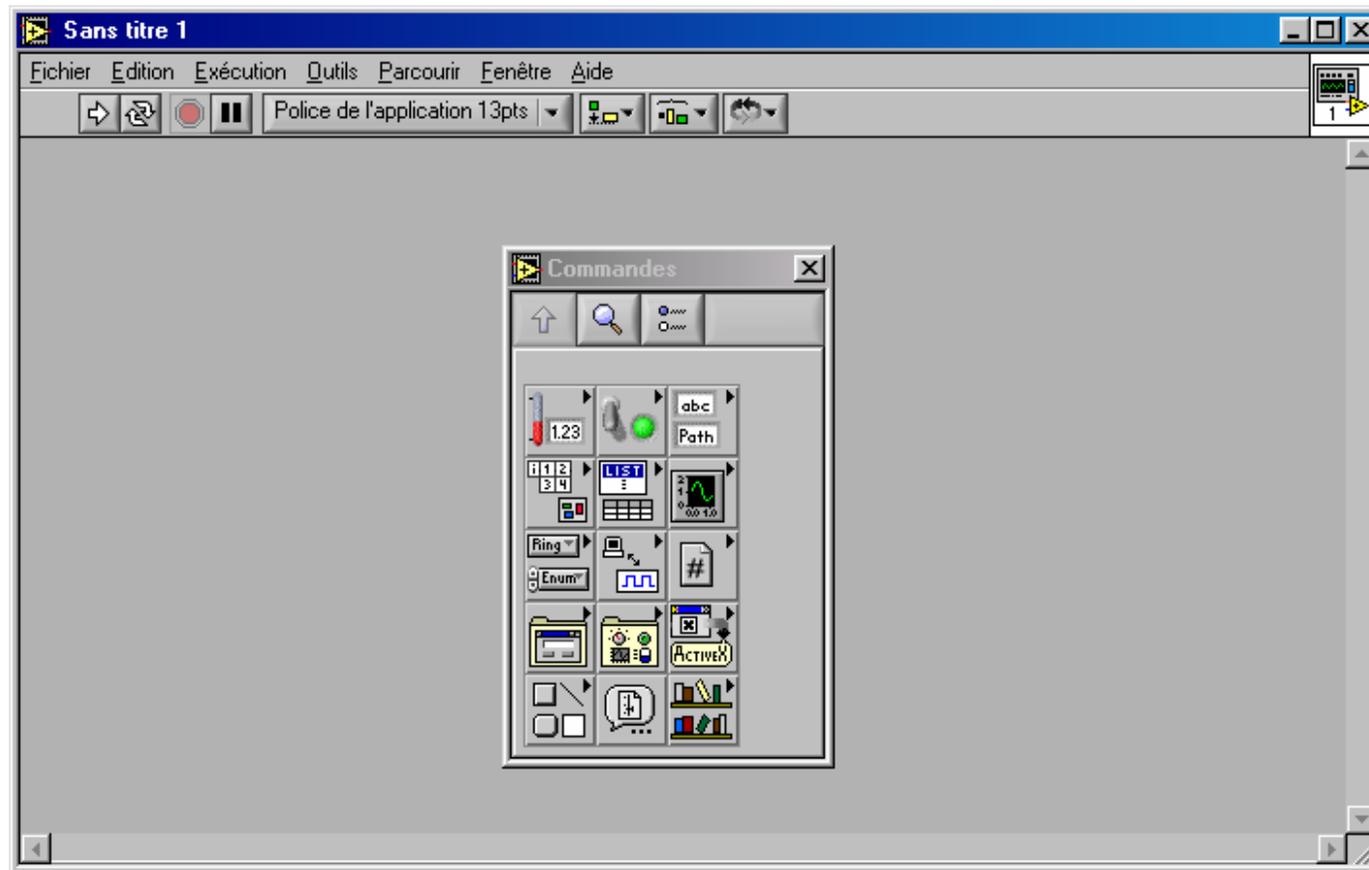


Face avant

Diagramme

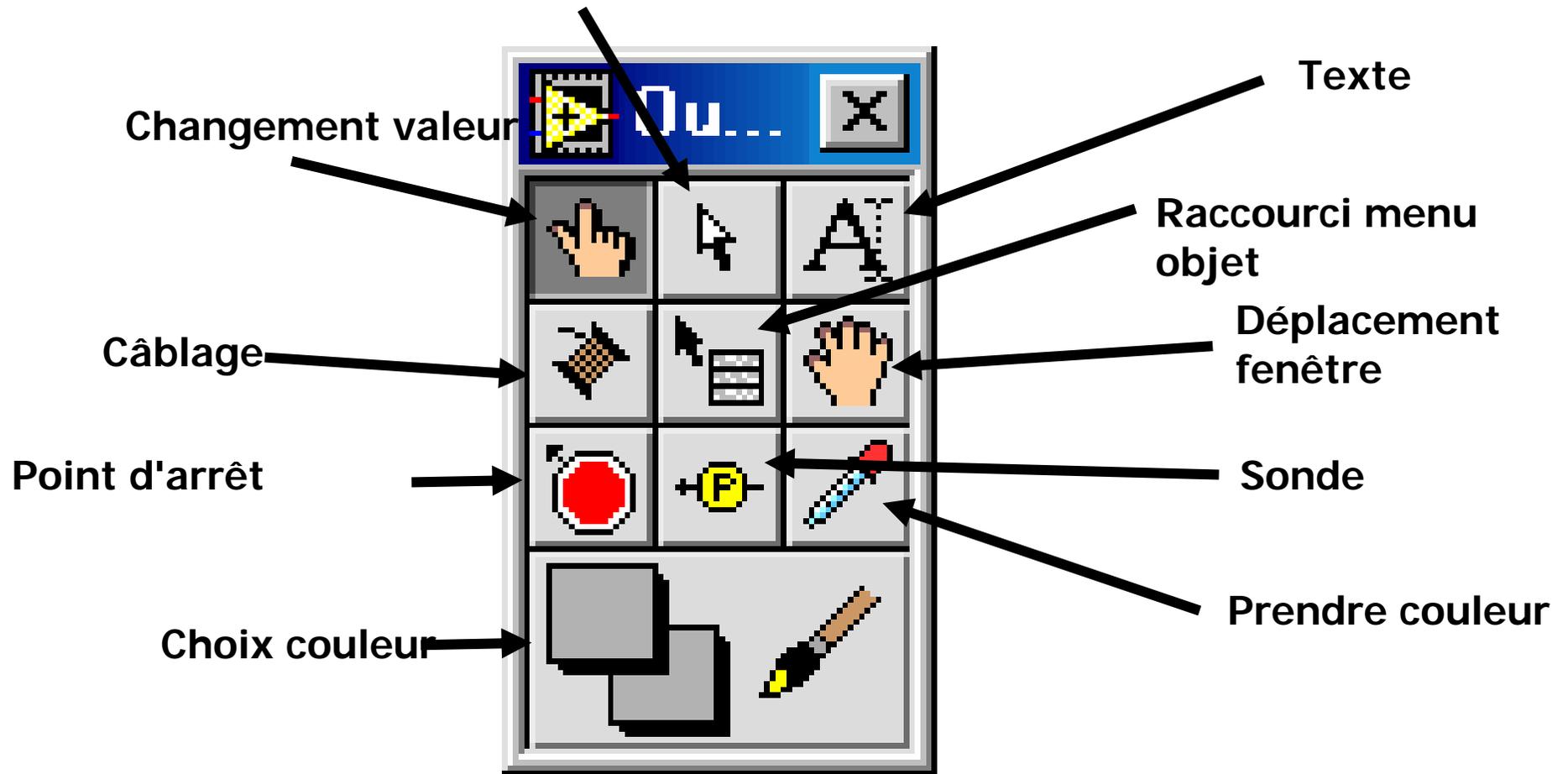
Face avant - commandes

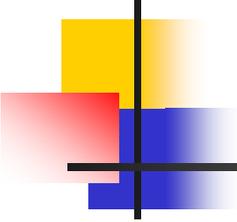
[Clic droit]



Outils

Position / Taille / Selection





Exemple

Création d'un VI simple :
fonctions mathématiques de base

But: créer un VI permettant de réaliser une
opération mathématique simple (addition) et
d'afficher le résultat.

Exemple: $1 + 3 \rightarrow 4$

Création d'un VI

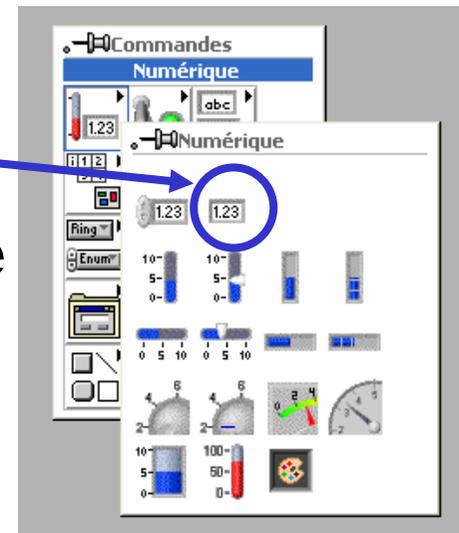
- Sélectionner : Nouveau VI (Virtual Instrument)
- Sélectionner la fenêtre face avant (grise)
- clic droit → Commandes
- Déplacer la souris sur ' *Numérique* '
- Une sous-fenêtre apparaît avec les commandes numériques



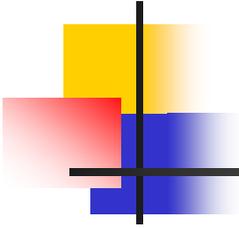
Indicateur numériques

- Sélectionner ' *indicateur numérique* ' (en haut à gauche) : déplacer la souris dessus puis clic gauche

L'indicateur numérique permettra l'affichage du résultat.



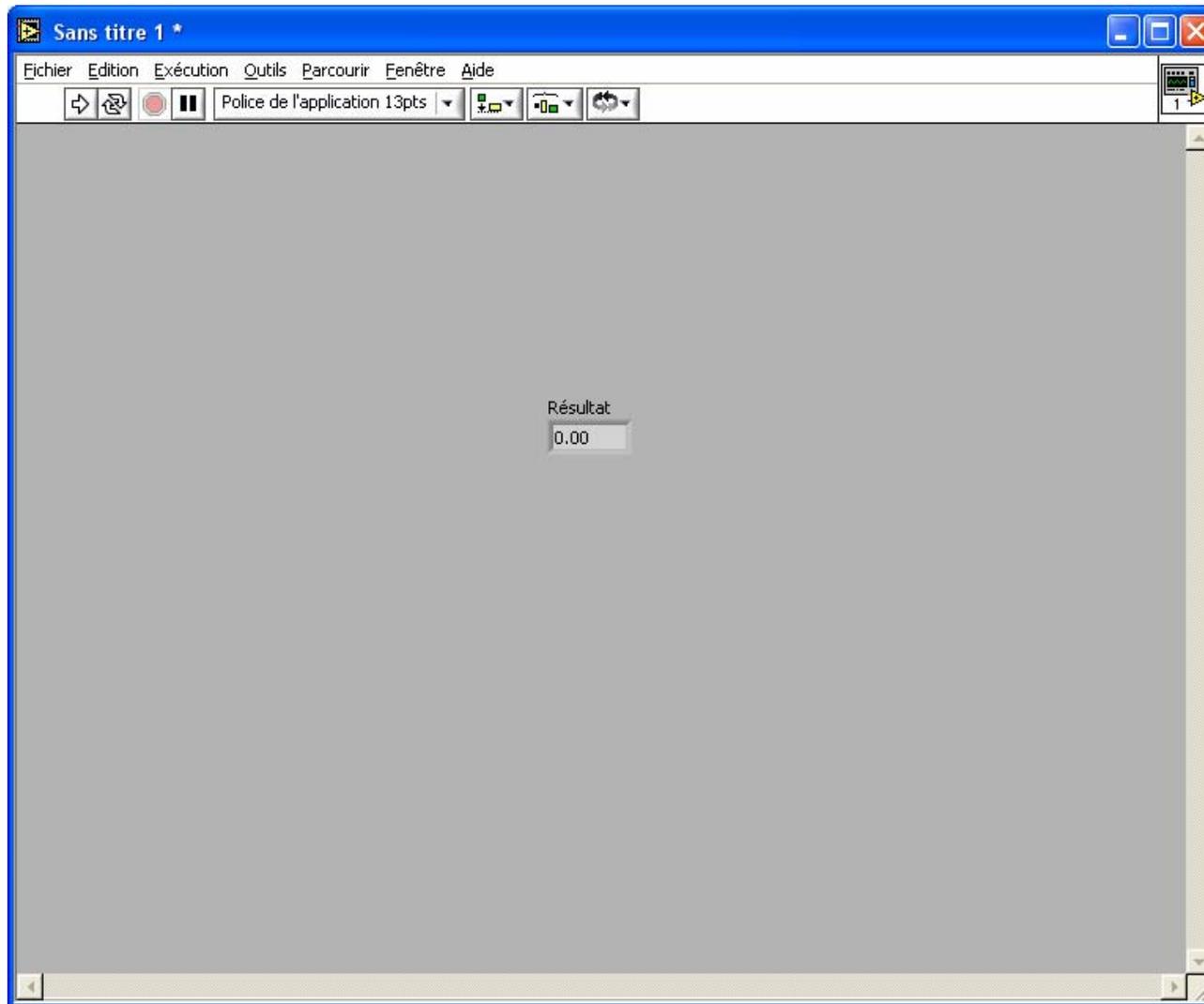
- Le curseur change → main
- Déplacer le curseur sur la face avant (grise) → double boîte en pointillés
- Amener cette boîte à la position souhaitée (clic gauche)



Indicateurs numériques - suite

- Une étiquette par défaut (' *Numérique* ') est automatiquement créée et surlignée
- On peut changer le texte aussitôt (par exemple : ' **résultat** ')
- On pourra aussi modifier l'étiquette plus tard

Exemple de VI de base



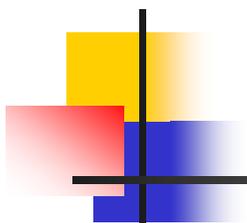
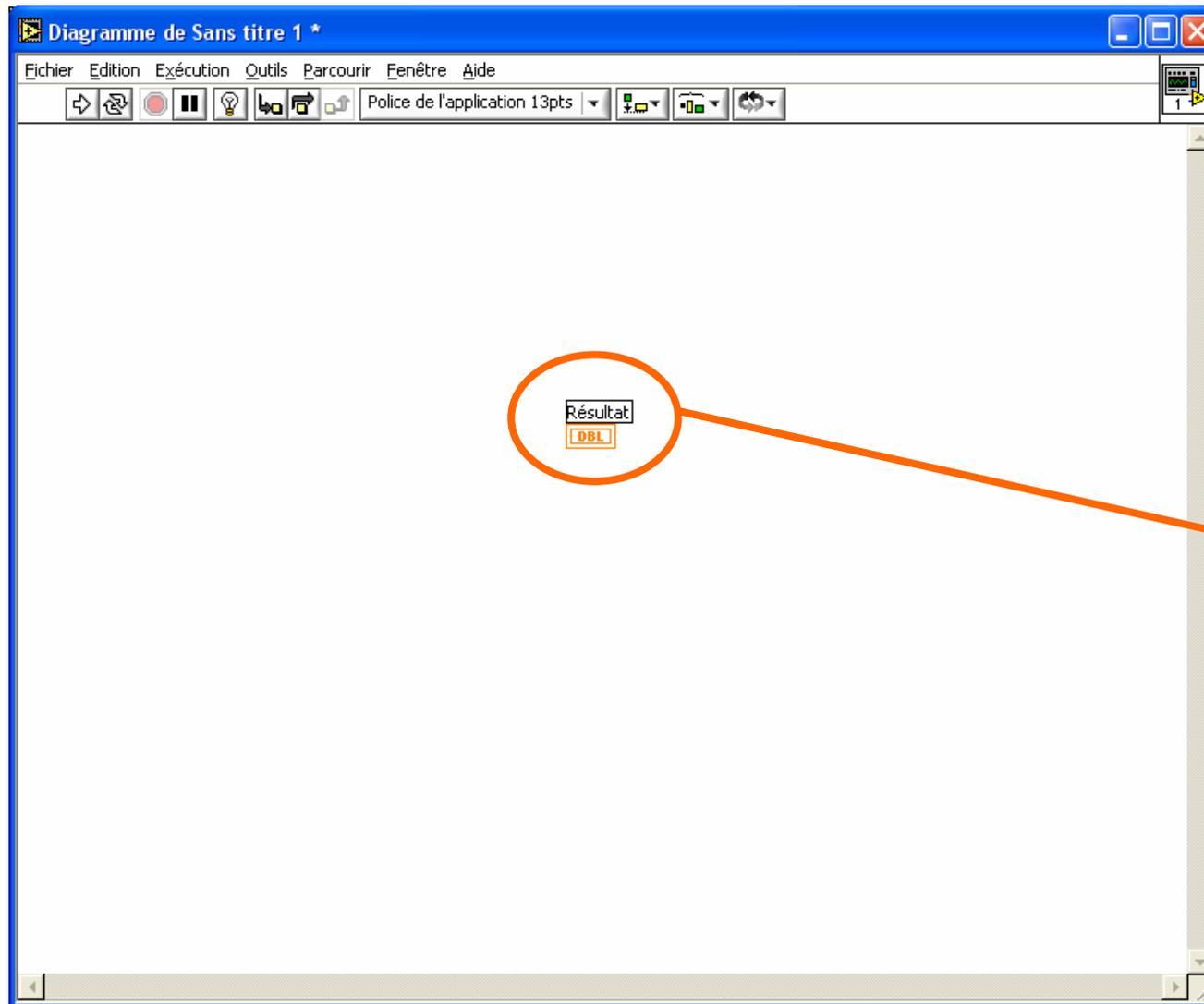


Diagramme (la partie opérationnelle)

- Visualiser le diagramme
 - `<ctrl+E>` pour passer de face avant ↔ diagramme
- LabVIEW a placé un terminal sur le diagramme qui est lié à l'indicateur placé sur la face avant
- On peut déplacer ce terminal où l'on veut sur le diagramme, cela ne change rien en face avant
- En double cliquant
 - sur un objet de face avant → terminal du diagramme
 - sur terminal du diagramme → objet de face avant

Diagramme (la partie opérationnelle)

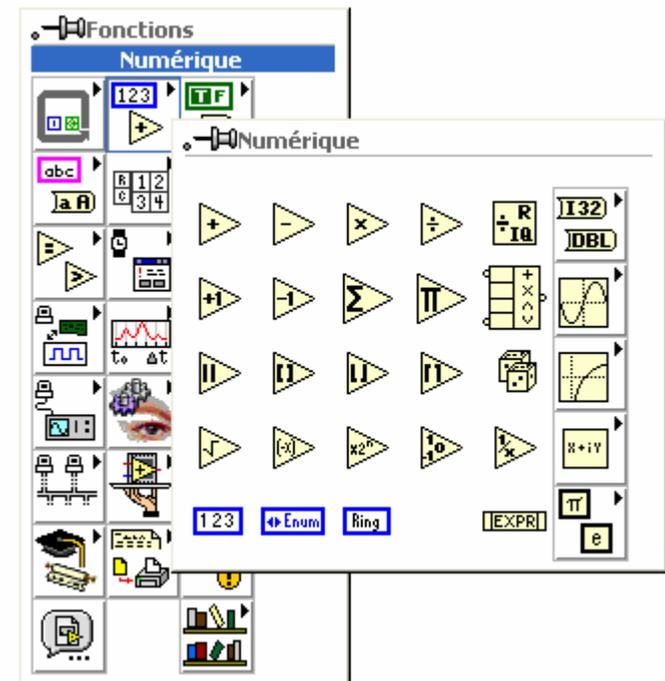


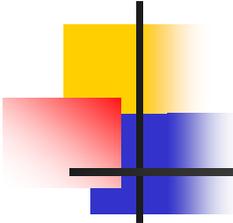
DBL est le type de sortie
(réel double)
On pourra changer
ce type plus tard

Ajouter une fonction

Nous allons maintenant ajouter la fonction mathématique d'addition.

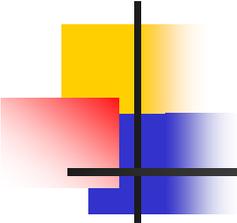
- Toujours sur diagramme, <clic droit> → la palette de fonction apparaît
- Déplacer la souris → sur numérique → la palette ' *numérique* ' apparaît avec les fonctions mathématiques
- Cliquer sur la fonction ' *ajouter* '
- Déplacer la souris vers le diagramme → on voit la fonction sous le curseur en forme de main
- Placer le curseur à gauche du terminal indicateur ' *résultat* '
- <clic gauche> pour poser la fonction





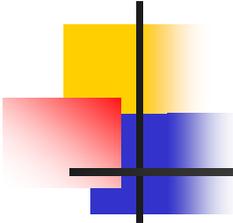
Ajouter les constantes

- Toujours sur diagramme, <clic droit> → la palette de fonction apparaît
- Déplacer la souris → sur numérique → la palette '*numérique*' apparaît avec les fonctions mathématiques
- Cliquer sur la fonction '*constante numérique*'
- Déplacer la souris vers le diagramme → on voit la fonction sous le curseur en forme de main
- Placer le curseur à gauche l'opérateur 'somme'
- <clic gauche> pour poser la fonction
- changer la valeur de la constante à 1
- ajouter une deuxième constante = 3



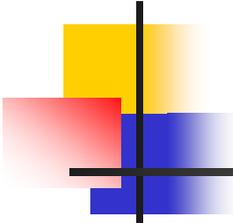
Câblage

- Dans la palette d'outils <shift+clic droit>, sélectionner l'outil de câblage (comme une bobine de fil)
le curseur change de forme (bobine)
- positionner le curseur sur la première constante
→ il clignote
- <clic gauche>, déplacer la souris → ligne en pointillés
- Amener le curseur sur la fonction ' *ajouter* ' (+)
- ...



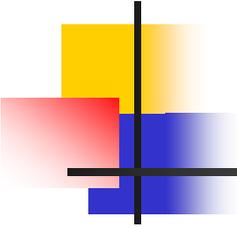
Câblage - suite

- Lorsque le curseur arrive sur la fonction, elle se met à clignoter sous le curseur au niveau des terminaux de connexion
- De courtes lignes de connexion apparaissent aussi
- Convention standard :
 - terminaux d'entrée à gauche
 - terminaux de sortie à droite
- Cliquer sur le connecteur en haut à gauche
- Le câble passe en trait continu orange : câblage valide



Câblage - directions

- Les câbles vont toujours horizontalement ou verticalement, jamais en diagonale
- Par défaut, il y a un changement de direction par câble, mais
 - On peut changer la direction initiale (horizontale ou verticale) en appuyant sur `<barre d'espace>`
 - On peut imposer des points intermédiaires par `<clic gauche>`

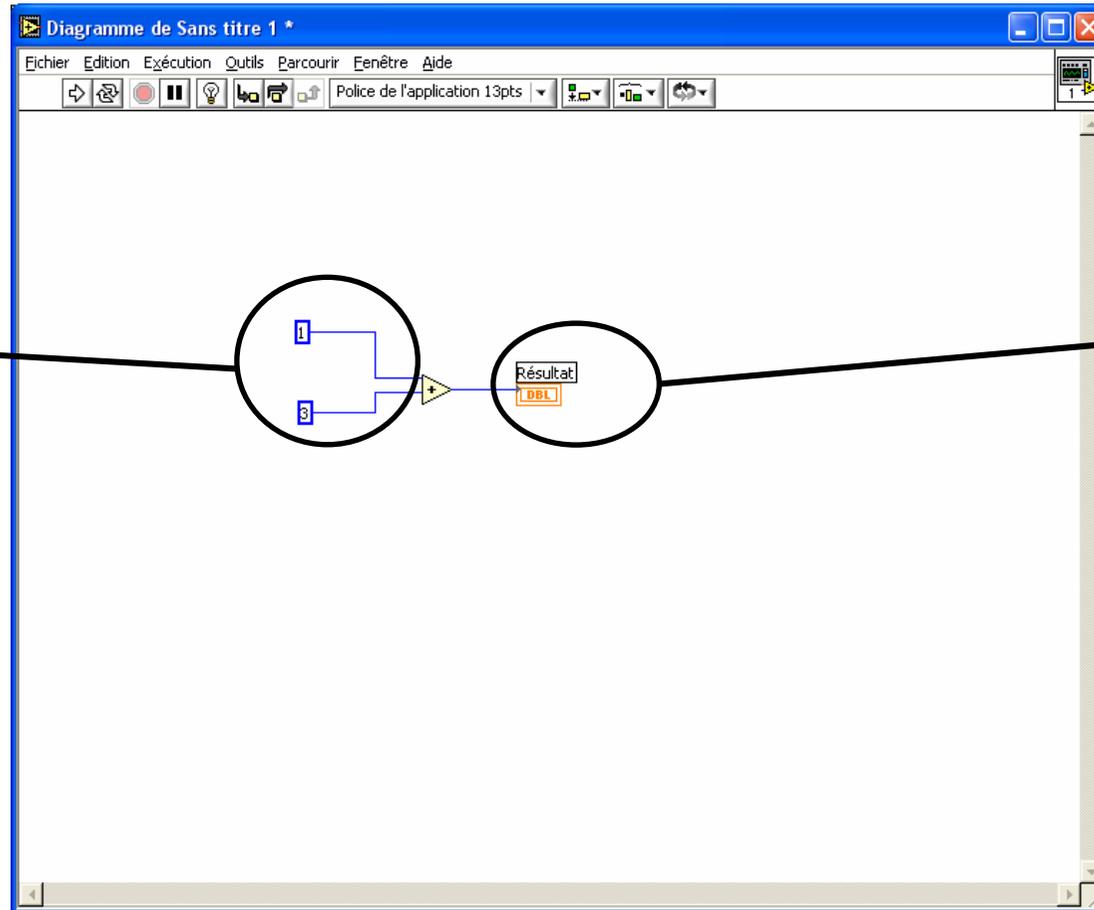


Terminer le câblage de la fonction

- Câbler la deuxième constante à la fonction (+)
- Câbler le terminal de sortie de cette fonction à l'indicateur ' **résultat** '

Le diagramme terminé

Remarquer la couleur bleue des constantes et des fils
Cette couleur indique le type des données (entier)



Remarquer le points gris indiquant le conflit entre les deux types (ce conflit n'implique cependant pas la non-exécution du programme)

Changer la représentation d'un indicateur

Nous allons donc changer le type de l'indicateur : réel double → entier

- Pour cela, faire un <clic droit> sur l'indicateur '**Résultat**'
- Sélectionner le menu 'Représentation'
- Choisir le type '**I32**'



L'indicateur devient bleu

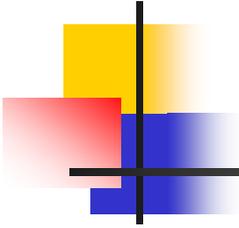
The screenshot shows a context menu for an indicator named 'résultat'. The menu items are:

- Eléments visibles
- Rechercher l'indicateur
- Masquer l'indicateur
- Changer en commande
- Changer en constante
- Description et info-bulle...
- Créer
- Opérations sur les données
- Avancé
- Représentation (highlighted)

The 'Représentation' sub-menu shows the following options:

?	EXT	DBL	SGL
I32	I16	I8	
U32	U16	U8	
CXT	CDB	C5G	

The 'I32' option is highlighted with a blue border.



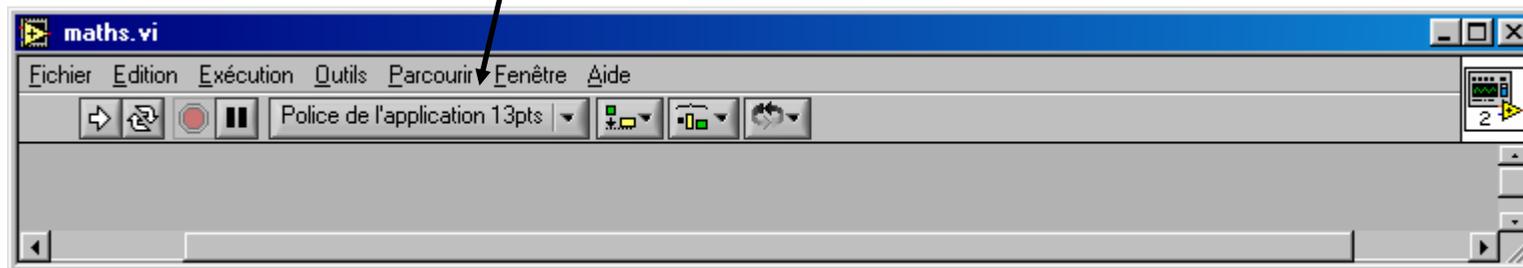
Changer les textes d'étiquettes

- Faire apparaître le menu '*outils*' :
<shift + clic droit>
- faire glisser la souris sur l'outil texte (lettre **A**)
clic gauche
- positionner le curseur dans l'étiquette à modifier,
éditer le texte (les raccourcis habituels sont disponibles)

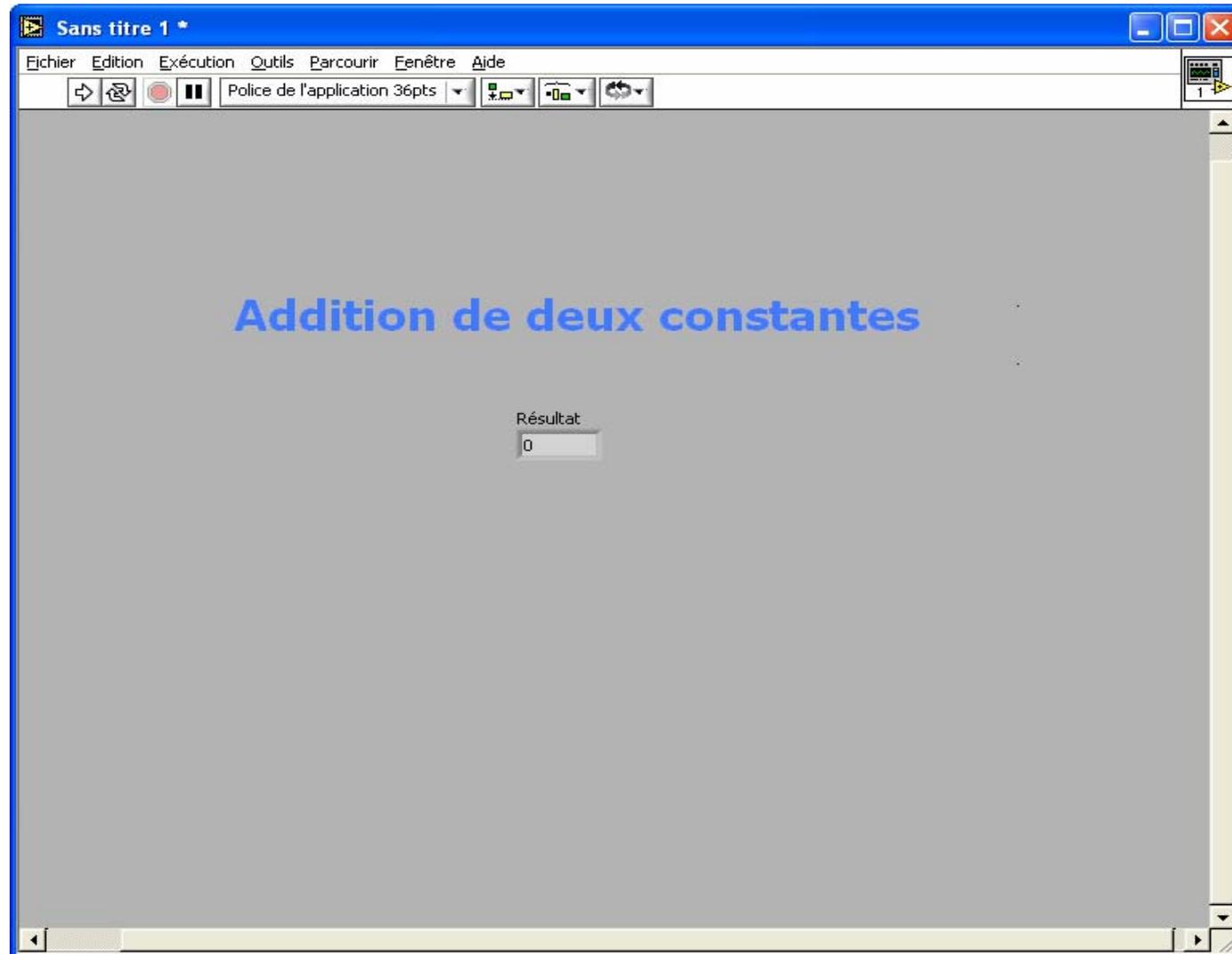
Texte libre

On peut aussi ajouter du texte où l'on veut,
en face avant ou sur le diagramme → commentaires

- Ajouter en face avant du texte :
' Addition de deux constantes '
- Changer la police de caractères :
menu déroulant dans la barre de menu, au milieu

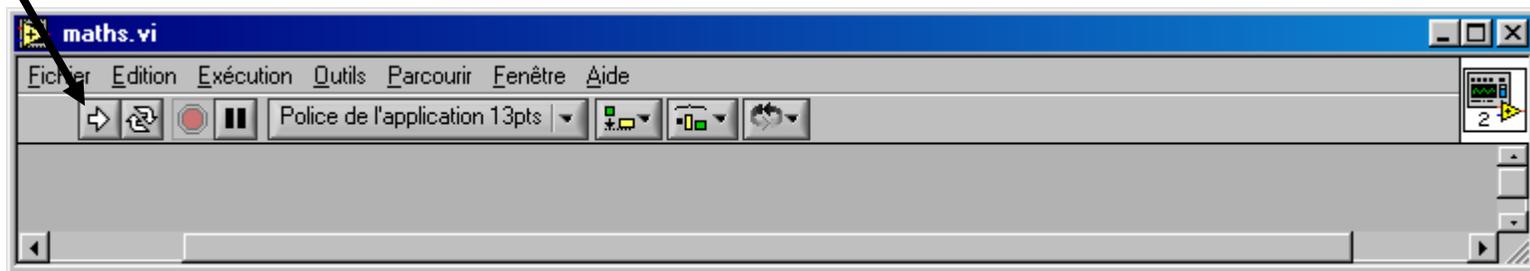


La face avant terminée



Exécution

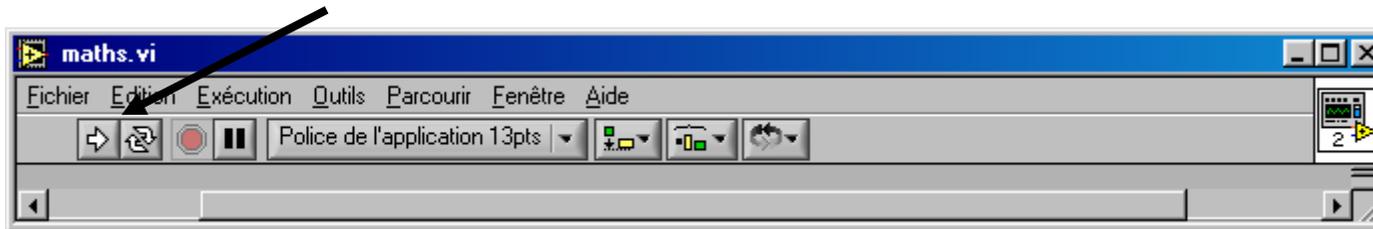
- Changer les valeurs
(outil ' main ' puis taper valeurs ou utiliser les flèches d'incrémentation)
- <clic gauche> sur la flèche blanche dans la barre de menu
(bulle d'aide : ' exécuter ')



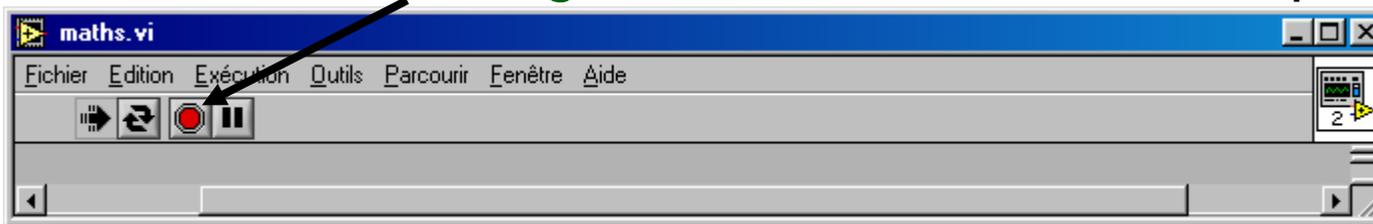
- Les résultats des opérations apparaissent dans les indicateurs

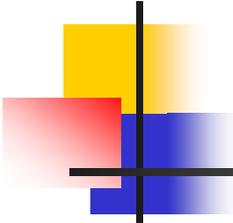
Exécution en continu

<clic gauche> sur la double flèche :



Pour arrêter, <clic gauche> sur le bouton stop :



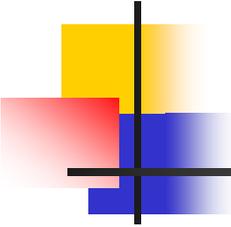


Création d'un VI

Nous allons maintenant compliquer un peu...

Nous allons créer un VI permettant de réaliser plusieurs opérations (+ ; - ; / ; x) sur des nombres réels qui peuvent être changés à volonté.

- Sélectionner : Nouveau VI (Virtual Instrument)
- Sélectionner la fenêtre face avant (grise)
- clic droit → Commandes
- Déplacer la souris sur ' *Numérique* '
- Une sous-fenêtre apparaît avec les commandes numériques



Indicateurs numériques

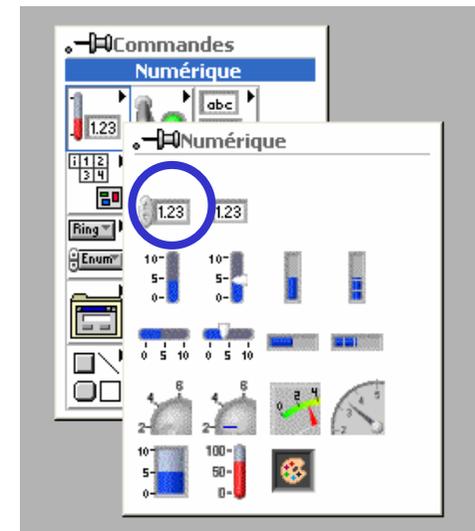
Créer quatre indicateurs numériques
que vous pourrez nommer

- ' somme '
- ' soustraction '
- ' multiplication '
- ' division '

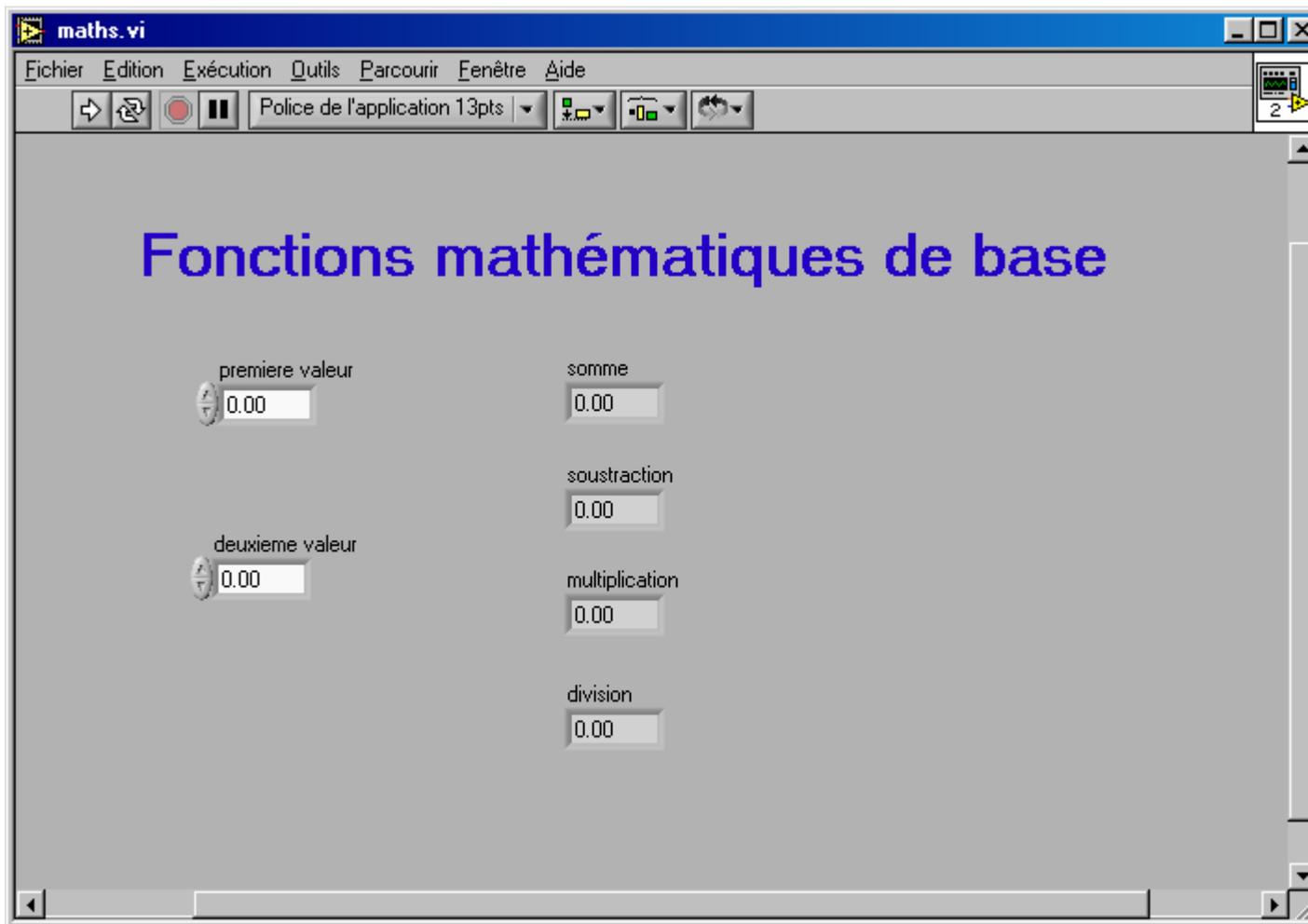
Commandes numériques

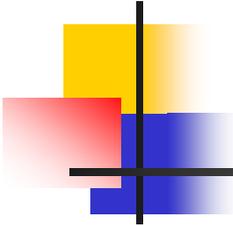
Créer deux commandes numériques
que vous pourrez nommer

- ' première valeur '
- ' deuxième valeur '



La face avant terminée





Terminaux

Deux types de terminaux

- Terminal de commande : **cadre en gras**
- Terminal d'indicateur : cadre fin



Reste à rajouter les fonctions et câbler...

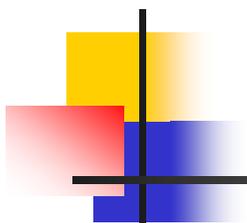
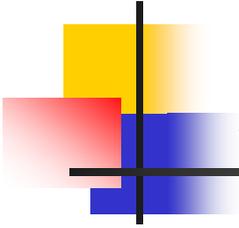


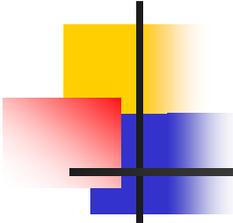
Diagramme (la partie opérationnelle)

- Visualiser le diagramme
 - <ctrl+E> pour passer de face avant ↔ diagramme
- LabVIEW a placé des terminaux sur le diagramme qui sont liés aux objets de la face avant (à part le texte libre)
chaque objet a son terminal
- On peut déplacer ces terminaux où l'on veut sur le diagramme, cela ne change rien en face avant
- En double cliquant
 - sur un objet de face avant → terminal du diagramme
 - sur terminal du diagramme → objet de face avant



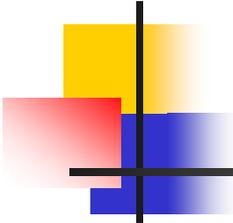
Ajouter les fonctions

- Toujours sur diagramme, <clic droit> → la palette de fonction apparaît
- Déplacer la souris → sur numérique → la palette ' *numérique* ' apparaît avec les fonctions mathématiques
- Cliquer sur la fonction ' *additionner* '
- Déplacer la souris vers le diagramme → on voit la fonction sous le curseur en forme de main
- Placer le curseur à gauche du terminal indicateur ' *somme* '
- <clic gauche> pour poser la fonction



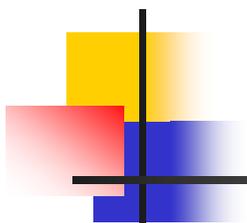
Câblage

- Dans la palette d'outils <shift+clic droit>, sélectionner l'outil de câblage (comme une bobine de fil)
le curseur change de forme (bobine)
- positionner le curseur sur le terminal de commande
' **premiere valeur** ' → il clignote
- <clic gauche>, déplacer la souris → ligne en pointillés
- Amener le curseur sur la fonction ' *additionner* ' (+)
- ...



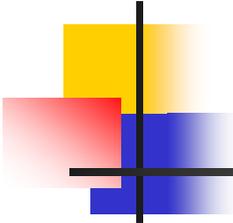
Câblage - suite

- Lorsque le curseur arrive sur la fonction, elle se met à clignoter sous le curseur au niveau des terminaux de connexion
- De courtes lignes de connexion apparaissent aussi
- Convention standard :
 - terminaux d'entrée à gauche
 - terminaux de sortie à droite
- Cliquer sur le connecteur en haut à gauche
- Le câble passe en trait continu orange : câblage valide



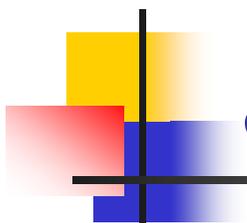
Terminer le câblage de la fonction

- Câbler la commande ' **deuxieme valeur** ' à l'autre terminal d'entrée de la fonction (+)
- Câbler le terminal de sortie de cette fonction à l'indicateur ' **somme** '
- Ajouter trois fonctions supplémentaires : (-), (*) et (/), en face des indicateurs correspondants



Câblage - directions

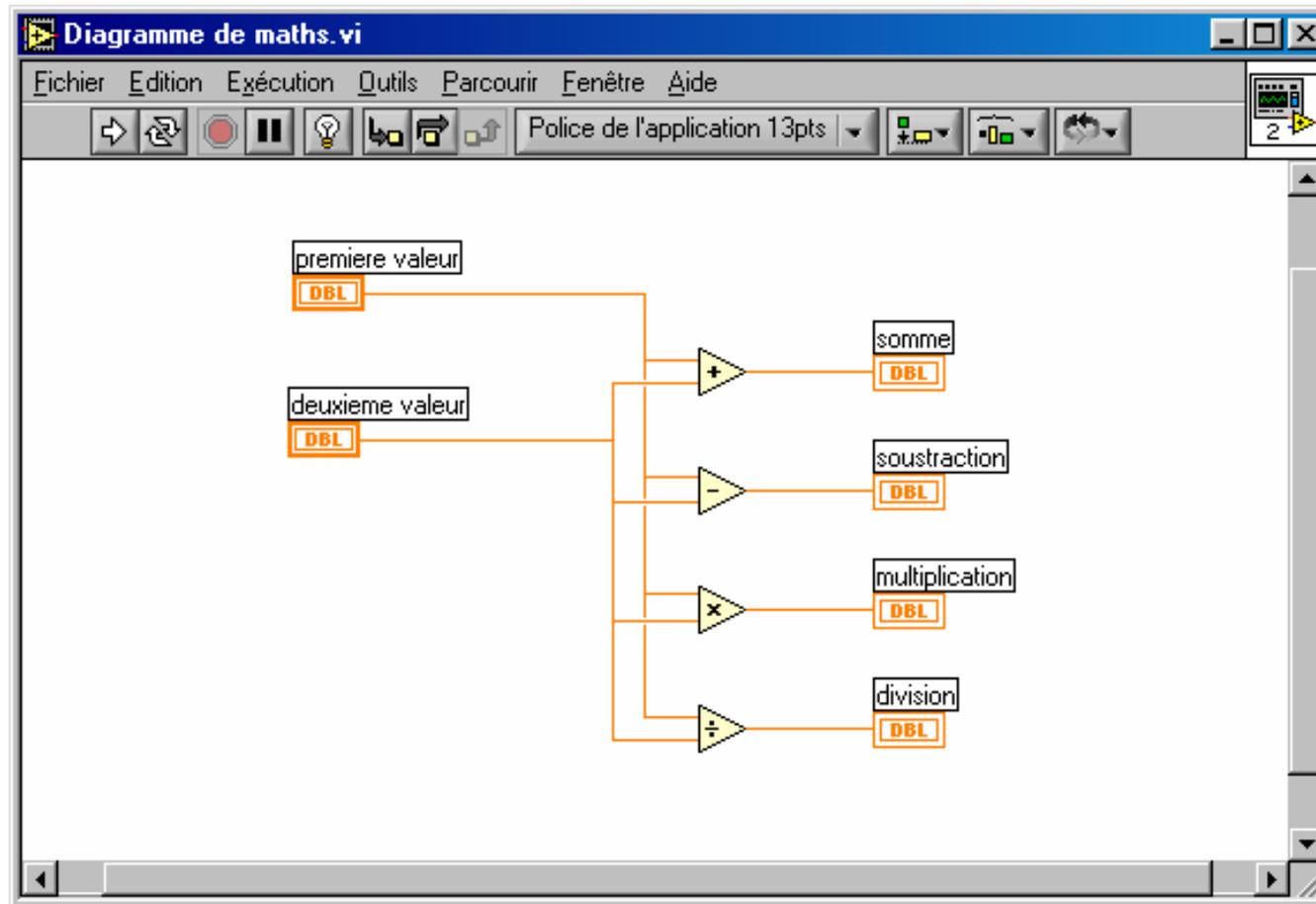
- Les câbles vont toujours horizontalement ou verticalement, jamais en diagonale
- Par défaut, il y a un changement de direction par câble, mais
 - On peut changer la direction initiale (horizontale ou verticale) en appuyant sur `<barre d'espace>`
 - On peut imposer des points intermédiaires par `<clic gauche>`

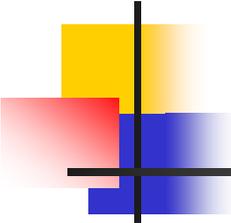


Raccordement aux câbles existants

- Avec l'outil de câblage, cliquer (gauche) sur le câble reliant la commande ' **premiere valeur** ' à la fonction (+) (il clignote lorsqu'on est dessus)
- Descendre jusqu'au terminal d'entrée du haut de la fonction (/)
- Recommencer avec la commande ' **seconde valeur** '
- Les câbles qui se croisent se voient au blanc qui est laissé de part et d'autre
- Terminer les câblages

Le diagramme terminé



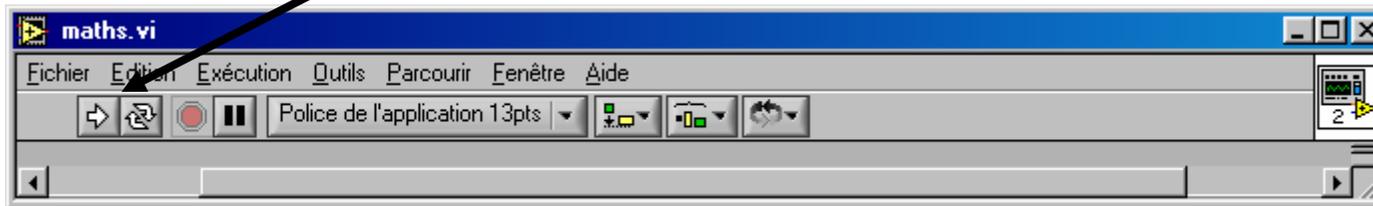


Sélectionner les câbles

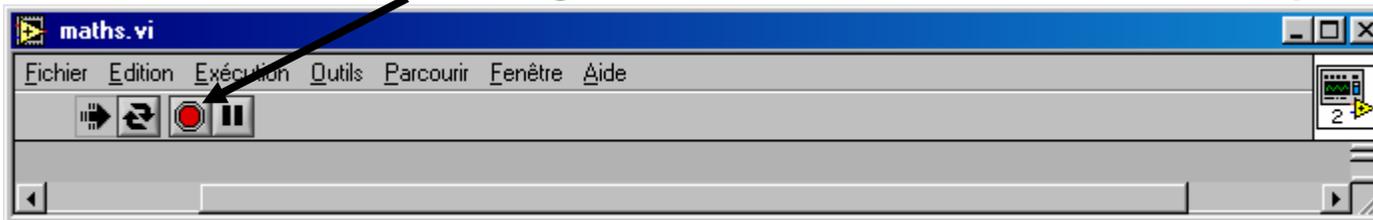
- Dans le diagramme
outil : sélectionner (flèche)
Rq: on peut aussi utiliser <barre espace> pour passer de l'outil bobine à l'outil sélection
- Essayer simple, double et triple clic gauche sur un fil (câble)
- Une fois sélectionné on peut le déplacer en utilisant
 - souris
 - flèches du clavier ou <shift + flèche> (plus rapide)
- Utiliser l'outil bobine pour modifier les câblages
- Clarifier le diagramme

Exécution en continu

<clic gauche> sur la double flèche :

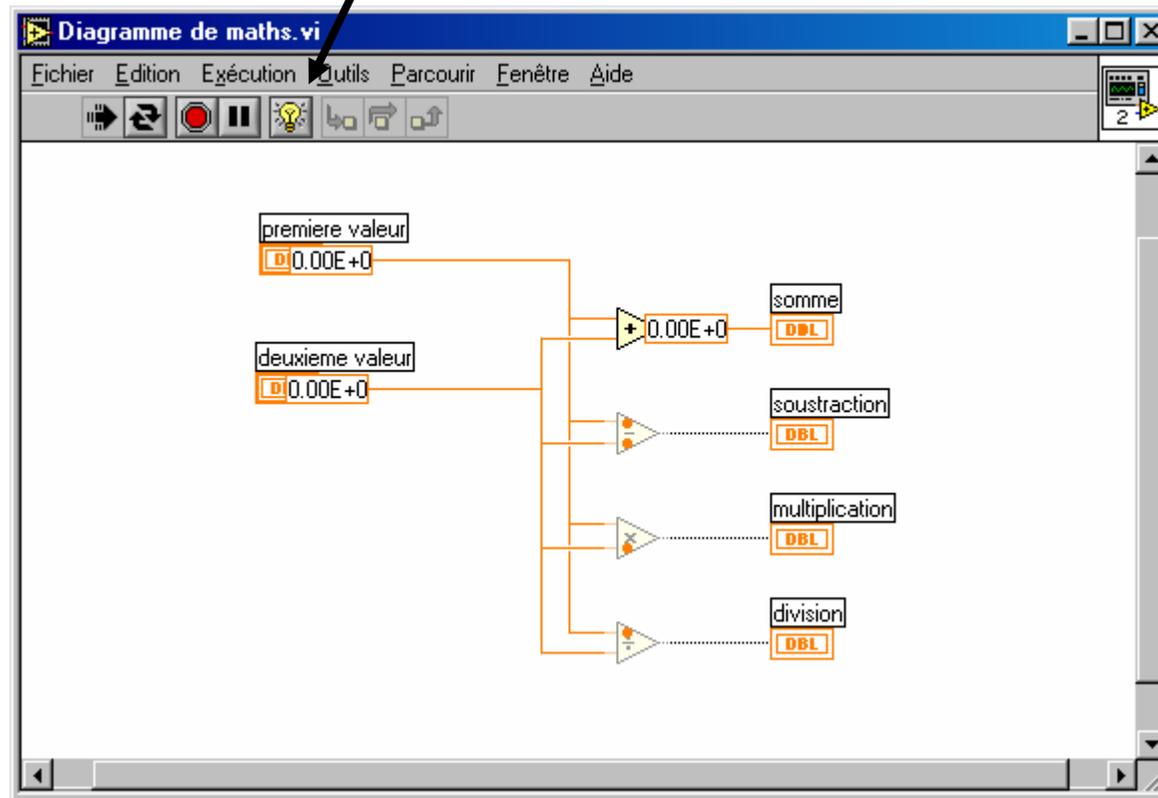


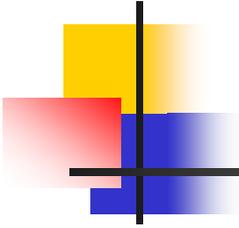
Pour arrêter, <clic gauche> sur le bouton stop :



flot de données

<ctrl-E> → diagramme, puis
cliquer sur ' *animer l'exécution* ', puis exécuter en continu



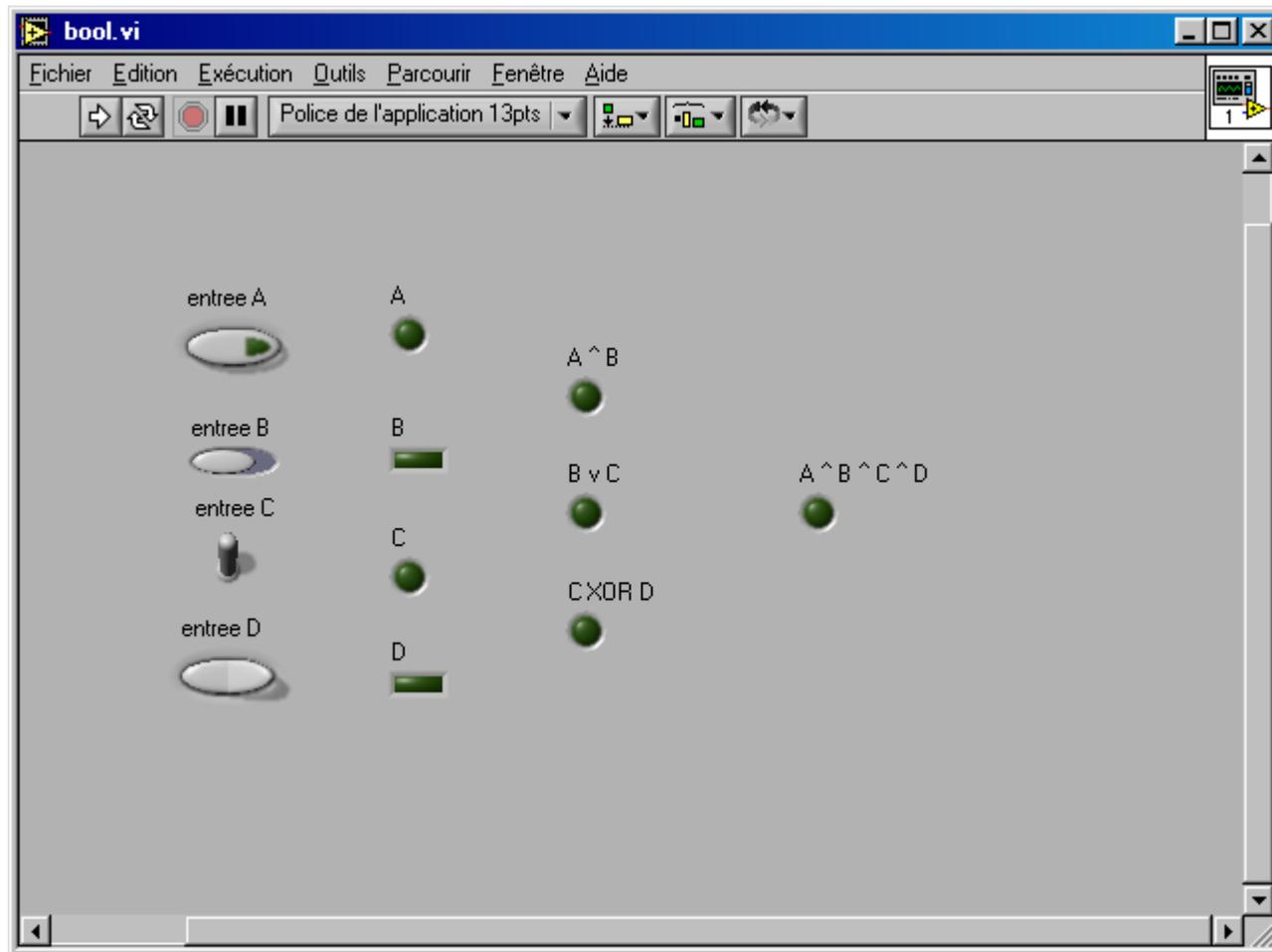


Opérations booléennes

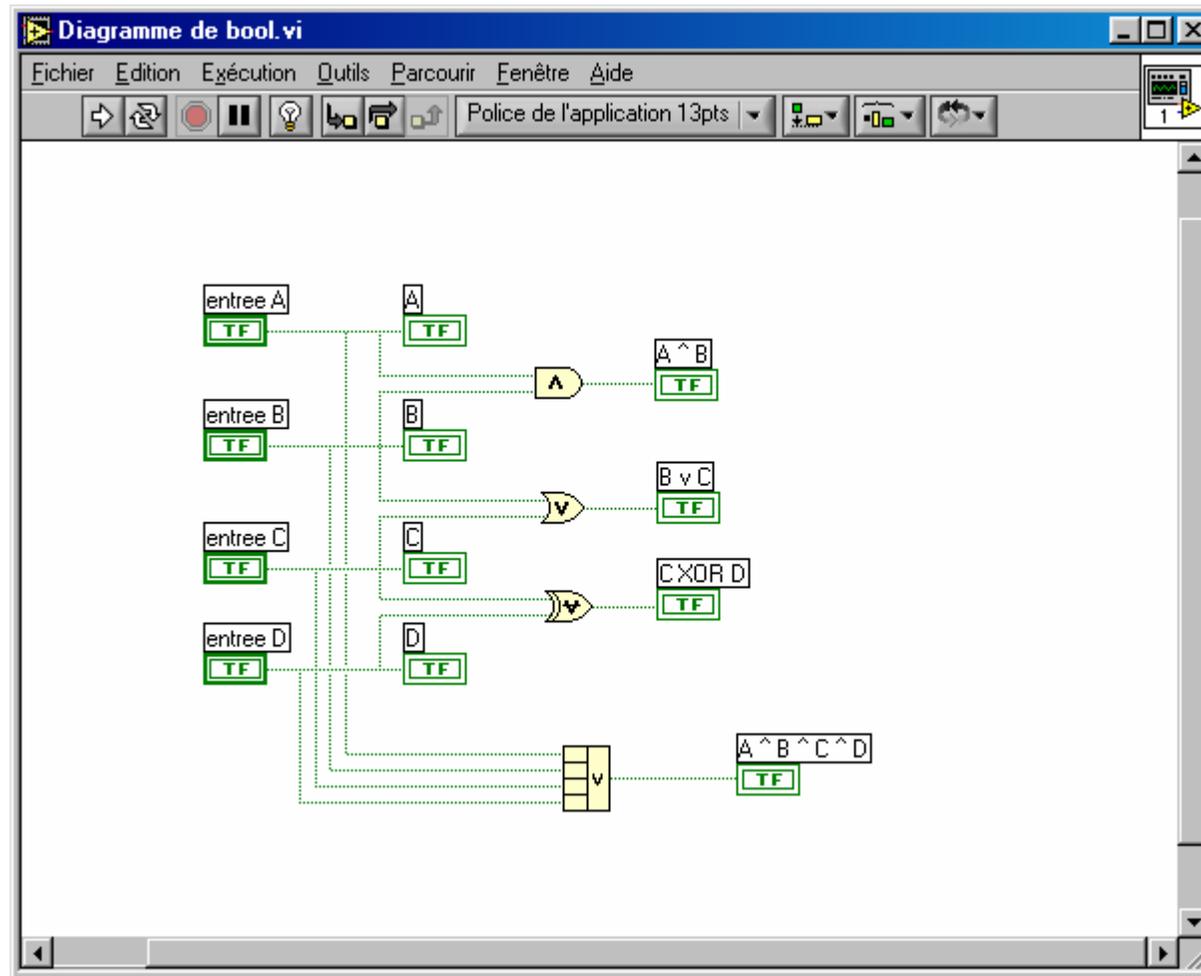
Réaliser la face avant et le diagramme d'un VI montrant quelques opérations booléennes :

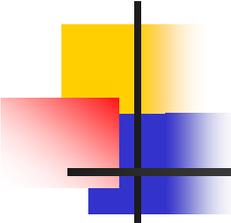
- ET
- OU
- OU exclusif (XOR)
- opération composée (plusieurs entrées)

Booléens - face avant



Booléens - diagramme

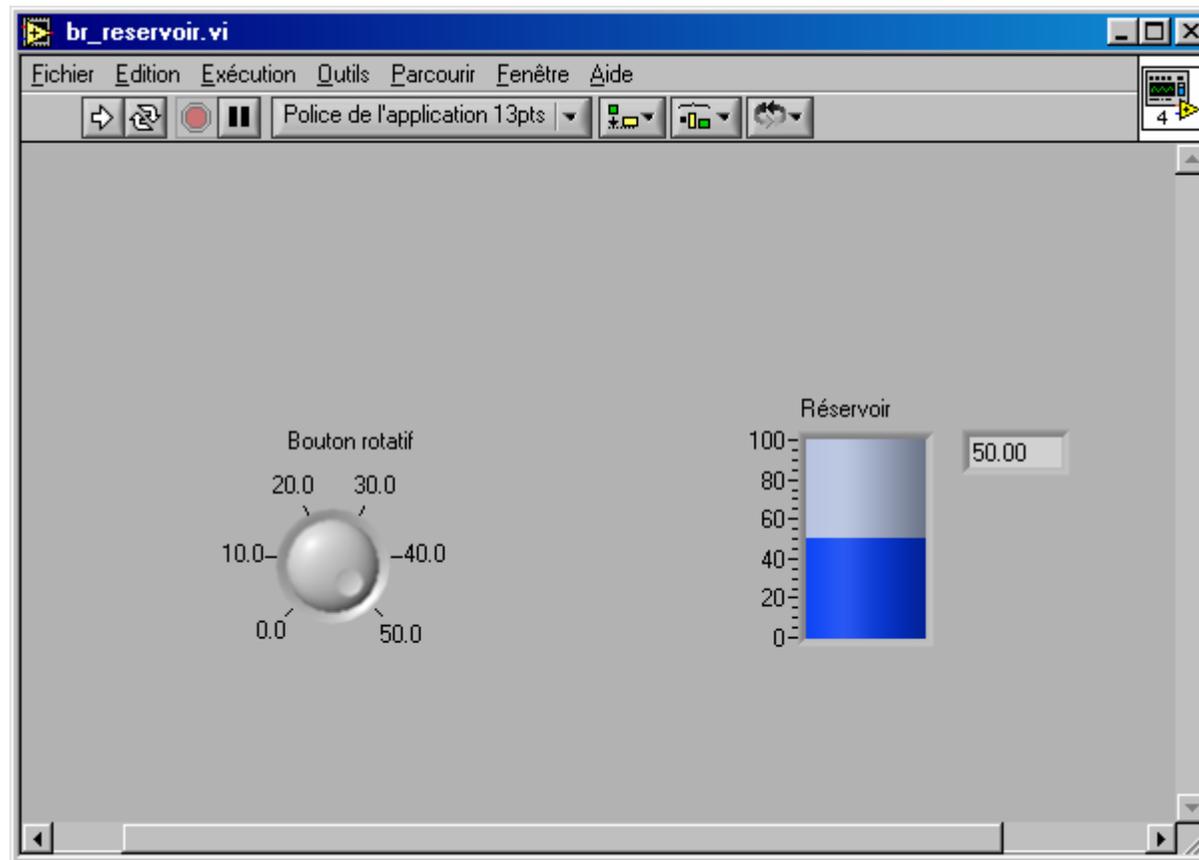




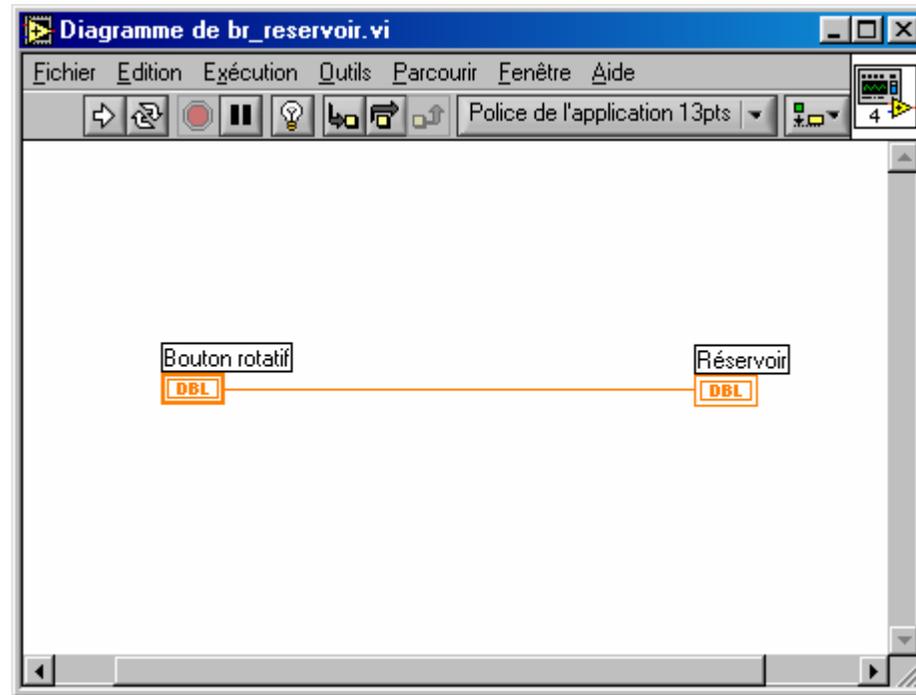
Changements d'échelle

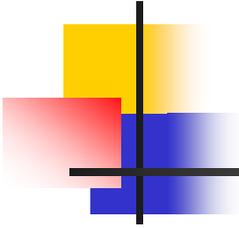
- créer un nouveau VI
- Ajouter une commande numérique ' *bouton rotatif* '
- Ajouter un indicateur numérique ' *réservoir* '
- <clic droit> sur le réservoir ; sélectionner ' *afficheur numérique* ' dans le menu ' *éléments visibles* '
- relier le bouton rotatif au réservoir
- Changer les échelles (outil ' *main* ' ou ' *texte* ')
éditer les valeurs min ou max affichées → l'échelle change
- Explorer les menus <clic droit>

Face avant



Diagramme



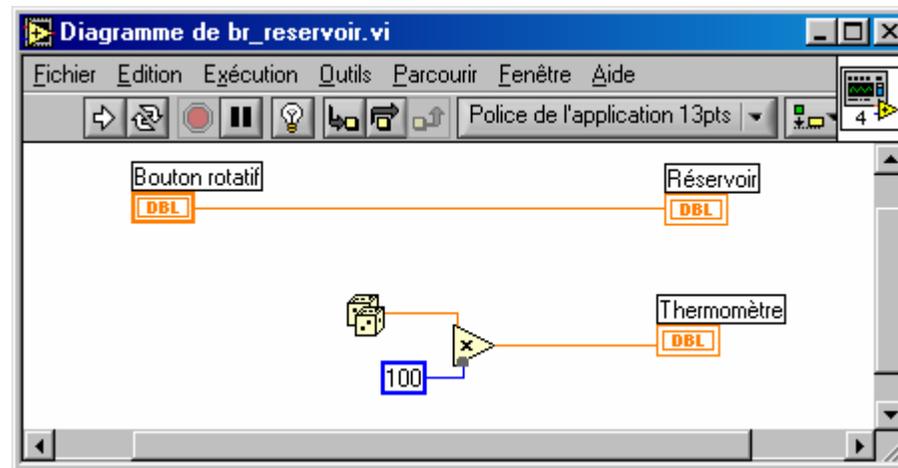


Ajouter une jauge de température

- Ajouter un thermomètre (indicateur numérique)
- Ajouter
 - un générateur de nombres aléatoires,
 - une fonction ' *multiplier* '
 - une constante numérique
- Changer la valeur de la constante à 100

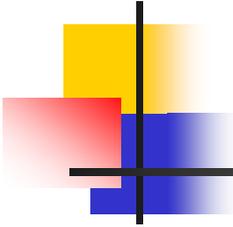
Ajouter une jauge de température

Relier les éléments de façon à multiplier par 100 la sortie du générateur aléatoire (entre 0 et 1) :



Noter les différentes couleurs

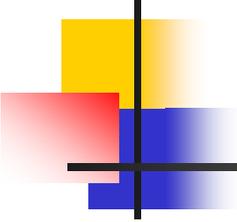
<clic droit> sur la constante, représentation → DBL
→ changement couleur



Types de données

Comme dans des langages de programmation classiques, différents types de données, par exemple :

- Entiers (bleu)
 - mot long (32), mot (16), octet (8)
- Réels (orange)
 - précision étendue (64), double (32), simple (16)
- Booléens (vert)
- Chaînes (magenta)

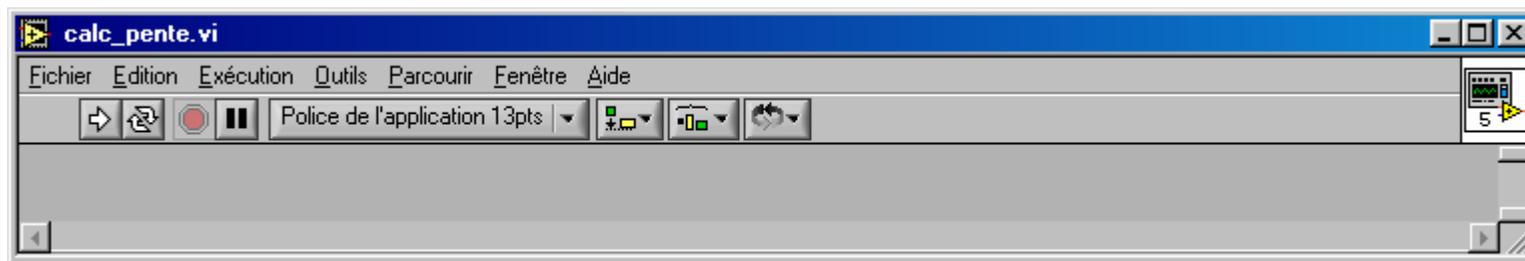


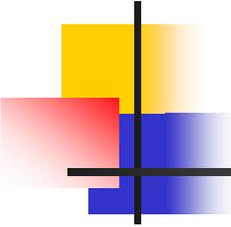
Sous VIs

- Équivalent des fonctions avec
 - entrées multiples
 - sorties multiples
- Aident à alléger les diagrammes, à les rendre plus lisibles
- Peuvent être utilisés plusieurs fois dans le même VI

Exemple de sousVIs

- Créer un nouveau VI : calcul de pente entre deux points
- On peut le documenter :
Fichier → propriétés du VI → catégorie: documentation
- Éditer l'icône : < clic droit > sur l'icône en haut à droite



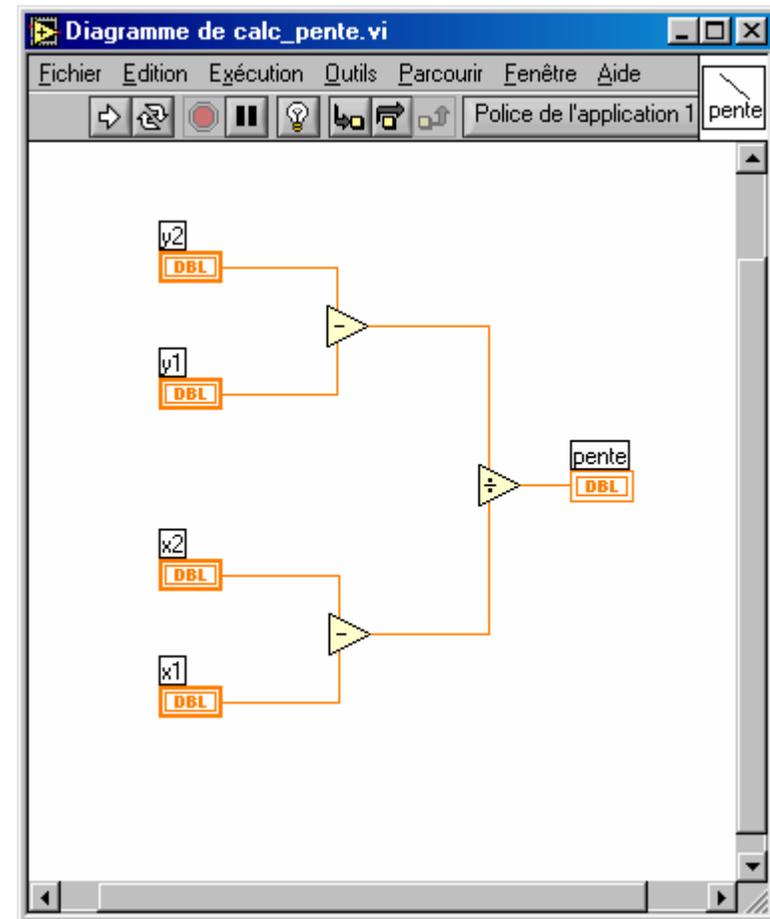
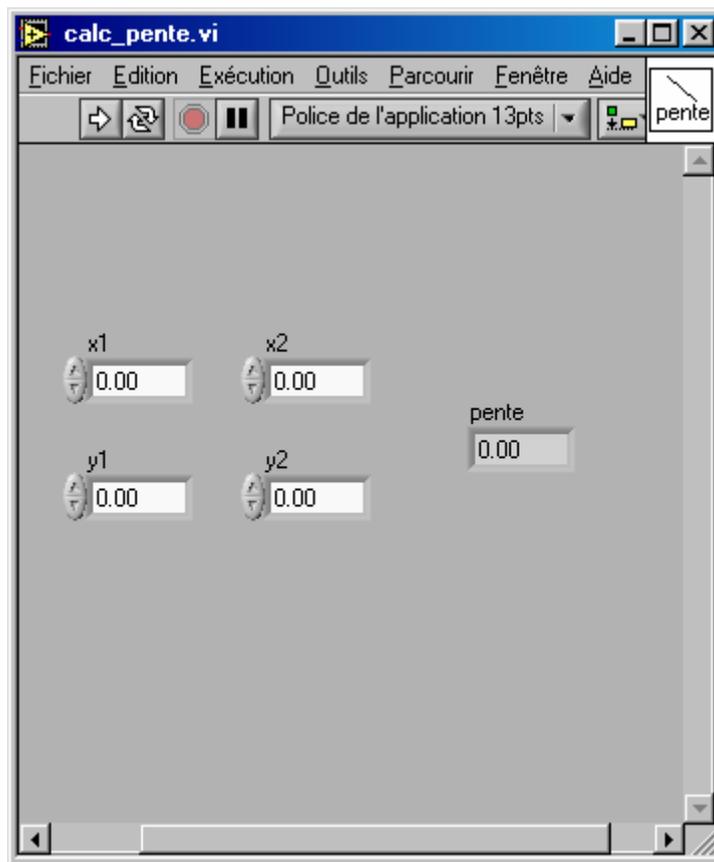


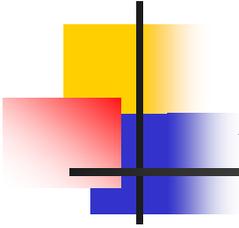
Éditer l'icône

- utiliser les outils pour dessiner l'icône
- on peut utiliser la couleur
- dessiner en 256 couleurs puis faire copier à partir de 256 couleurs pour noir et blanc et 16 couleurs

Calcul de pente

créer la face avant et le diagramme :



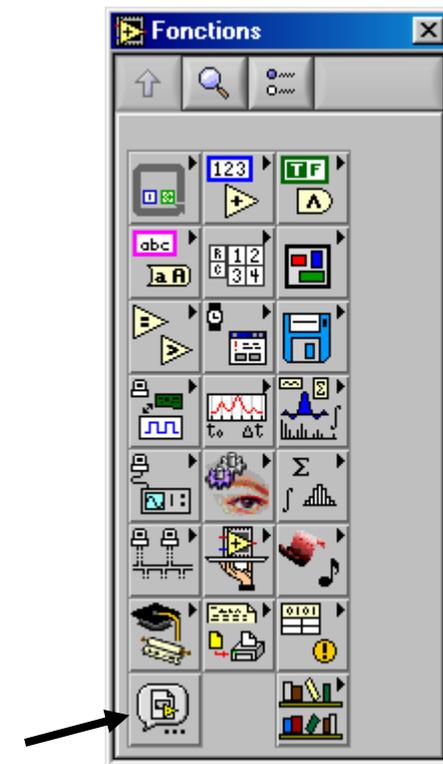


Ajouter les terminaux au connecteur

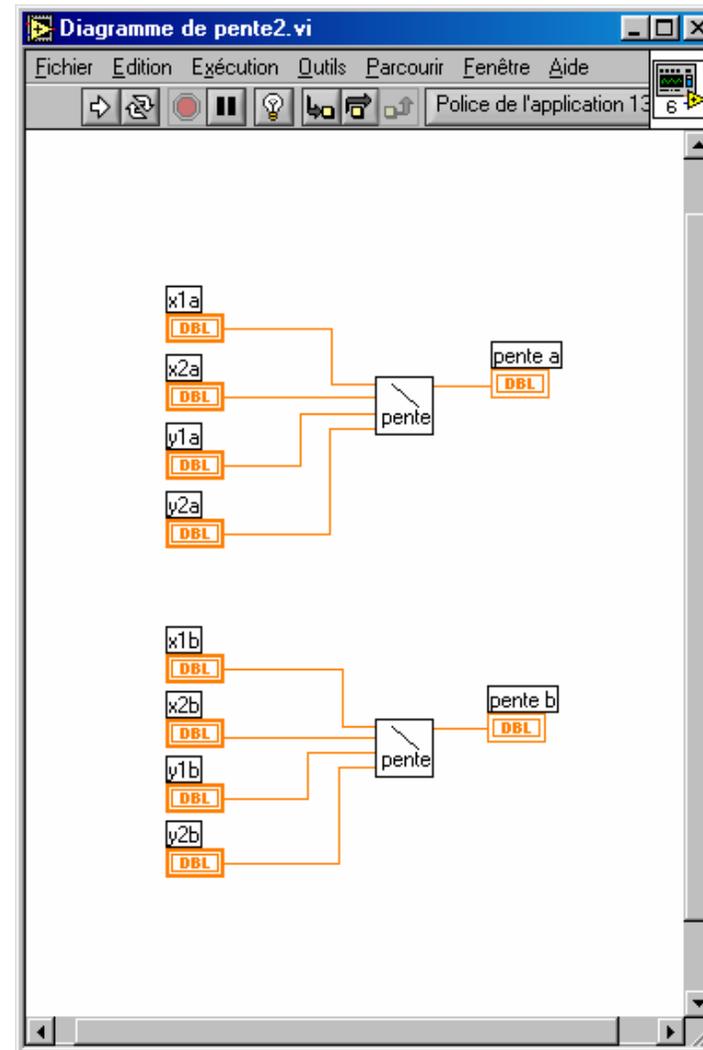
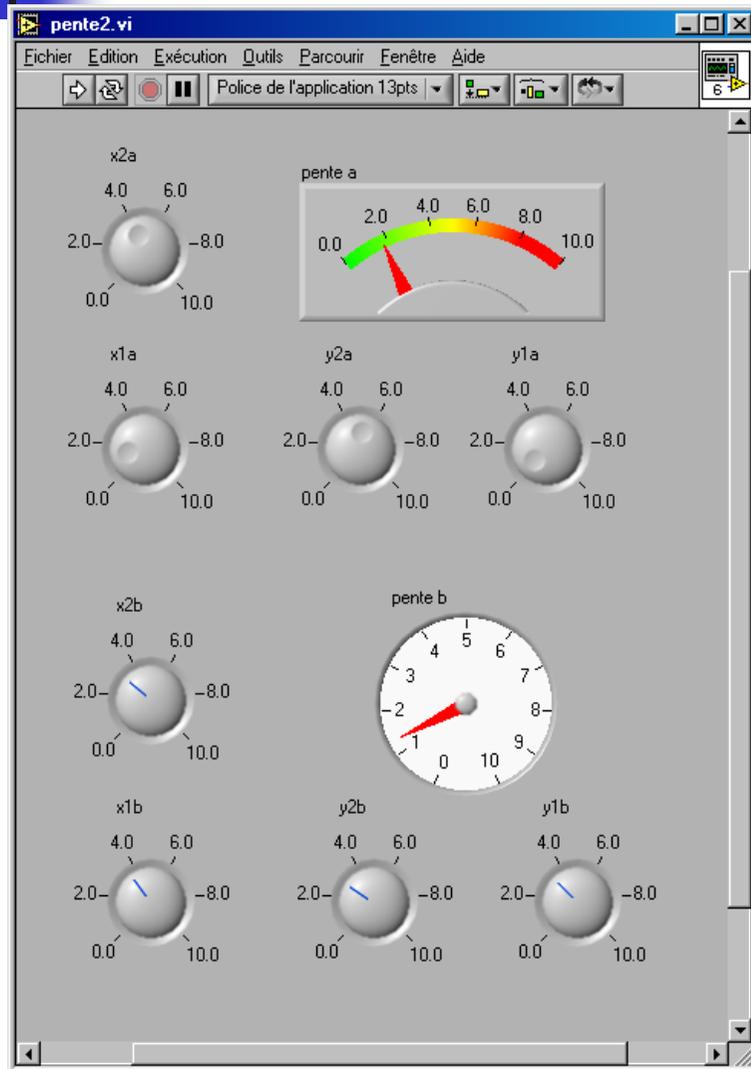
- terminaux d'entrée (à gauche) ↔ paramètres d'entrée
- terminaux de sortie (à droite) ↔ valeurs de sortie
- <clik droit> sur carré icône *face avant* → visualiser le connecteur
- choisir un modèle avec assez d'entrées et de sorties
- outil bobine → cliquer sur un terminal face avant, puis sur le terminal connecteur souhaité (ou inversement)
- sauvegarder le VI (par exemple: ' **calc_pente.vi**')

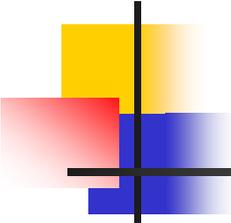
Utilisation de sousVis

- pour utiliser un sousVi, dans le diagramme, choisir dans la palette de fonctions ' *sélectionner un VI* '
- choisir le sousVI, ouvrir
- le déposer sur le diagramme
- on peut le relier aux autres éléments
- on peut avoir une vue d'ensemble du sousVI avec l'aide en ligne <ctrl-H>



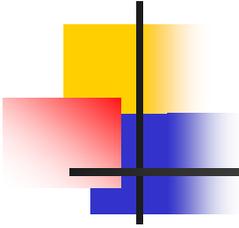
ré-utilisation de sousVIs





Fonction ' sélectionner '

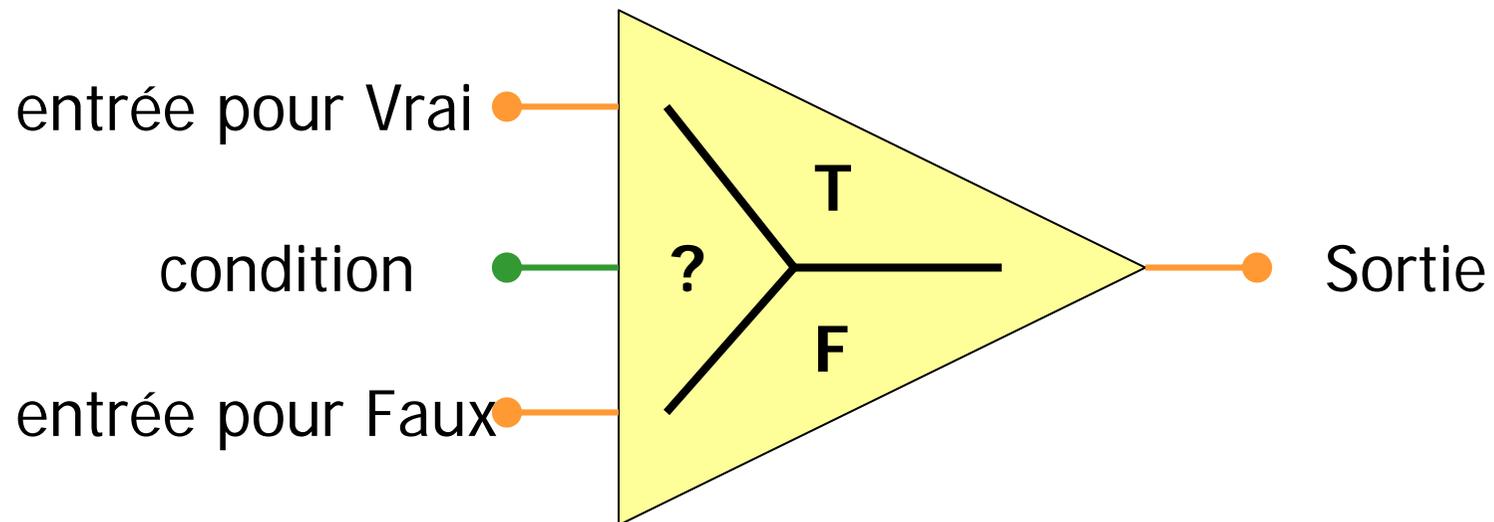
- c'est l'équivalent de ' `cond ? vrai : faux` ' en C/C++
- Créer un nouveau VI
- Dans le diagramme, importer ' `capteur_pression.vi` ' qui fournit une pression lue en hPa
- Ajouter un vumètre sur la face avant, le nommer ' `pression` '
- Relier la sortie du capteur au vumètre
- Observer l'exécution en continu



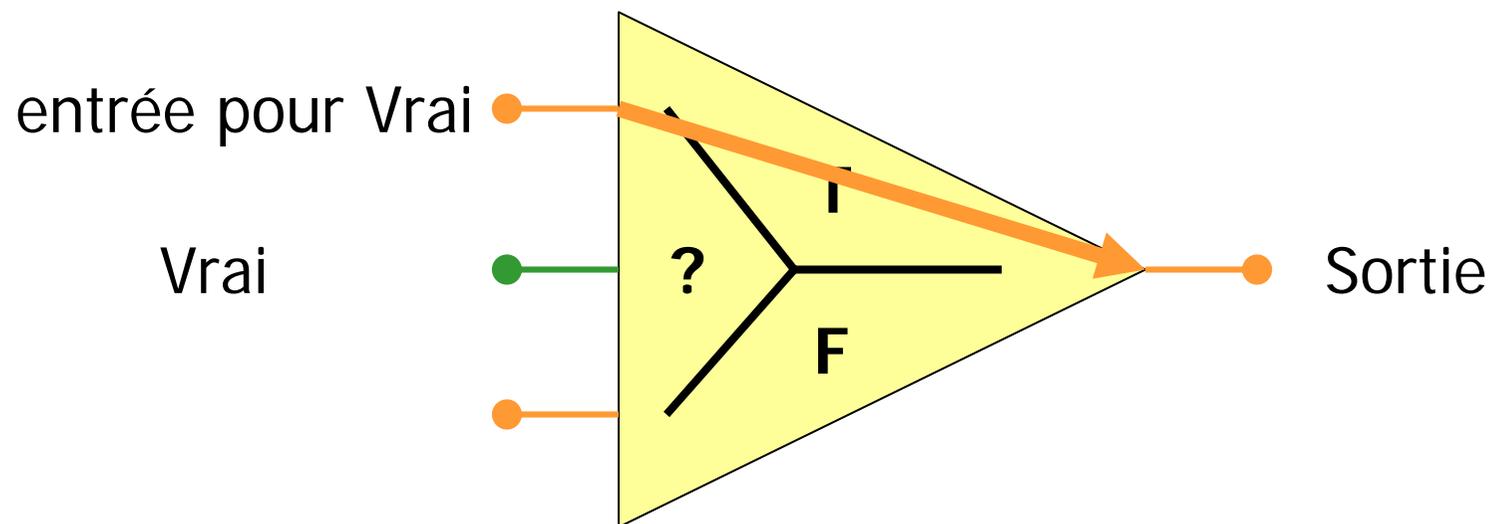
fonction ' sélectionner '

- Remarquer que le curseur est bloqué au maximum
- Adapter l'échelle (pour se guider on peut utiliser l'afficheur numérique)
- On pourrait vouloir choisir entre un affichage en hPa ou en mmHg

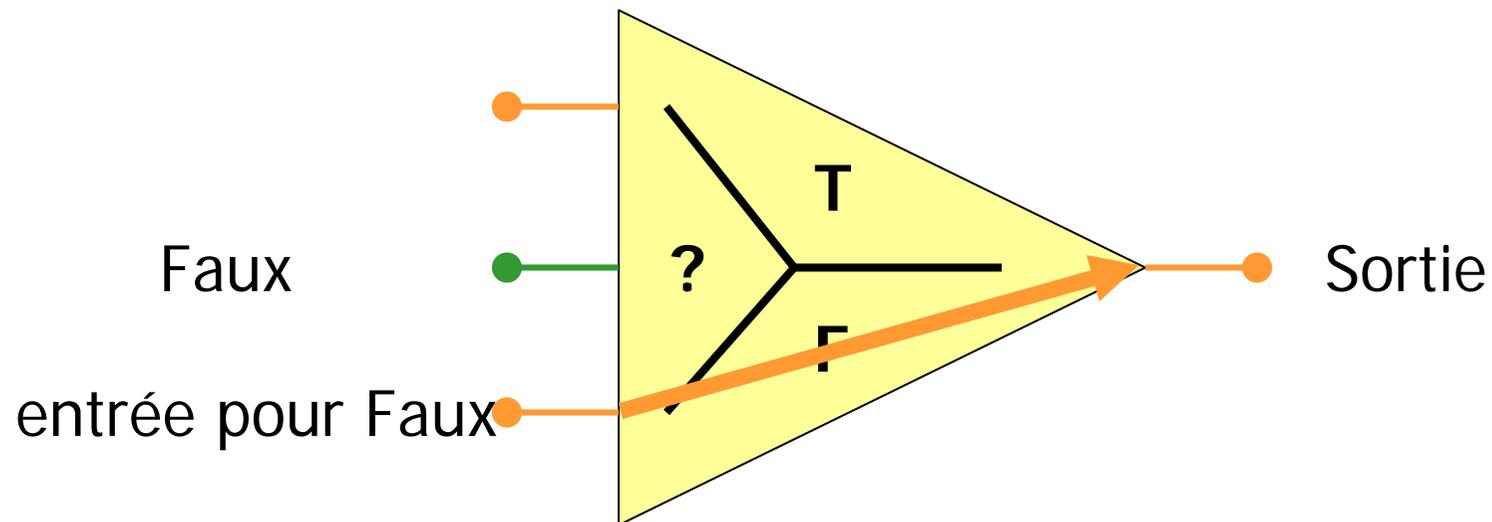
fonction ' sélectionner '

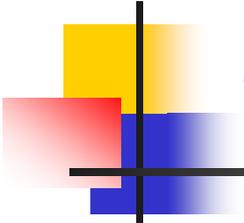


Chemin des données si Vrai



Chemin des données si Faux

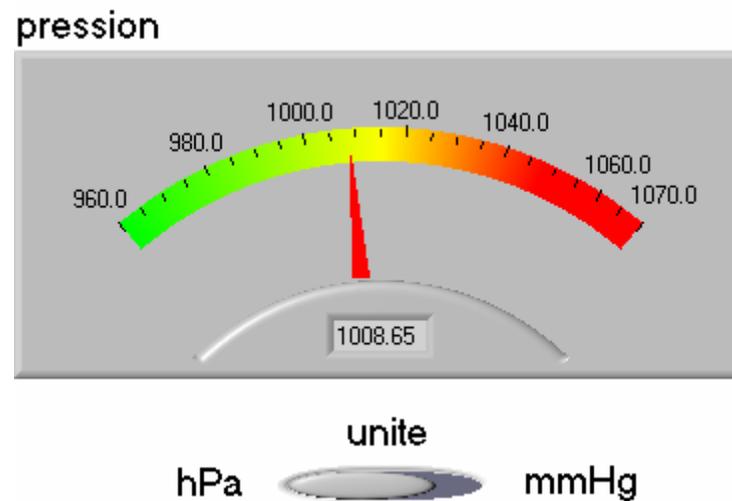


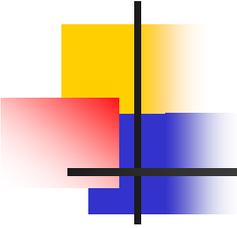


' Sélectionner ' - face avant

- Ajouter un interrupteur horizontal (' unite ') de la palette booléens
- Ajouter deux textes libres ' hPa ' et ' mmHg '
- Agrandir le vumètre et l'interrupteur
- Changer la taille de caractères des textes
agrandir : < ctrl+ = >
diminuer : < ctrl+ - >

' Sélectionner ' - face avant

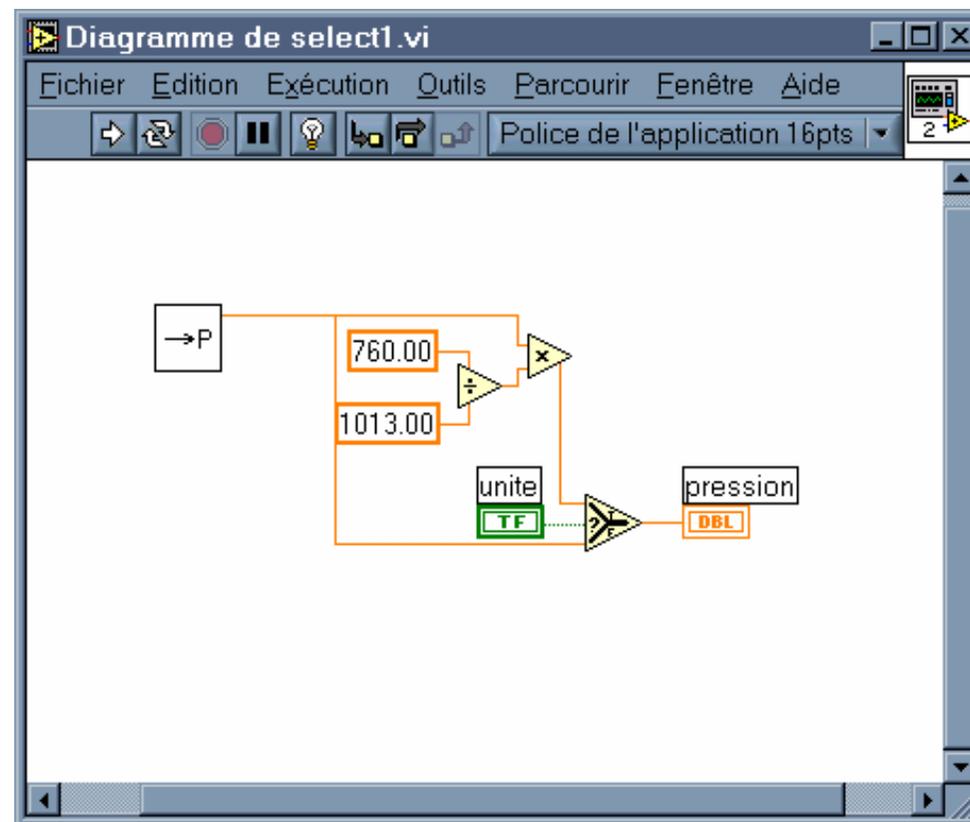




' Sélectionner ' - diagramme

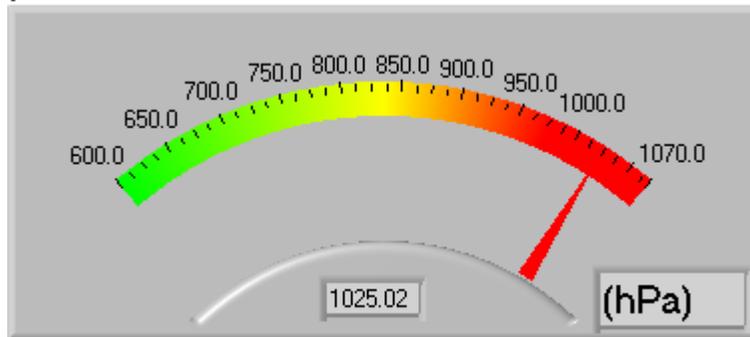
- Ajouter une fonction ' *sélectionner* ' (palette ' *comparaison* ')
- Relier la commande booléenne ' unite ' à l'entrée ' *selecteur* '
- Relier la sortie pression à l'entrée ' *cas faux* '
- diviser 732 par 1013 et multiplier la pression par le résultat, relier la sortie à l'entrée ' *cas vrai* '
- exécuter en continu et changer d'unité

' Sélectionner ' - diagramme

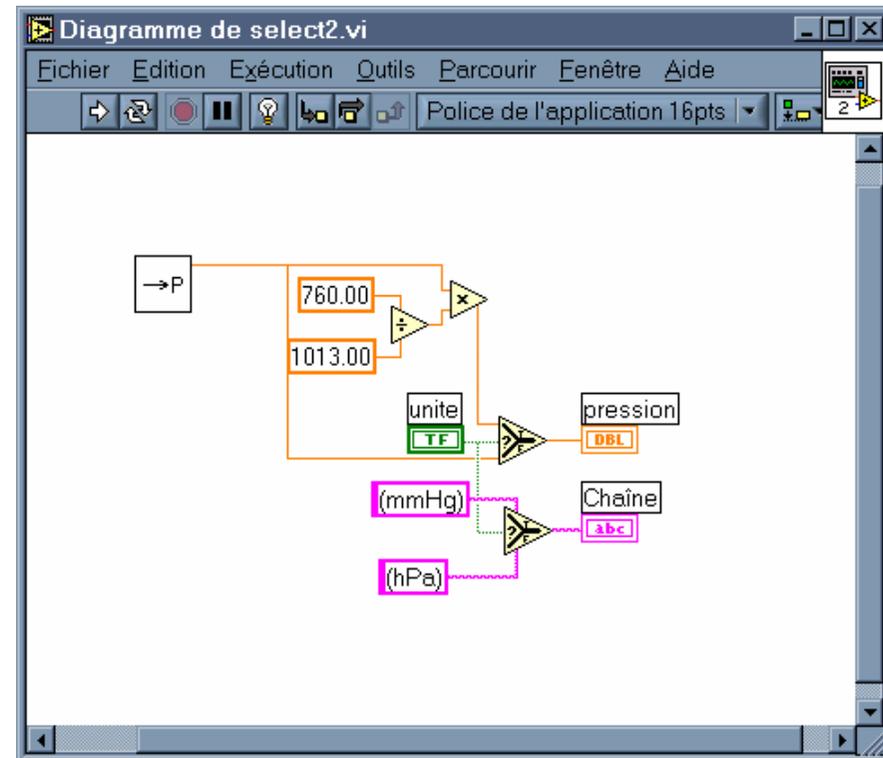


' sélectionner ': affichage unités

pression

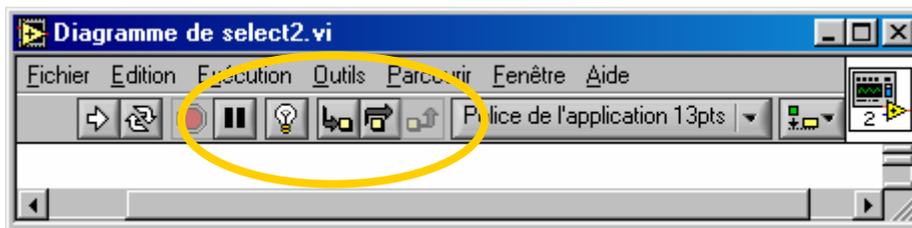


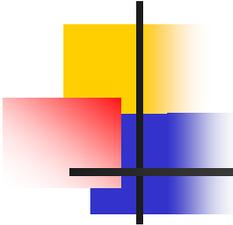
hPa unite mmHg



LabVIEW - debugage

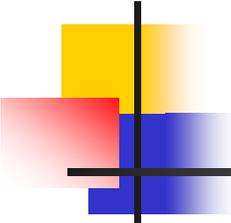
- ' *animer l'exécution* '
- indicateurs intermédiaires (<clic droit> → créer → indicateur)
- outil ' *sonde* '
- outil ' *point d'arrêt* '
- mode pas à pas (en entrant dans les sousVIs ou non)





Trace pression

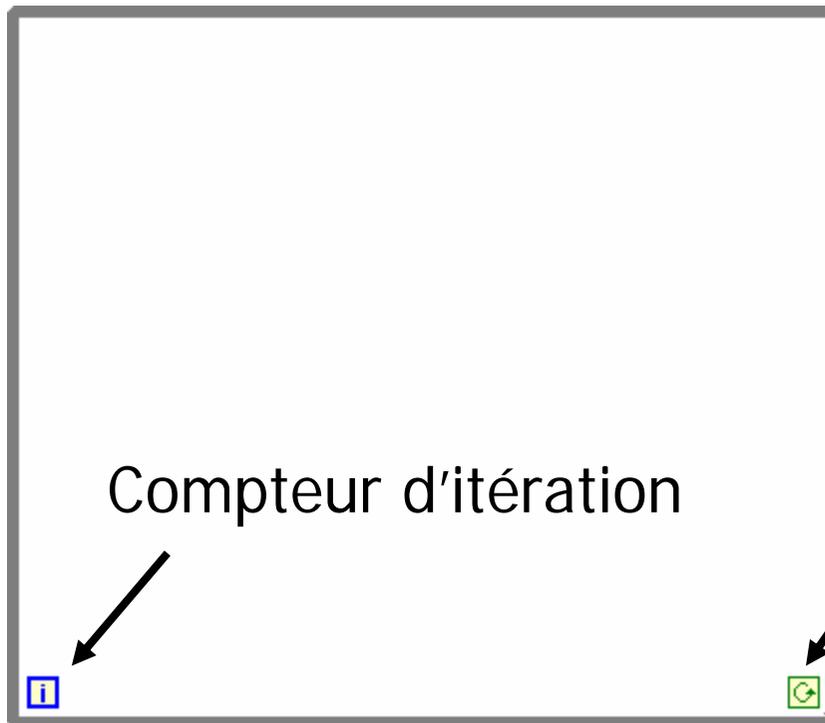
- objectif: tracer au cours du temps la pression
- créer un nouveau VI ' **trace_pression.vi** '
- sur la face avant, poser
 - un bouton ' **STOP** '
 - un graphe déroulant (palette ' *graphe* ')



LabVIEW - boucle while

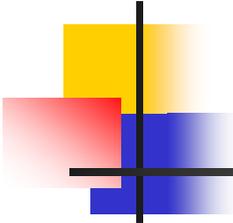
- Placer sur le diagramme une boucle while (palette '*structures*')
- L'étirer de façon à contenir les éléments nécessaires
- La boucle while continuera jusqu'à ce que son terminal conditionnel reçoive '*faux*'
- Elle s'exécute au moins une fois (c'est plutôt une boucle '*repeat until*')
- Elle contient un compteur d'itération

LabVIEW - boucle while



Terminal conditionnel
choix (<clik droit>) :

- arrêter si VRAI
- arrêter si FAUX

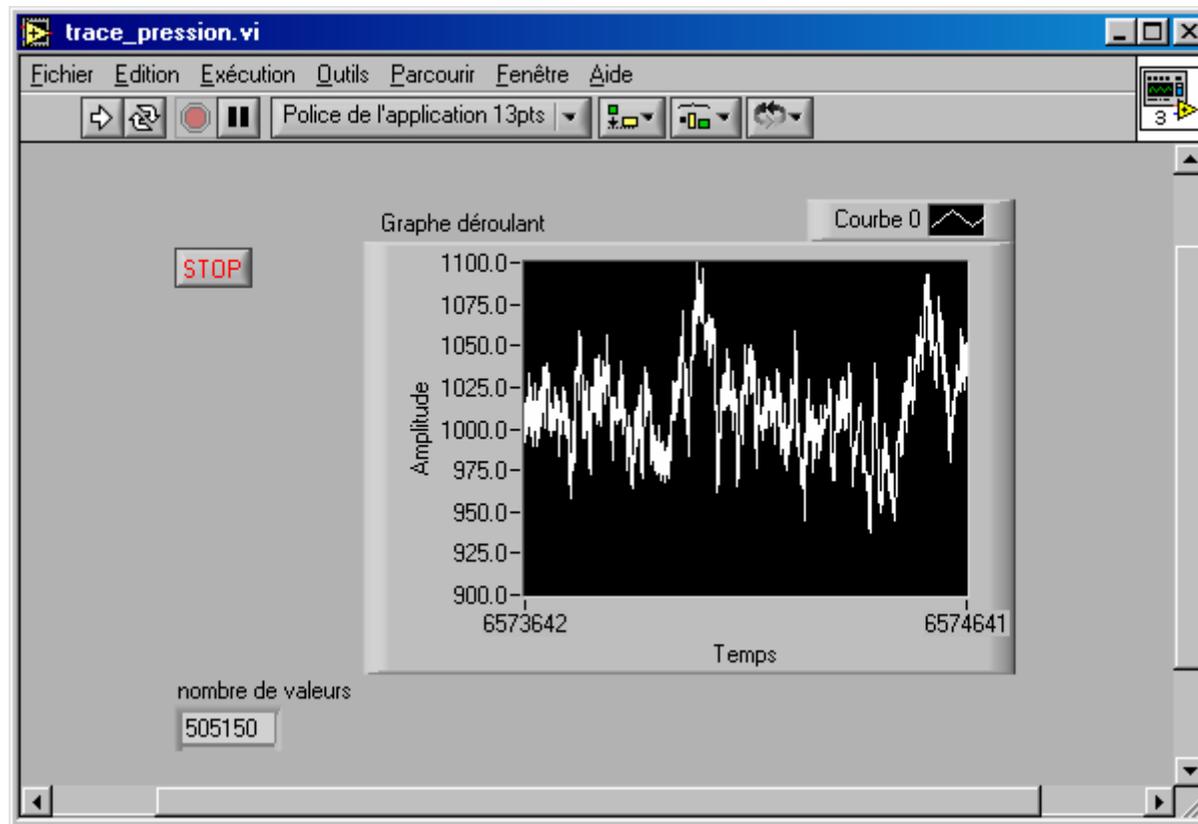


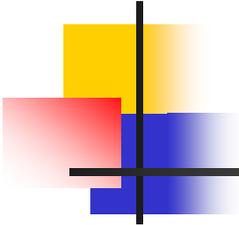
Trace pression

- Dans la boucle while, poser le sousVI ' **capteur_pression.vi** '
- Relier sa sortie au graphe déroulant
- Relier le bouton STOP au terminal conditionnel de la boucle
- Configurer ' **arrêter sur condition vraie** '
- Créer un indicateur numérique ' **nombre de valeurs** ',
à relier au compteur d'itérations
(essayer <**clic droit**> sur le compteur → créer un indicateur

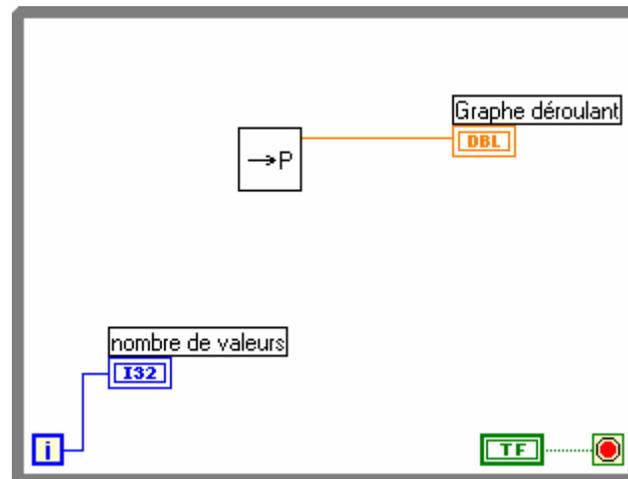
- Observer l'exécution (une fois)
très rapide !
- ajuster l'échelle

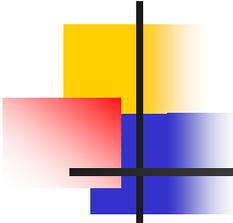
Trace pression -face avant





Trace pression -diagramme

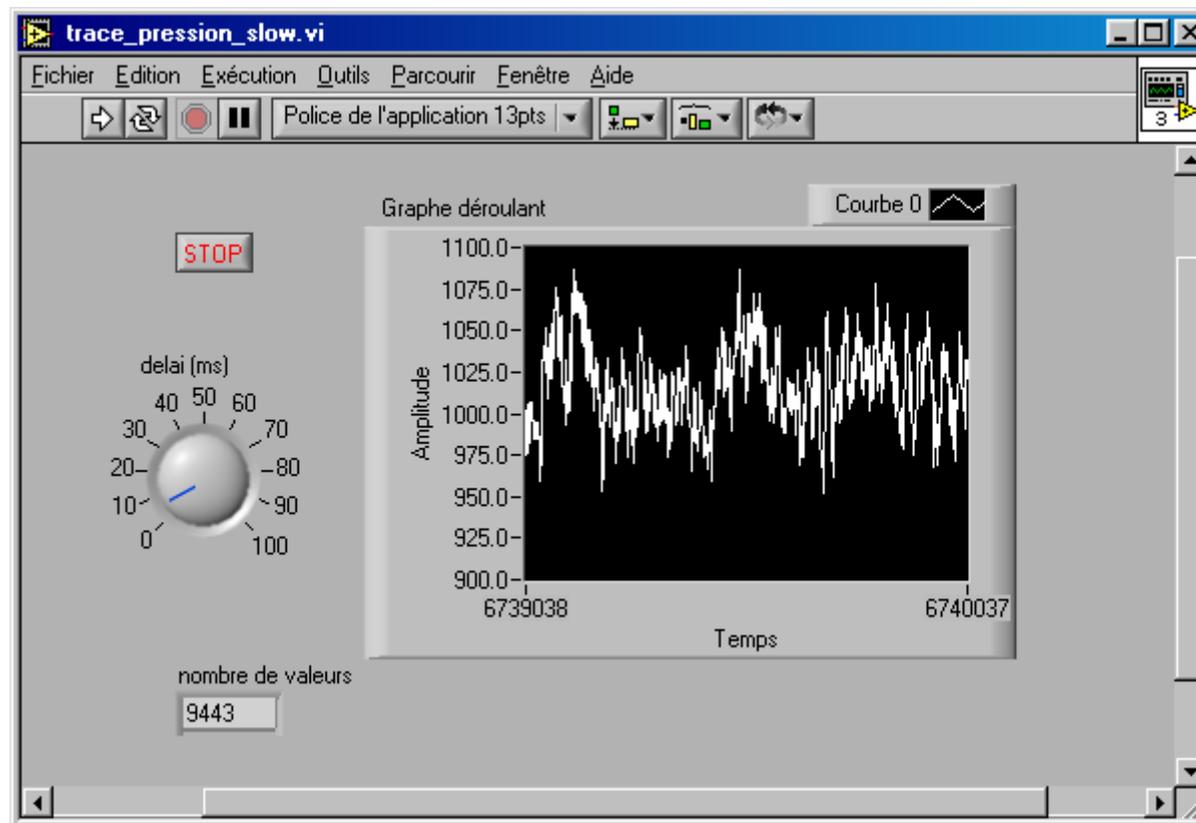




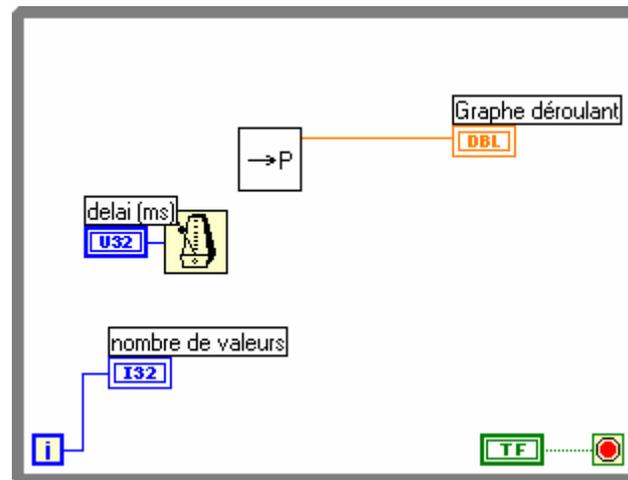
boucle while -temporisation

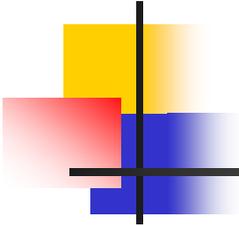
- On peut ralentir l'exécution de façon contrôlée :
- Ajouter dans la boucle ' *attendre un multiple de ms* '
- (métronome dans la palette ' temps et dialogue '
- Créer une molette numérique ' *delai (ms)* ',
la relier à l'entrée du métronome
- exécuter, changer la valeur du délai
(ajuster l'échelle du délai au besoin)
- examiner les différents modes de mise à jour du graphe

Trace pression 2 - face avant



Trace pression 2 - diagramme



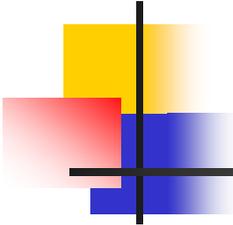


LabVIEW - registres à décalage

Le signal précédent est assez ' bruité '; on pourrait vouloir afficher une moyenne continue des 4 précédentes valeur, par exemple

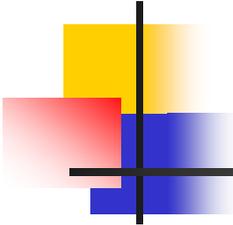
Ceci peut être réalisé en ajoutant un registre à décalage à la boucle





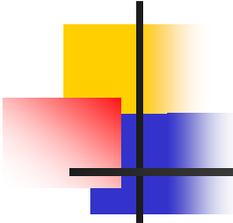
LabVIEW - registres à décalage

- Un registre à décalage prend des données du côté droit et les reporte du côté gauche à l'itération suivante
- Création: <clic droit> sur un des bords de la boucle
→ ajouter un registre à décalage
- <clic droit> sur le registre pour rajouter un élément
- On peut le déplacer en hauteur (les deux côtés restent alignés)



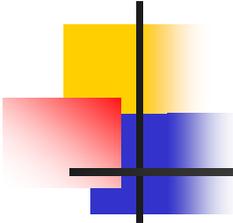
LabVIEW - registres à décalage

- On peut placer plusieurs registres à décalage sur une boucle
- Chaque registre n'accepte qu'une valeur d'entrée (à droite)
- Le type de cette valeur fixe le type du registre (ex: DBL)
ce sera aussi le type des valeurs restituées (à gauche)
- Un registre peut restituer la valeur fournie v_i , mais aussi v_{i-1} , v_{i-2} , ... en rajoutant des éléments



Trace pression - moyennée

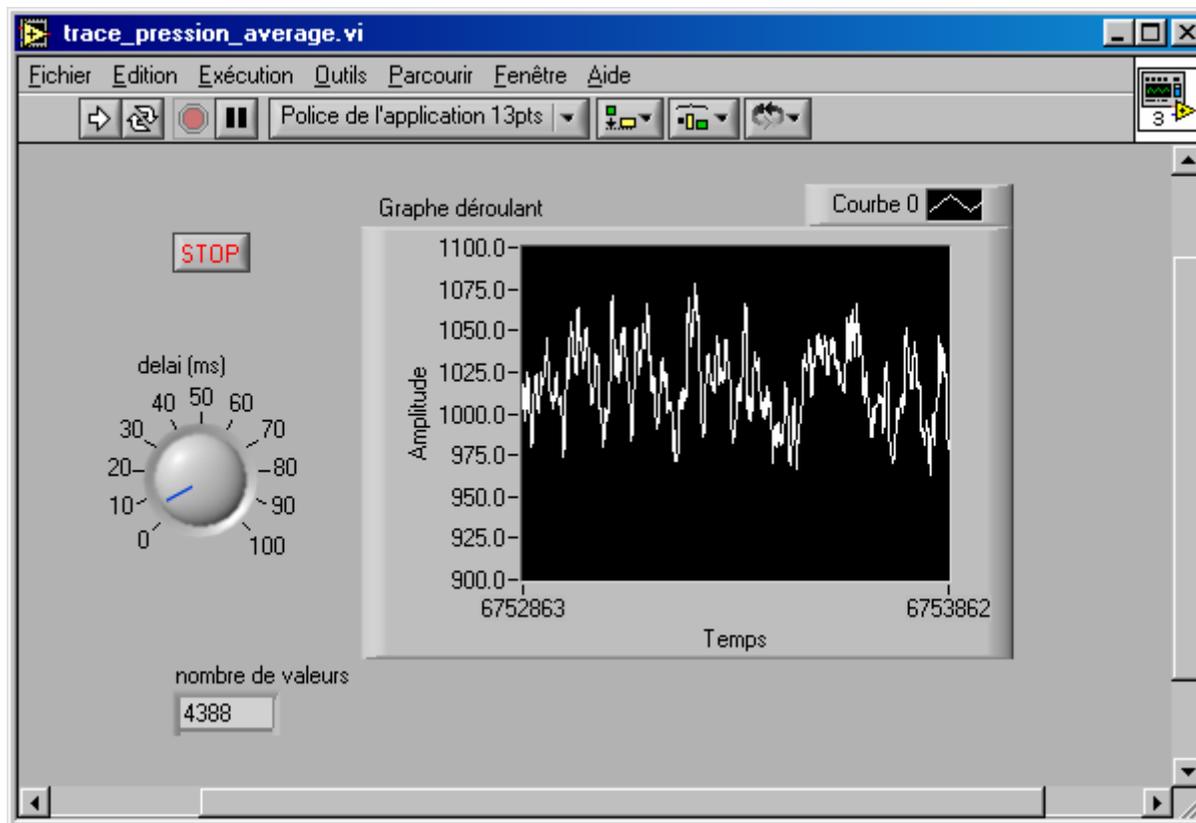
- sauvegarder le VI précédent sous ' **trace_pression_average.vi** '
- Ajouter un registre à décalage
- L'étendre à 4 éléments
- Ajouter une fonction ' *Opérateur arithmétique* ' (palette numérique)
l'étendre à 4 éléments
S'assurer que son mode est bien ' *addition* ' (<clic droit>)
- Relier les 4 sorties de registre (gauche) aux entrées de l'opérateur



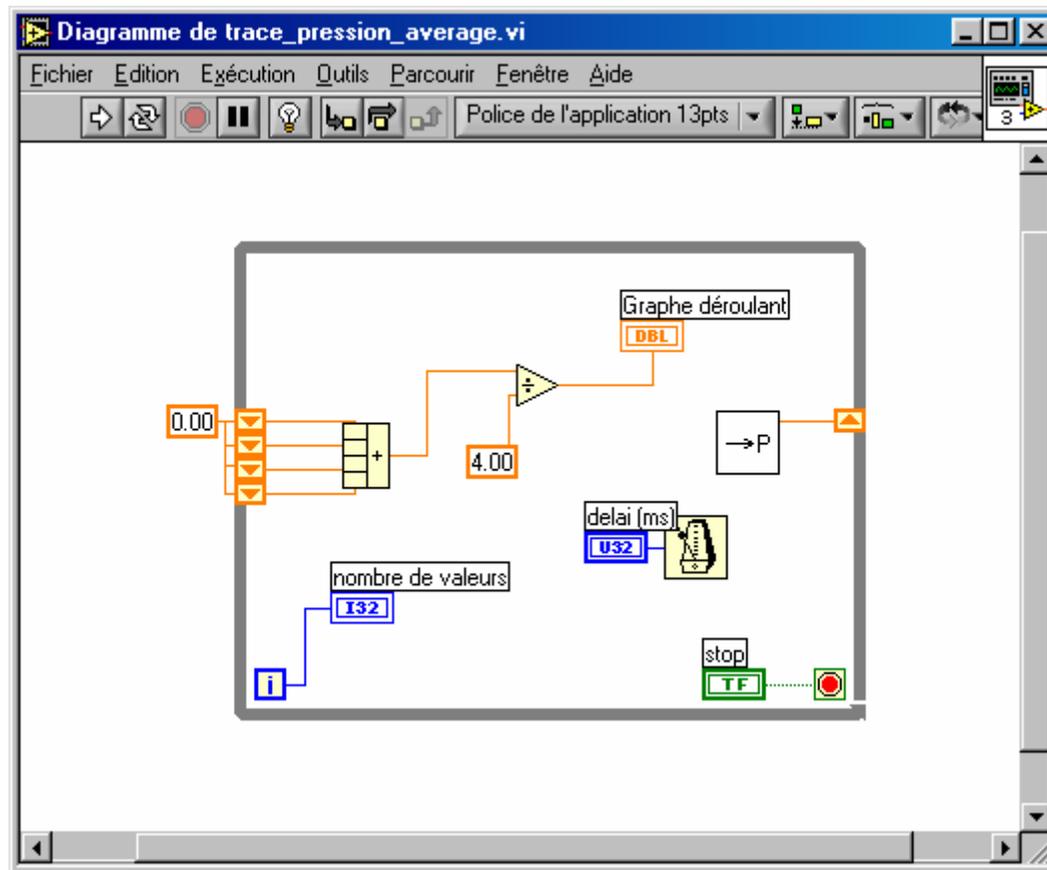
Trace pression - moyennée

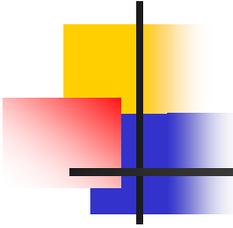
- Diviser la sortie de l'opérateur arithmétique par 4
- Relier la sortie au graphe
- Relier la sortie du capteur de pression à l'entrée du registre (à droite)
- Créer une constante numérique à l'extérieur de la boucle,
La relier aux sorties (gauche) du registre :
Ceci assure l'initialisation des valeurs restituées
par le registre
- Exécuter

Trace pression - moyennée



Trace pression - moyennée

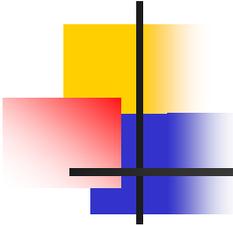




Plan

Nous allons voir maintenant :

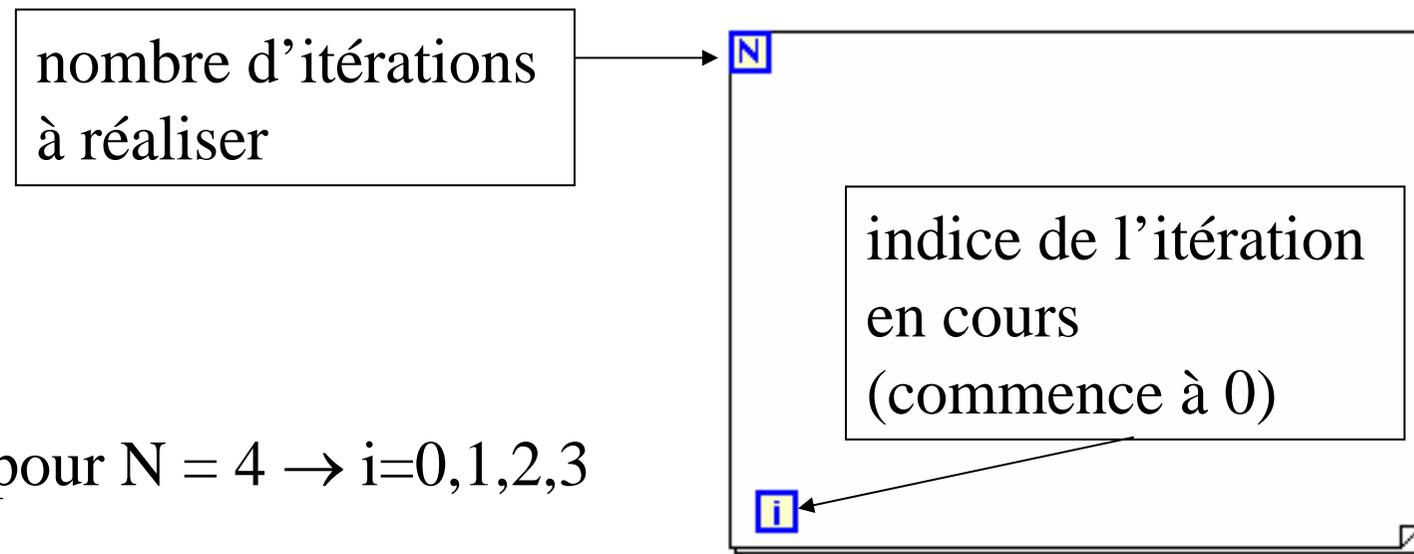
- boucle FOR
- structure condition
- structure séquence



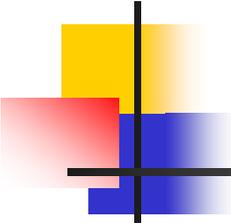
boucles FOR

Nouveau VI

diagramme : ajouter une boucle for (structures → Boucle For)



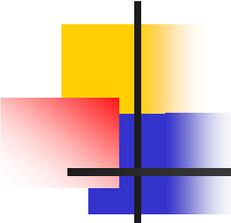
pour $N = 4 \rightarrow i=0,1,2,3$



Boucles FOR - exemple

Illustration : création d'un VI pour déterminer le plus petit de 100 nombres aléatoires

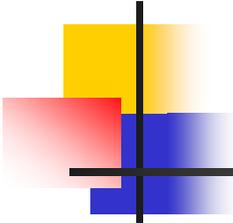
- face avant :
 - 2 indicateurs numériques : ' nombre ' et ' minimum '



Boucles FOR - exemple

Diagramme

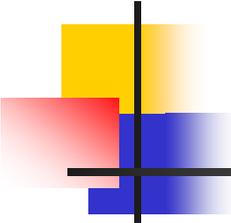
- ajouter une structure boucle FOR
- imposer 100 itérations en connectant une constante numérique au terminal N
remarque: la constante doit être à l'extérieur de la boucle
- placer un générateur de nombre aléatoire dans la boucle (fonction numérique)
- placer une fonction/comparaison/max&min



Boucles FOR - exemple

diagramme :

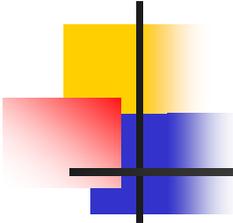
- ajouter un registre à décalage à la boucle FOR
- relier la sortie du générateur aléatoire à l'indicateur ' nombre ' (' nombre ' doit être dans la boucle)
- relier aussi cette sortie à l'entrée ' x ' de la fonction ' max&min '



Boucles FOR - exemple

diagramme :

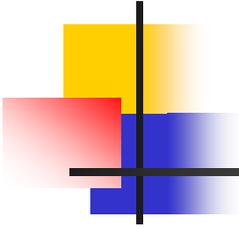
- relier la sortie ‘ $\min(x,y)$ ’ à l’entrée du registre à décalage (à droite)
- connecter une constante numérique de valeur ‘ 1 ’ au registre à décalage (à gauche) pour l’initialisation



Boucles FOR - exemple

diagramme :

- relier la valeur de sortie externe du registre à décalage à l'indicateur ' minimum ' (à droite, à l'extérieur de la boucle)
- relier la sortie interne du registre à décalage à l'entrée ' y ' de la fonction ' max&min ' (à gauche, à l'intérieur de la boucle)
- exécuter (1 fois, pas en continu) :
la valeur minimum parmi 100 nombres aléatoires est affichée



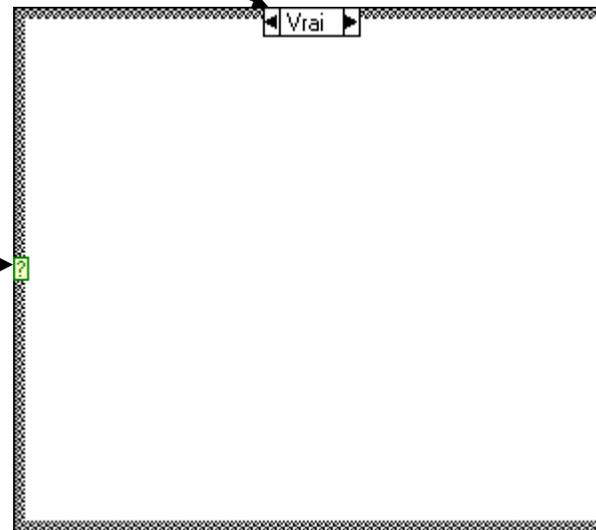
Structure Condition

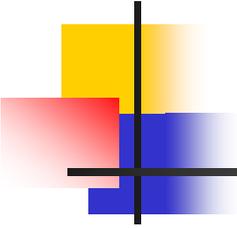
- Permet d'exécuter différentes actions en fonction de la valeur d'une entrée.
(cf. switch/case en C)
- Par défaut, entrée booléenne, mais on peut relier aussi une entrée numérique, chaîne, énumération...
la structure s'adapte au type d'entrée

Structure Condition

pour visualiser les
différentes valeurs possibles,
utiliser les flèches

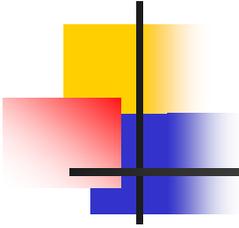
valeur d'entrée →





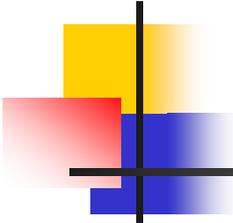
Structure Condition

- On peut ajouter/supprimer des cas possibles (<clic droit>)
- Un cas par défaut *doit* être spécifié pour gérer les valeurs non précisées



Structure Condition - exemple

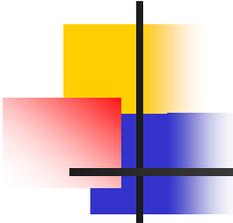
- Créer un nouveau VI - on veut calculer le racine carrée d'un nombre, en gérant les nombres négatifs
- Face avant :
 - Une commande ' nombre '
 - Un indicateur ' racine carree '



Structure Condition - exemple

diagramme :

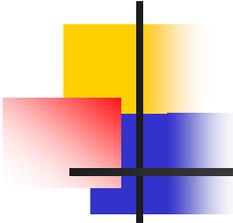
- Ajouter une structure condition
- ajouter une fonction de comparaison ' ≥ 0 '
- ' nombre ' et ' ≥ 0 ' doivent être à l'extérieur, à gauche de la structure condition
- relier nombre à l'entrée de ' ≥ 0 ' ;
la sortie est un booléen, à relier à l'entrée de la structure condition [?] pour la contrôler.



Structure Condition - exemple

diagramme :

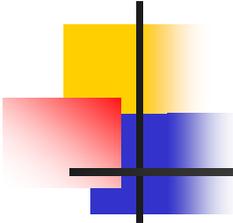
- sélectionner l'option ' Vrai ' dans la structure condition
- Ajouter une fonction numérique ' racine carrée ' dans la structure condition
- Placer l'indicateur ' racine carree ' à l'extérieur, à droite de la structure condition



Structure Condition - exemple

diagramme :

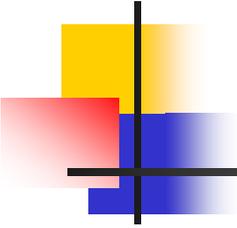
- Relier ‘ nombre ’ à l’entrée de la fonction ‘ racine carrée ’
 - remarquer le tunnel (carré orange) dans la paroi gauche de la structure
- Relier la sortie de la fonction ‘ racine carrée ’ à l’indicateur ‘ racine carree ’
 - remarquer le tunnel dont l’intérieur est blanc car aucune valeur n’est encore fournie dans le cas Faux d’ailleurs le VI n’est pas encore exécutable (flèche brisée)



Structure Condition - exemple

diagramme :

- Sélectionner le cas ' Faux ' dans la structure condition
- Placer dedans une constante numérique ' -999 '
à relier au tunnel de sortie
 - Remarquer qu'il se remplit et que le VI est exécutable
- Placer une fonction
' Temps et dialogue/Boîte de dialogue 1 bouton '
- Relier une chaîne ' Erreur, nombre négatif ' à la 1ère entrée
- Relier une chaîne ' Ok, mea culpa ' à la 2e entrée



Structure Condition - exemple

Sauver

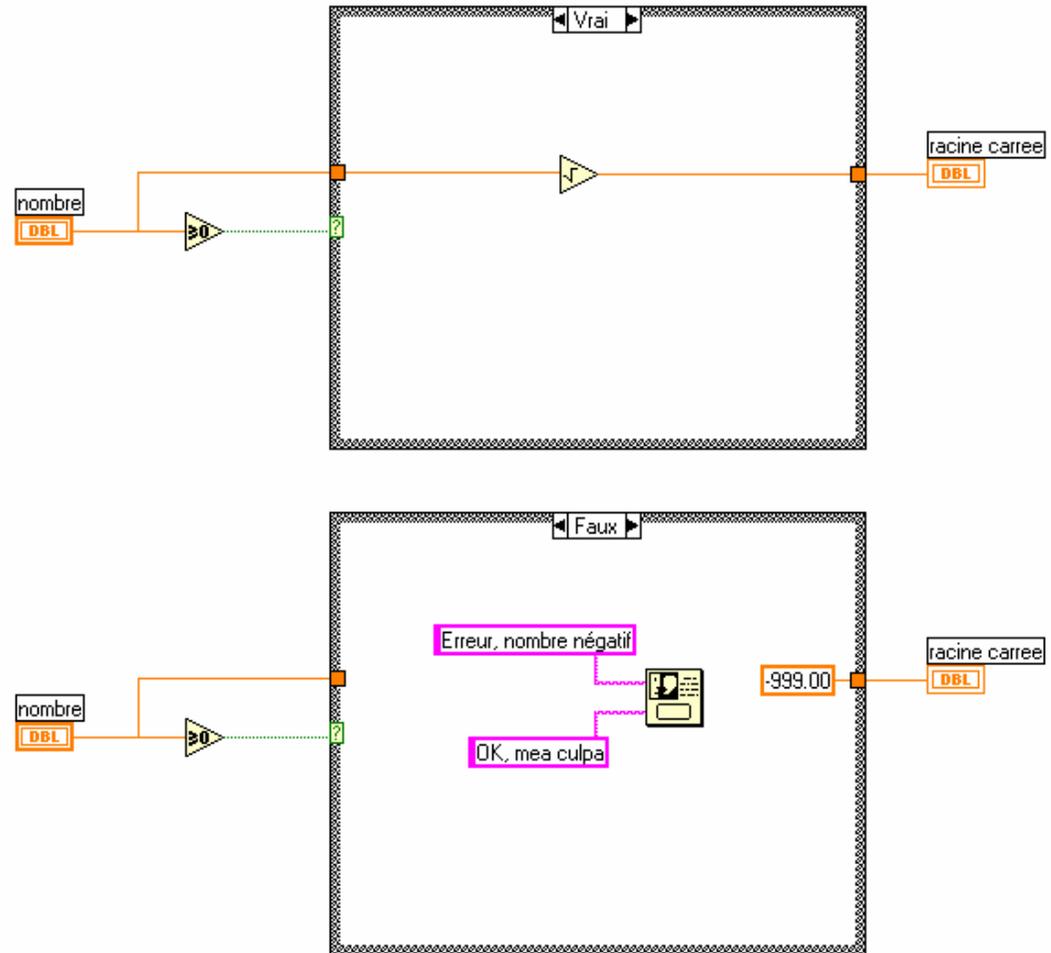
Changer ' nombre ' (positif puis négatif)

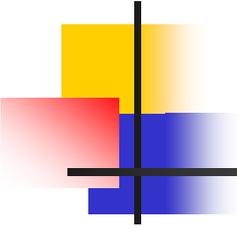
et exécuter (1 fois, **pas en continu** à cause de la boîte de dialogue)

vérifier que tout va bien.

Structure Condition - exemple

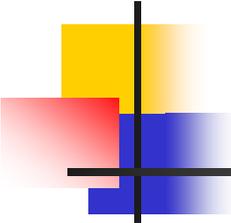
nombre 3.00
racine carree 1.73





Structure Séquence

- Permet de programmer différents sous-diagrammes à exécuter séquentiellement
- A utiliser seulement à bon escient



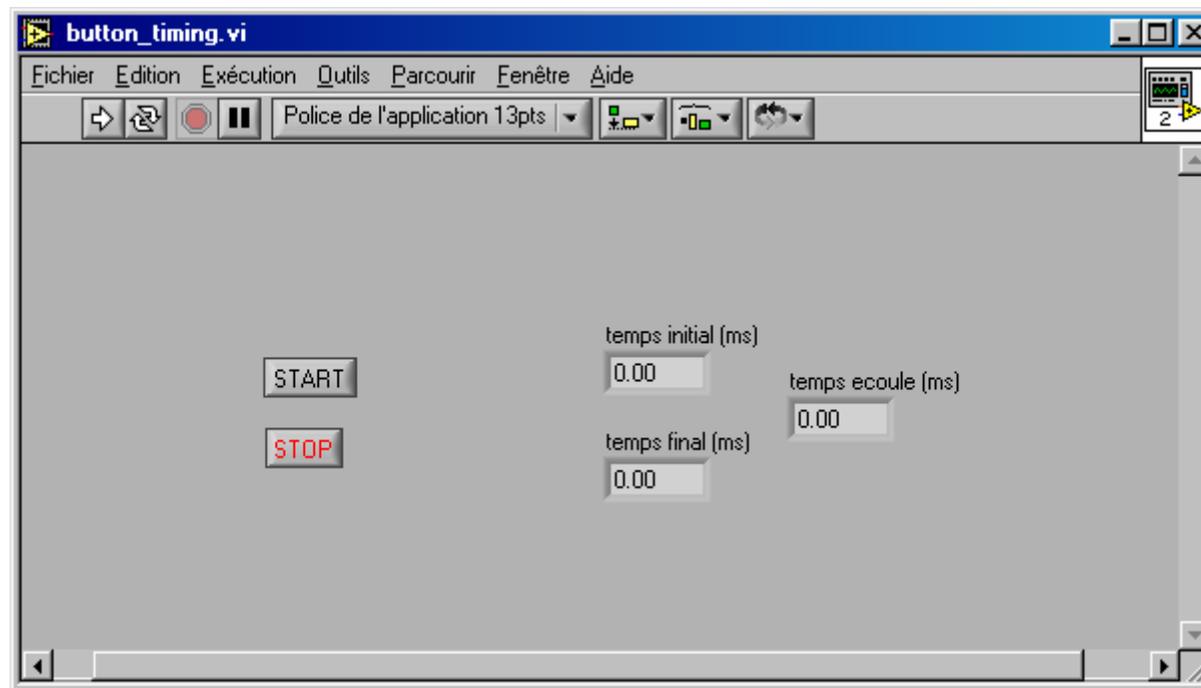
Séquence - exemple

A titre d 'exemple, mesurer le temps entre deux pressions de boutons

Face avant :

- Deux boutons ' start ' et ' stop '
 - Action mécanique : ' commutation jusqu'au relâchement '
- Trois indicateurs numériques :
' temps initial (ms)', ' temps final (ms)', ' duree ecoulee (ms) '
représentations : U32

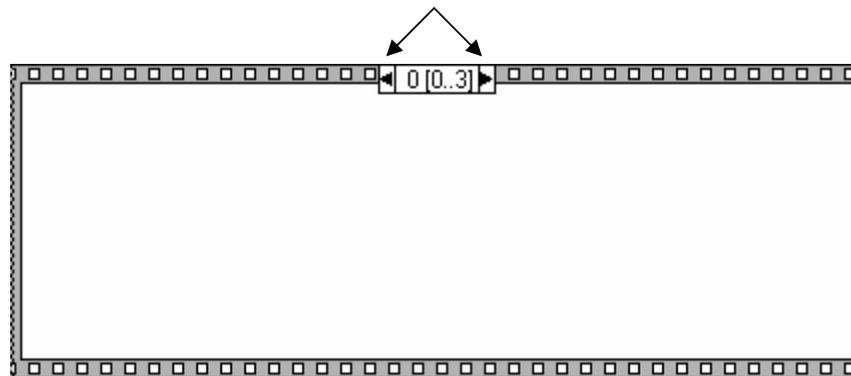
Séquence - exemple



Séquence - exemple

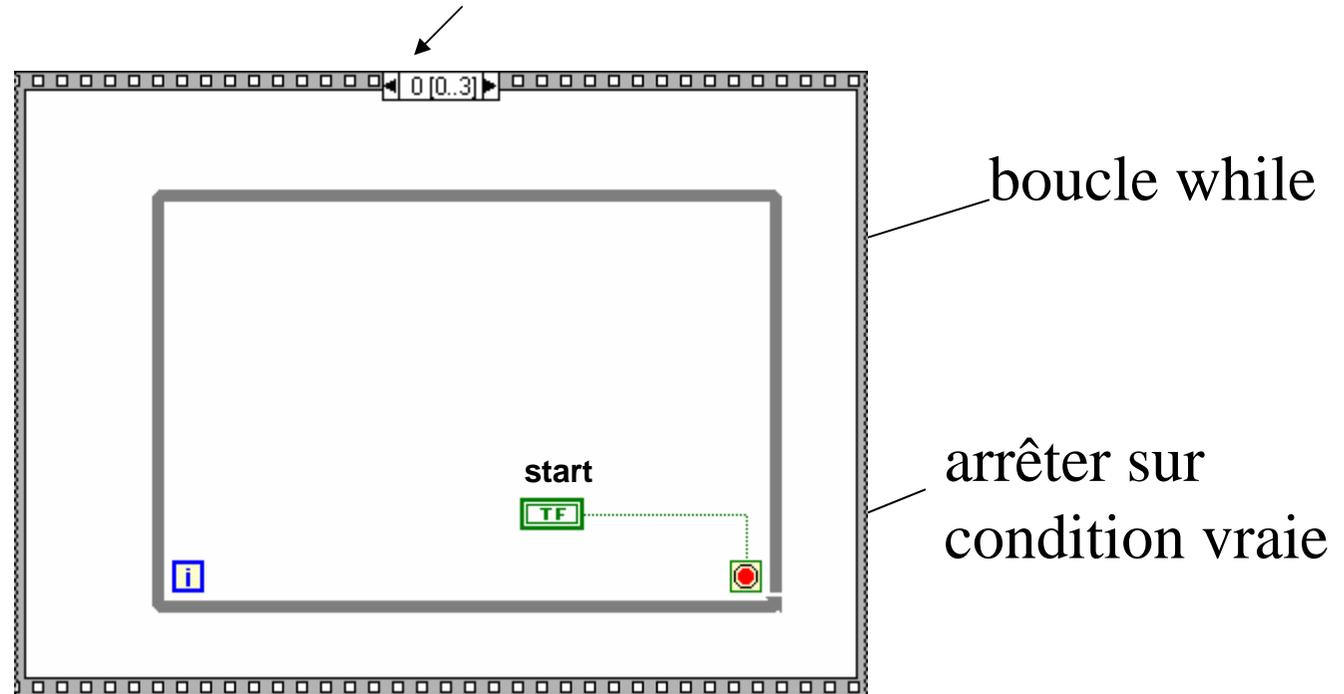
Diagramme :

- ajouter une structure ‘ Séquence ’
- <clic droit> sur la structure, ‘ ajouter une étape après ’
- la séquence doit alors comprendre deux étapes 0..1
- Recommencer jusqu’à ce qu’il y ait 4 étapes (0..3)
- Sélectionner la première étape (0) en utilisant les flèches



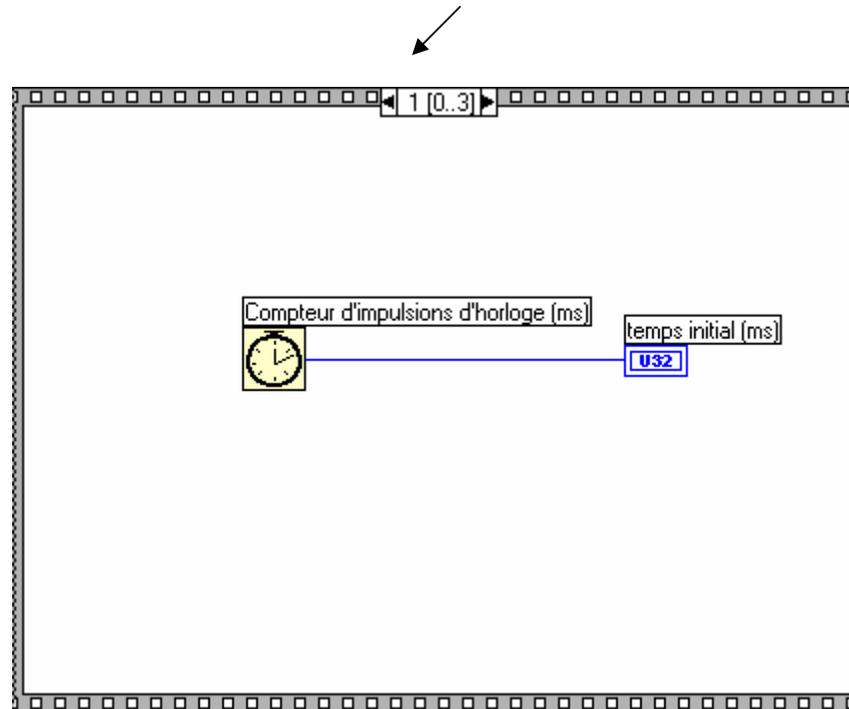
Séquence - étape 0

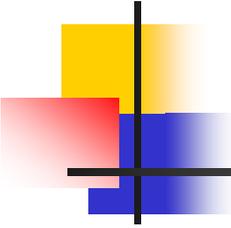
Créer les diagrammes des différentes étapes, comme indiqué :



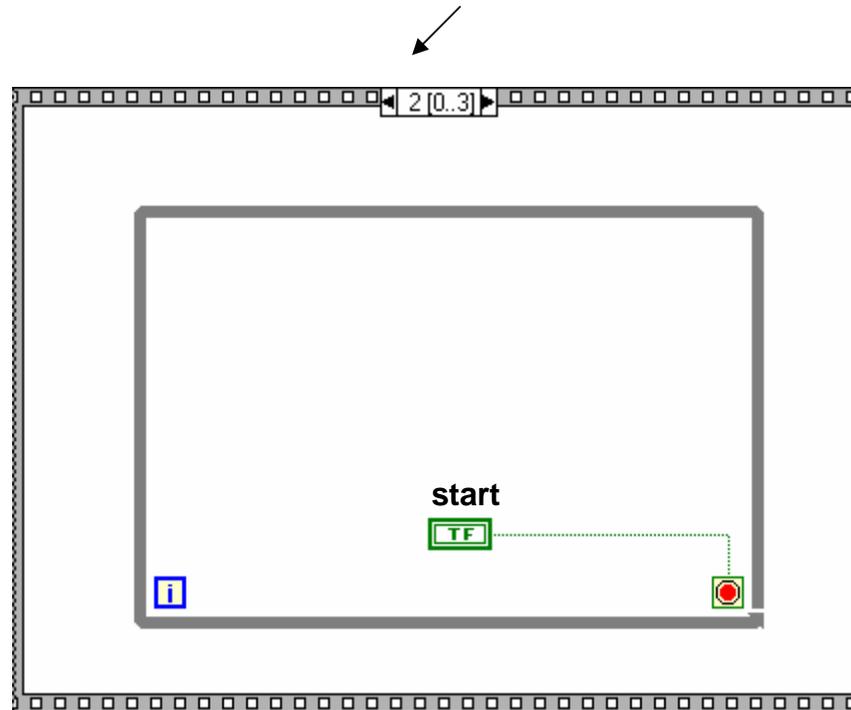
Séquence - étape 1

Créer les diagrammes des différentes étapes, comme indiqué :

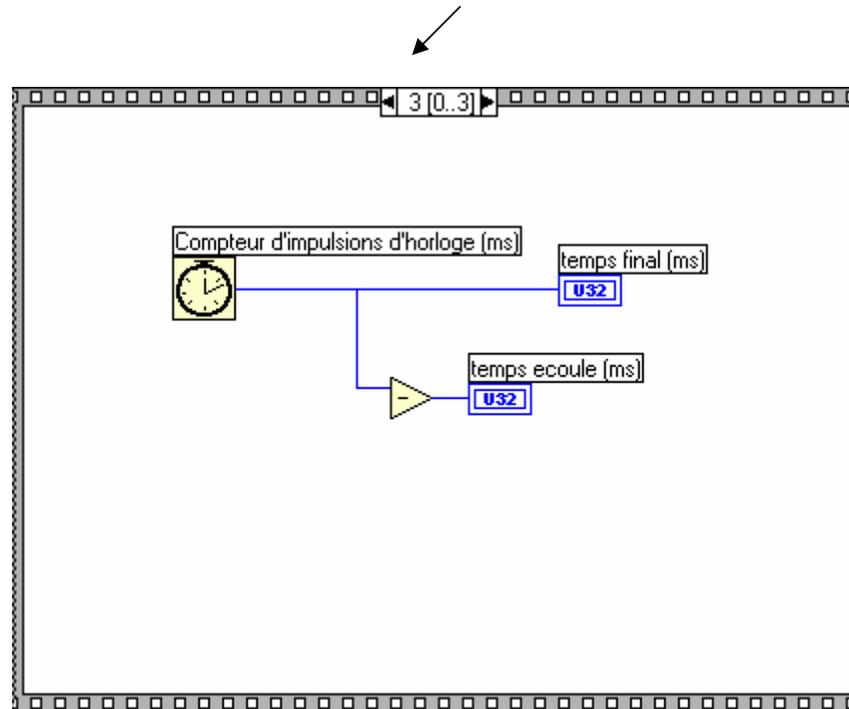




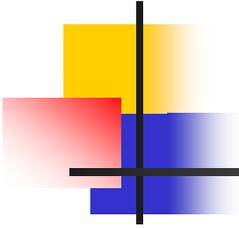
Séquence - étape 2



Séquence - étape 3



Incomplet pour
l'instant
Il faut le temps
initial...

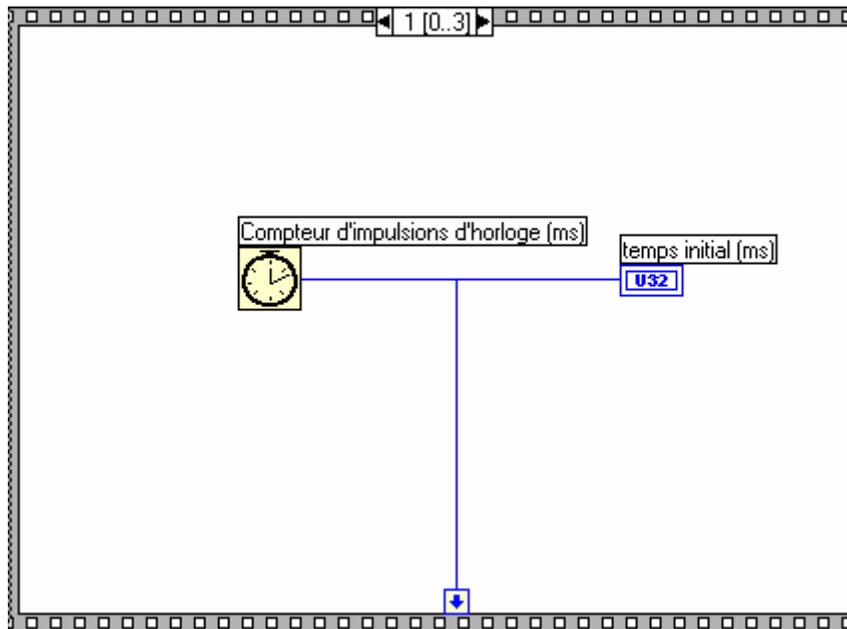


Variable locale de Séquence

Pour disposer de la valeur du temps initial, on va utiliser une ‘ variable locale de séquence ’

- sélectionner la 2e étape (1)
- <clic droit> sur le bord inférieur de la structure, ‘ ajouter une variable locale de séquence ’
- un carré jaune apparaît.
- relier ce carré au fil existant (voir transparent suivant)

Variable locale de Séquence



Une variable locale de séquence permet de transmettre une valeur aux étapes suivantes (pas aux précédentes)

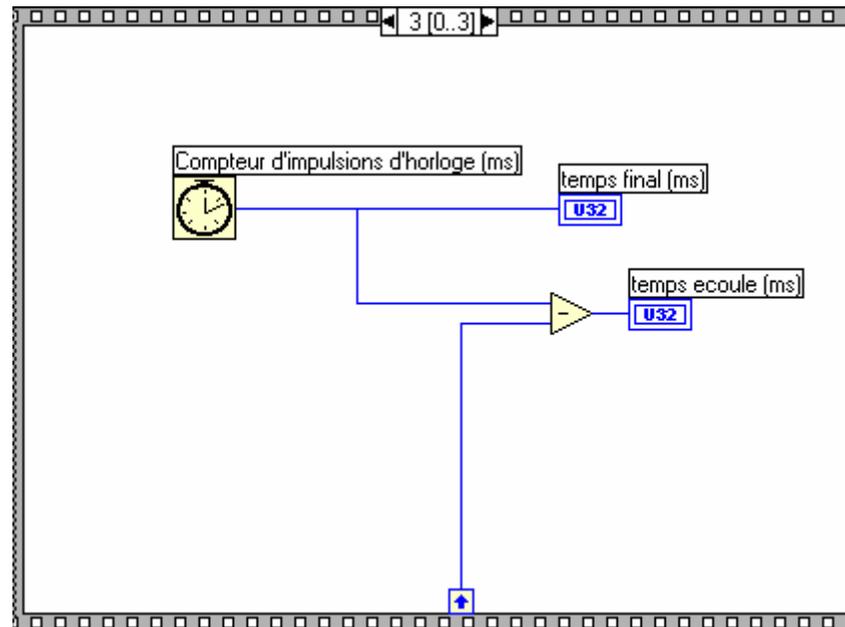
Rq. la flèche pointe vers l'extérieur

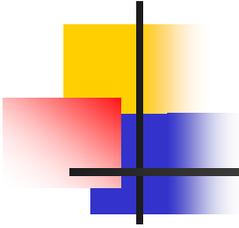
Variable locale de Séquence

Sélectionner la dernière étape

relier la variable à la fonction ‘ - ’

Rq: la flèche pointe vers l’intérieur, indiquant que la valeur est disponible





Variable locale de Séquence

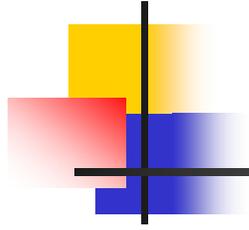
Exécuter (1 fois)

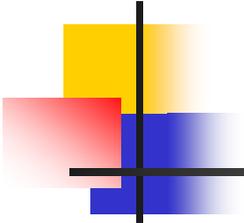
Cliquer sur le bouton ‘ start ’, puis ‘ stop ’

La sortie indique le temps (en ms) entre les deux clics

Voir si la valeur est disponible aussi aux étapes 0 et 2

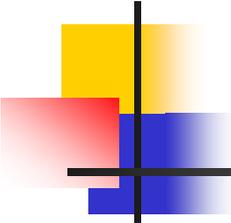
Il est possible d’avoir plusieurs variables locales de sequences





La boîte de calcul

- ✿ La fonction ' boîte de calcul ' est disponible dans le sous-menu ' Structures '
- ✿ Permet d'intégrer des formules dans un diagramme sans utiliser un câblage compliqué
- ✿ Un nombre limité de fonctionnalités est disponible (voir l'aide Labview pour plus de détails)

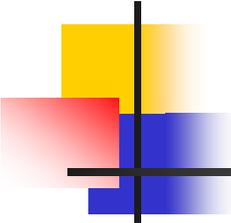


Boîte de calcul - exemple

☀ Démarrer un nouveau VI

☀ Créer sur la face avant:

- Une commande numérique – ' X ' (gamme 1-100 , Incrément 1 , entier)
- Un indicateur numérique



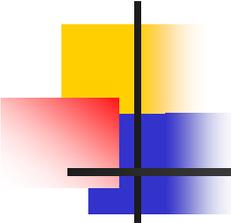
Boîte de calcul - exemple

✿ Créer sur le diagramme:

- Une boîte de calcul
- Ecrire dans la boîte la formule mathématique :
 $Y=X^{*}2 + 2*X +7;$

Remarques:

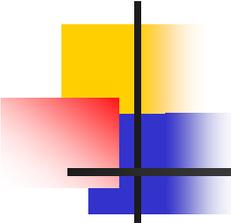
- *Noter que ' ** ' est la fonction ' puissance '*
(ne pas confondre avec ' ^ ' qui est la fonction ' OU exclusif ')
- *Ne pas oublier le point-virgule à la fin de chaque ligne de commande!*



Boîte de calcul - exemple

☀ Sur le diagramme:

- Faire un ' clic droit ' sur le côté gauche de la boîte de calcul et sélectionner ' ajouter une entrée ' que l'on renommera ' X '
- Faire un ' clic droit ' sur le côté droit de la boîte de calcul et sélectionner ' ajouter une sortie ' que l'on renommera ' Y '



Boîte de calcul - exemple

✿ Sur le diagramme:

- Connecter la commande numérique ' X ' à l'entrée ' X ' de la boîte de calcul
- Connecter l'indicateur numérique ' Y ' à la sortie ' Y ' de la boîte de calcul

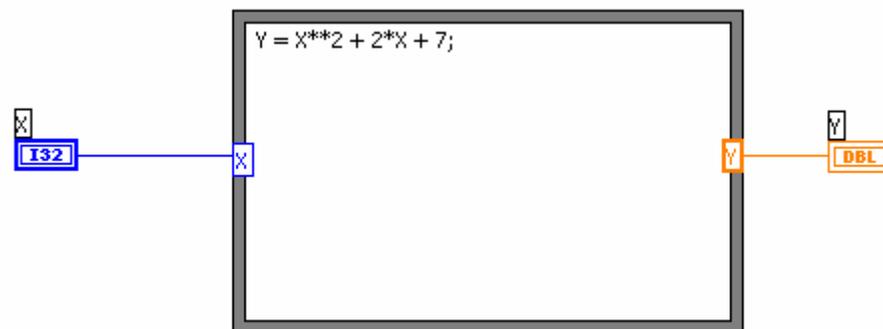
Boîte de calcul - exemple

On obtient alors:

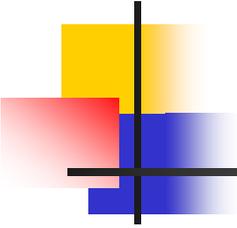


Face-avant

Diagramme



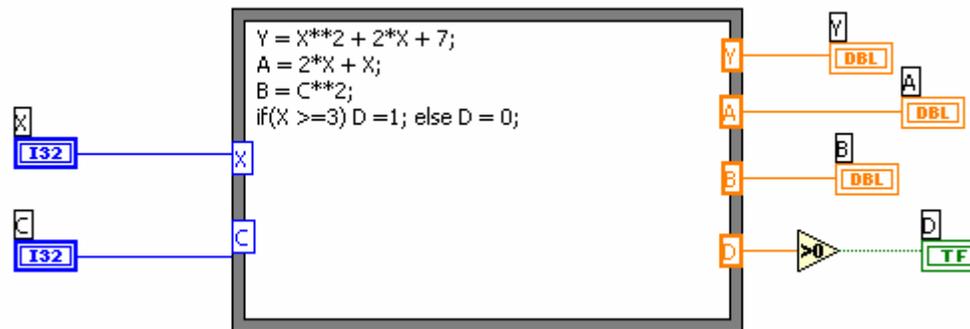
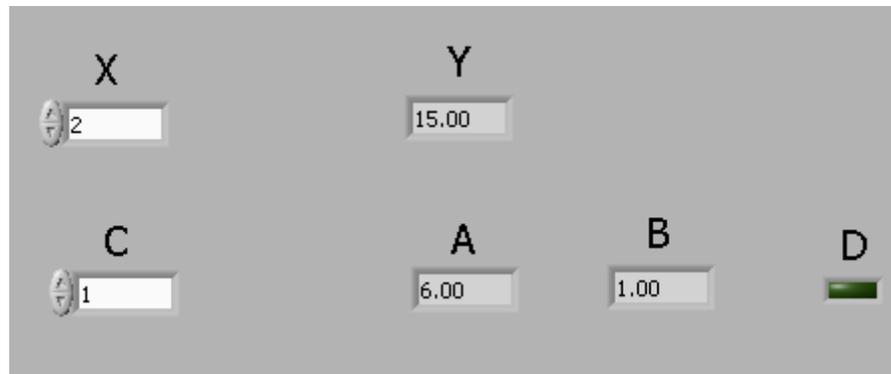
Faire tourner plusieurs fois la programme avec plusieurs valeurs de ' X ' et vérifier le bon fonctionnement



Boîte de calcul - exemple

- ✿ Une boîte de calcul peut contenir plusieurs lignes de commande (chaque ligne étant terminée par ' ; ')
- ✿ Des choix peuvent être inclus (par exemple une fonction *if ... else ...*)
- ✿ Même s'il est possible de placer des entrées et des sorties à n'importe quelle position sur la boîte de calcul, il est conseillé d'appliquer la convention suivante:
entrées sur le côté gauche et sorties sur le côté droit

Boîte de calcul - exemple

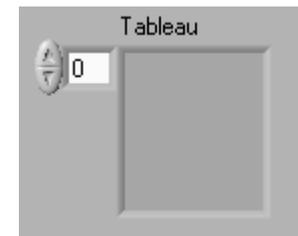


Les tableaux sous Labview

☀ Démarrer un nouveau VI

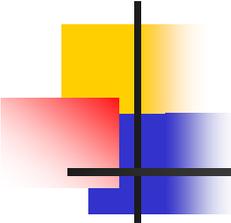
☀ Sur la face avant:

- Créer un tableau, fonction dans le sous-menu 'Tableau et Cluster'
- Noter l'apparence du tableau sur la face-avant: il apparaît vide lorsque aucun type de donnée lui a été assigné:



- Faire glisser un indicateur numérique dans le tableau. Noter alors la modification d'aspect du tableau:





Les tableaux sous Labview

☀ Sur le diagramme:

- Noter comment le symbole tableau a été modifié en orange ce qui indique qu'on lui a assigné un type de donnée (ici double précision)



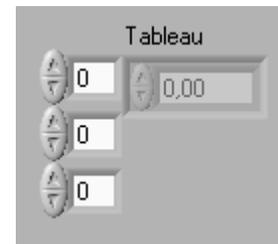
- Il est également possible de modifier le type et la représentation par un 'clic-droit' (cf. indicateurs numériques)

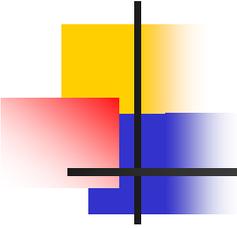
Les tableaux sous Labview

☀ Sur la face-avant:

- Le tableau par défaut a une seule dimension. Il est possible d'ajouter des dimensions supplémentaires par les deux méthodes suivantes:
 - 'clic-droit' et sélectionner 'ajouter une dimension'
 - étendre à la souris la boîte d'indice
- Des dimensions peuvent être également supprimées
- Remarquer que les valeurs dans le tableau sont grisées indiquant qu'elles n'ont pas été initialisées

Exemple : un tableau à 3 dimensions

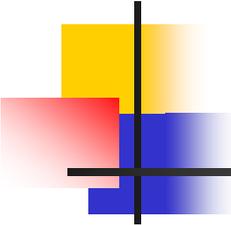




Les tableaux sous Labview

☀ Sur la face-avant:

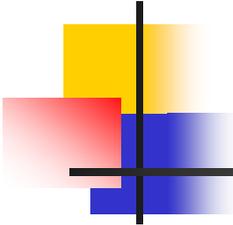
- Dans un tableau à une dimension, sélectionner l'indice '10' et initialiser le à la valeur 5 (noter le changement d'affichage)
- Sélectionner un indice en dessous de 10 : la valeur est toujours 0.00 mais son apparence n'est plus grisée indiquant que les valeurs des cellules ont été initialisées
- Observer les cellules d'indice supérieur à 10



Les tableaux sous Labview

☀ Sur la face-avant:

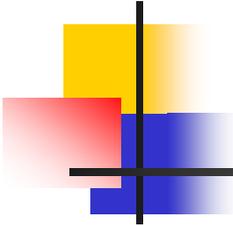
- Retourner à l'indice '10', positionner le curseur au dessus de la valeur de la cellule, puis faire un 'clic-droit' et sélectionner la fonction 'opérations sur les données/réinitialiser à la valeur par défaut'
- Toutes les cellules indexées de 0 à 10 sont maintenant initialisées à 0.00 (les cellules d'indices supérieurs à 10 apparaissent encore grisées, donc non initialisées)



Les tableaux sous Labview

☀ Sur la face-avant:

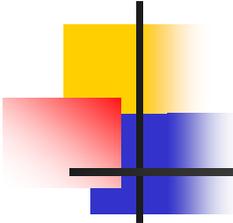
- Faire un 'clic-droit' sur la boîte d'indice et sélectionner à nouveau la fonction 'opérations sur les données/réinitialiser à la valeur par défaut'
- Cette fois-ci, toutes les cellules du tableau sont grisées signifiant que le tableau est vide
- Les tableaux peuvent avoir une ou plusieurs dimensions, chaque dimension pouvant contenir jusqu'à 2147483647 éléments ($2^{31} - 1$)



Tableaux – auto-indexation

- ✿ Si un tableau est connecté à une boucle FOR ou WHILE, on peut initialiser les éléments du tableau, l'indice du tableau étant lié au compteur de la boucle
- ✿ Cette facilité est activée par défaut dans une boucle FOR
- ✿ Elle est désactivée par défaut dans une boucle WHILE

Le prochain exemple illustre cette méthode



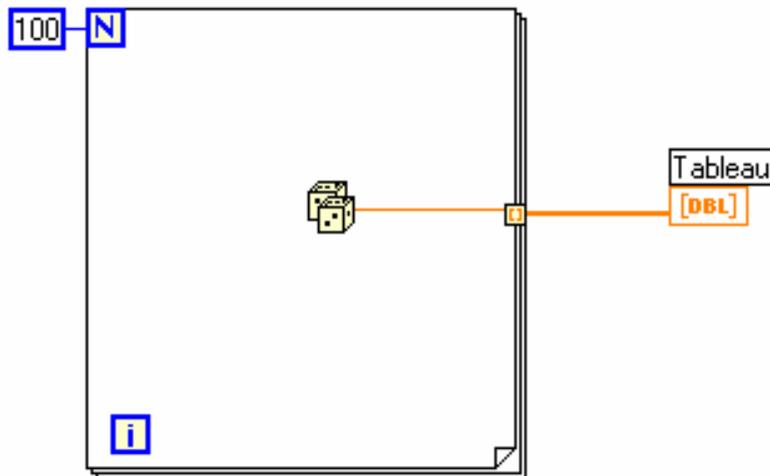
Auto-indexation exemple

- ✿ Créer un nouveau VI
- ✿ Sur la face avant, créer un tableau et ajouter dans ce tableau un indicateur numérique
- ✿ Sur le diagramme:
 - ajouter une boucle FOR et définir la valeur d'arrêt 'N' à 100 (**le tableau devant se trouver à l'extérieur de la boucle**)
 - à l'intérieur de la boucle, ajouter un générateur de nombre aléatoire (0-1)
 - Connecter le générateur de nombre aléatoire au tableau

Auto-indexation exemple

On obtient alors:

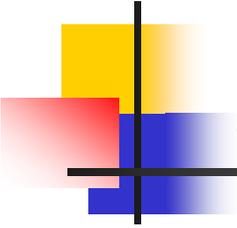
Diagramme:



Face-avant:

Tableau	
96	0,74
	0,07
	0,91
	0,69
	0,72
	0,00
	0,00
	0,00
	0,00

Après exécution, vérifier que le tableau soit rempli jusqu'à l'indice 100



Auto-indexation exemple

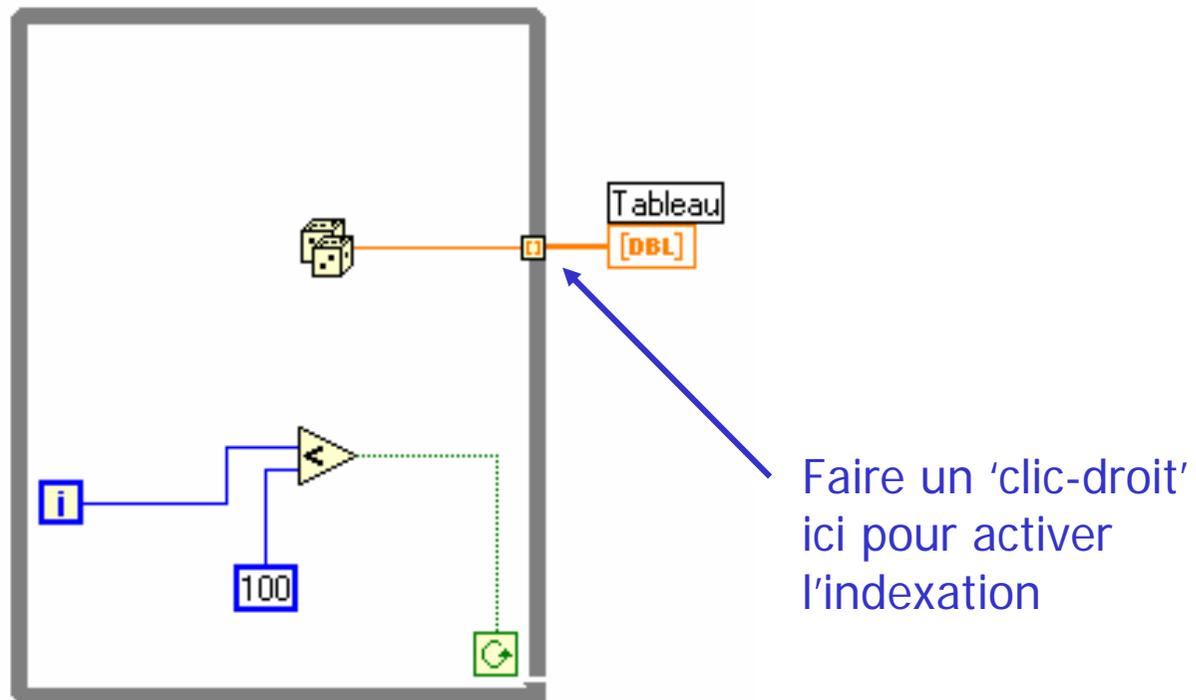
On peut également remplir un tableau à l'aide d'une boucle while.

Faire un nouveau VI permettant cette initialisation jusqu'à l'indice 100.

A VOUS DE JOUER!!!!

Auto-indexation exemple

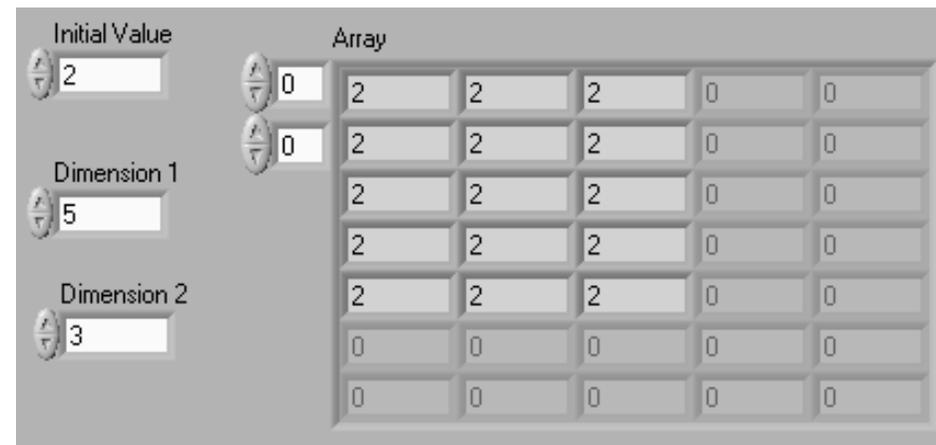
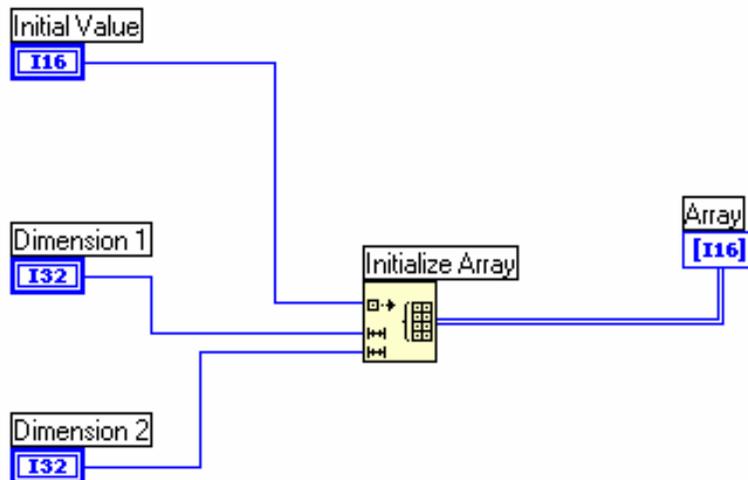
L'initialisation d'un tableau à l'aide d'une boucle WHILE:



Se souvenir que l'indexation n'est pas possible par défaut dans une boucle WHILE!!!

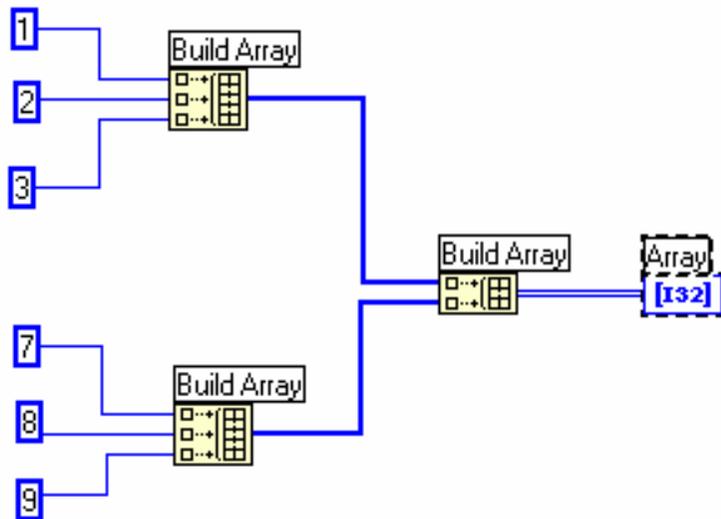
Initialisation d'un tableau

- ☀ Si on souhaite initialiser tous les éléments d'un tableau à une valeur identique, il existe une fonction dans Labview qui permet de le réaliser très simplement.
- ☀ Sur le diagramme, sélectionner la fonction '*initialiser un tableau*' dans le sous-menu '*tableau*'



Initialisation d'un tableau

- ☀ On peut également construire un tableau, éléments après éléments, dimension après dimension.
- ☀ Sur le diagramme, sélectionner la fonction '*construire un tableau*' dans le sous-menu '*tableau*'

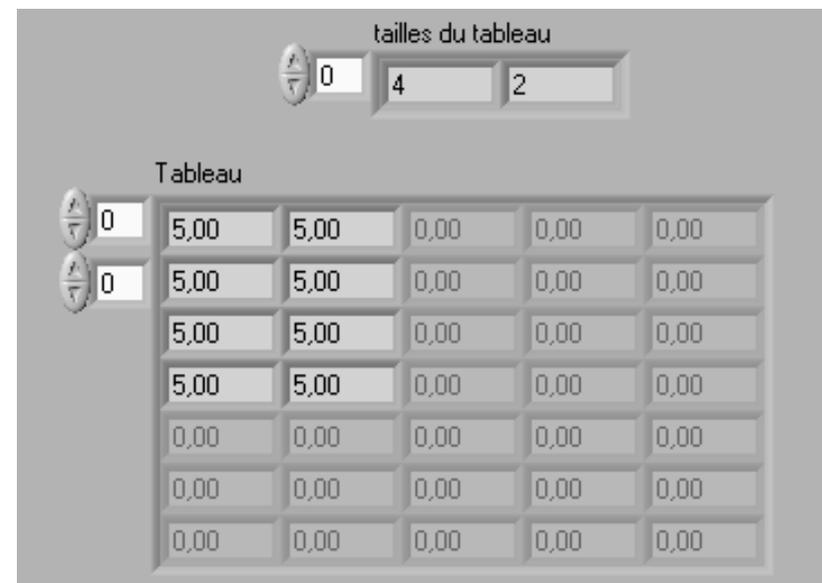
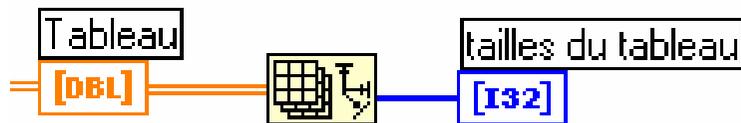


The screenshot shows the 'Array' control in LabVIEW. It consists of a grid of 4 rows and 4 columns of numeric indicators. The first two rows have their first three columns populated with the values 1, 2, 3 and 7, 8, 9 respectively. The remaining two rows are empty. To the left of the grid, there are two 'Array Size' indicators, both showing the value 0.

Array				
0	1	2	3	0
0	7	8	9	0
	0	0	0	0
	0	0	0	0

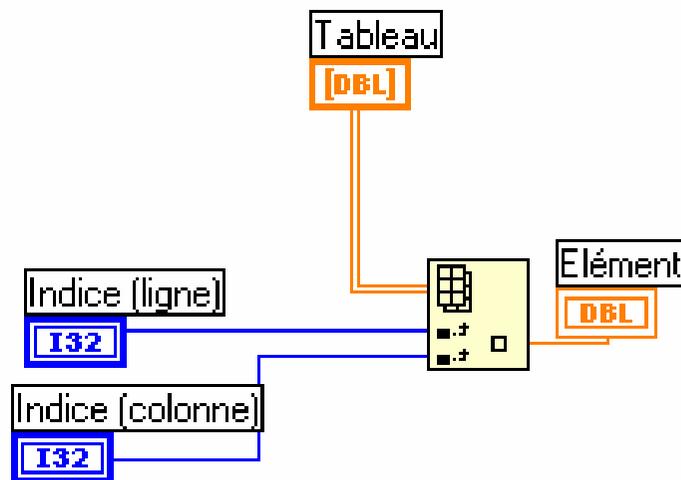
Tableaux - fonctions

- ☀ Il existe également une fonction permettant de retourner la taille d'un tableau
- ☀ Sur le diagramme, sélectionner la fonction '*taille d'un tableau*' dans le sous-menu '*tableau*'



Tableaux - fonctions

- ☀ Il est possible d'extraire la valeur de n'importe quelle cellule d'un tableau
- ☀ Sur le diagramme, sélectionner la fonction '*indexer un tableau*' dans le sous-menu '*tableau*'



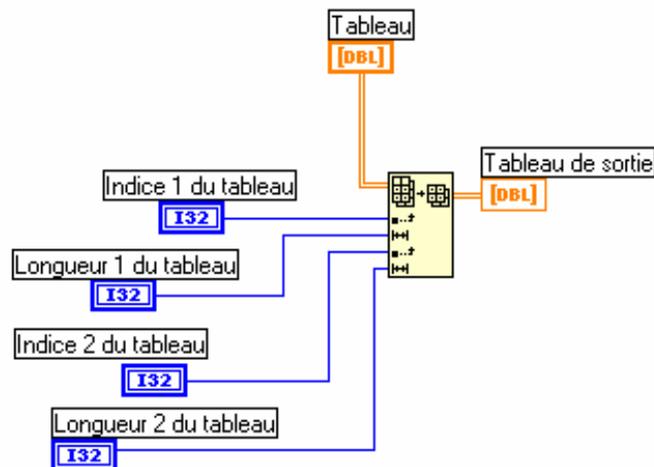
0	5,00	8,00	0,00	0,00	0,00
0	3,00	6,00	0,00	0,00	0,00
	2,00	1,00	0,00	0,00	0,00
	4,00	10,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00

Indice (ligne) 1 Indice (colonne) 1 Elément 6,00

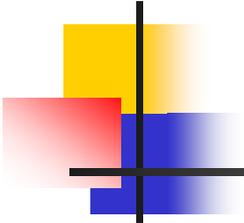
On peut également extraire une ligne avec cette fonction

Tableaux - fonctions

- ☀ Il est possible d'extraire une partie d'un tableau
- ☀ Sur le diagramme, sélectionner la fonction '*sous-ensemble d'un tableau*' dans le sous-menu '*tableau*'



The screenshot displays two tables. The 'Tableau' (input) is a 5x4 grid with values: (5.00, 8.00, 9.00, 0.00), (3.00, 6.00, 5.00, 0.00), (2.00, 1.00, 8.00, 0.00), (4.00, 10.00, 2.00, 0.00), and (0.00, 0.00, 0.00, 0.00). A red circle highlights the second row and third column. A red arrow points from this circle to the 'Tableau de sortie' (output), which is a 2x4 grid with values: (6.00, 5.00, 0.00, 0.00) and (1.00, 8.00, 0.00, 0.00). Below the tables are four input fields: 'Indice 1 du tableau' (value 1), 'Indice 2 du tableau' (value 1), 'Longueur 1 du tableau' (value 2), and 'Longueur 2 du tableau' (value 2).



Chaînes de caractères

Dans cette partie, nous présenterons la manipulation de chaînes de caractères sous Labview.

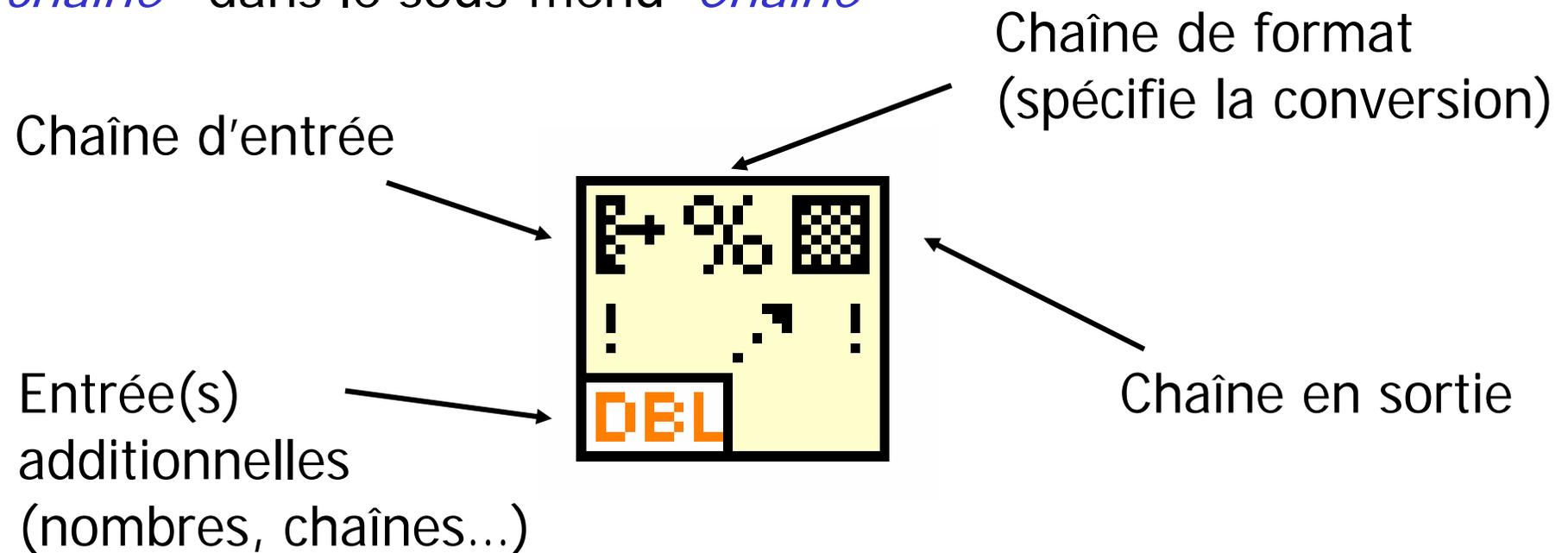
✿ Créer un nouveau VI

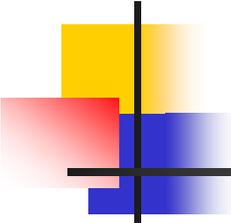
✿ Sur la face d'entrée:

- créer deux commandes chaîne que l'on nommera respectivement '**début**' et '**fin**' (menu '*chaîne&chemin*')
- Ajouter un indicateur chaîne '**chaîne sortie**'
- Ajouter une commande numérique '**Nombre**'

Formater en chaînes

- ☀ Sur le diagramme, ajouter la fonction '*formater en chaîne*' dans le sous-menu '*Chaîne*'





Formater en chaînes

☀ Sur le diagramme:

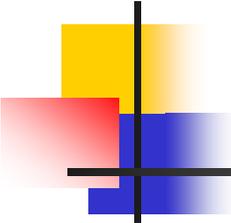
- Connecter 'Début' à l'entrée de chaîne de la fonction '*formater en chaîne*'
- Ajouter une deuxième entrée supplémentaire
- Connecter 'Nombre' à la première entrée supplémentaire
- Connecter 'Fin' à la deuxième entrée supplémentaire

Contrôles de chaînes

☀ Sur le diagramme, faire un 'clic-droit' sur la commande de chaîne

☀ Sélectionner '*mettre à jour la valeur pendant la saisie*', option permettant une mise à jour du contenu de la chaîne sans avoir à presser sur entrée ou sur le bouton 

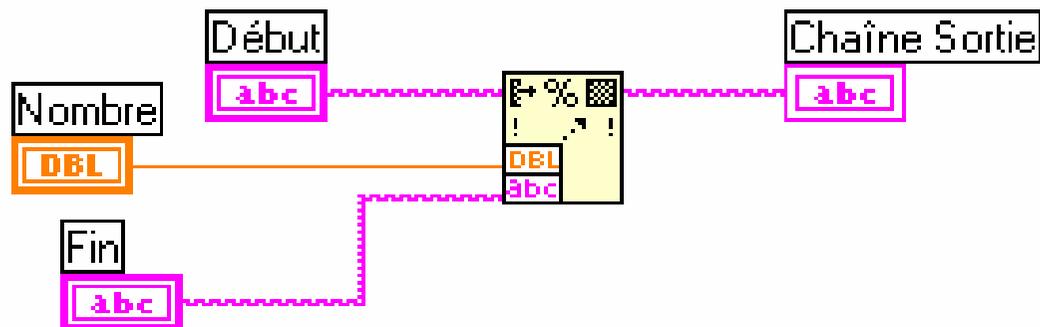
Remarque: On peut également sélectionner l'option '*limiter à une seule ligne*'



Formater en chaînes

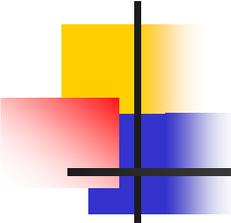
- ☀ Sur le diagramme, connecter la sortie de la fonction à l'indicateur '**chaîne sortie**'
- ☀ Lancer l'exécution en continu
- ☀ Ajouter du texte dans les commandes '**début**' et '**fin**' et un nombre dans la commande numérique '**Nombre**'
- ☀ L'indicateur de chaîne '**chaîne sortie**' doit montrer la concaténation des trois entrées...

Formater en chaînes



The screenshot shows a user interface with the following elements:

- Début**: A text input field containing the value **abc**.
- Nombre**: A numeric input field containing the value **5,00**.
- Fin**: A text input field containing the value **def**.
- Chaîne Sortie**: A large text output field containing the result **abc5,000000 def**.

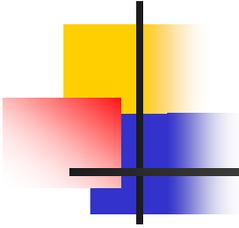


Formater en chaînes

Maintenant, il serait intéressant de mieux contrôler l'affichage de la chaîne de sortie. Ceci peut être réalisé en spécifiant le format de la chaîne de sortie.

Ces chaînes de caractères permettant cette spécification sont semblables à celles utilisées en C (pour les fonctions *printf*, *scanf* ...)

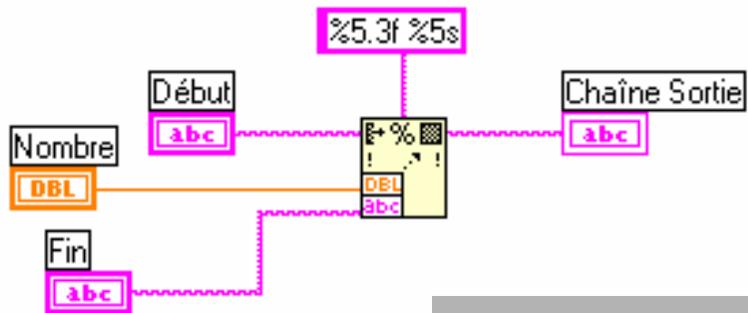
- ✿ Sur le diagramme, faire un 'clic-droit' sur la fonction '*formater en chaîne*'
- ✿ Sélectionner l'option '*Editer la chaîne de format*'



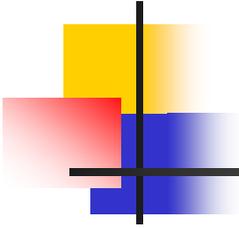
Formater en chaînes

- ☀ Sélectionner '*justifier à droite*', '*combler en utilisant des espaces*' , largeur de champ min=5 et précision=3
- ☀ Sélectionner '*format chaîne*', '*justifier à droite*' , largeur de champ min=5
 - Noter comment la chaîne de formatage est modifiée dans la fenêtre '*chaîne de format correspondante*'
 - Cliquer sur 'OK' et noter que la chaîne générée a été ajoutée au diagramme
- ☀ Lancer l'exécution du VI et observer

Formater en chaînes

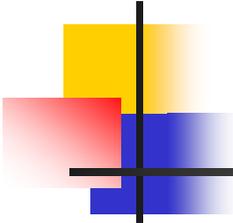


Début	<input type="text" value="abc"/>
Nombre	<input type="text" value="2,00"/>
Fin	<input type="text" value="def"/>
Chaîne Sortie	<input type="text" value="abc2,000 def"/>



Formater en chaînes

- ✿ On peut remarquer qu'il n'est pas possible de modifier l'affichage du texte 'Début' (chaîne d'entrée)
- ✿ Ceci peut être quand même réalisé en considérant le texte 'Début' comme une entrée supplémentaire et en laissant la chaîne d'entrée vide



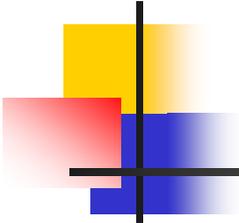
Extraire d'une chaîne

☀ Il est également possible d'extraire d'une chaîne de caractères des éléments individuels qui composent cette chaîne

☀ Créer un nouveau VI

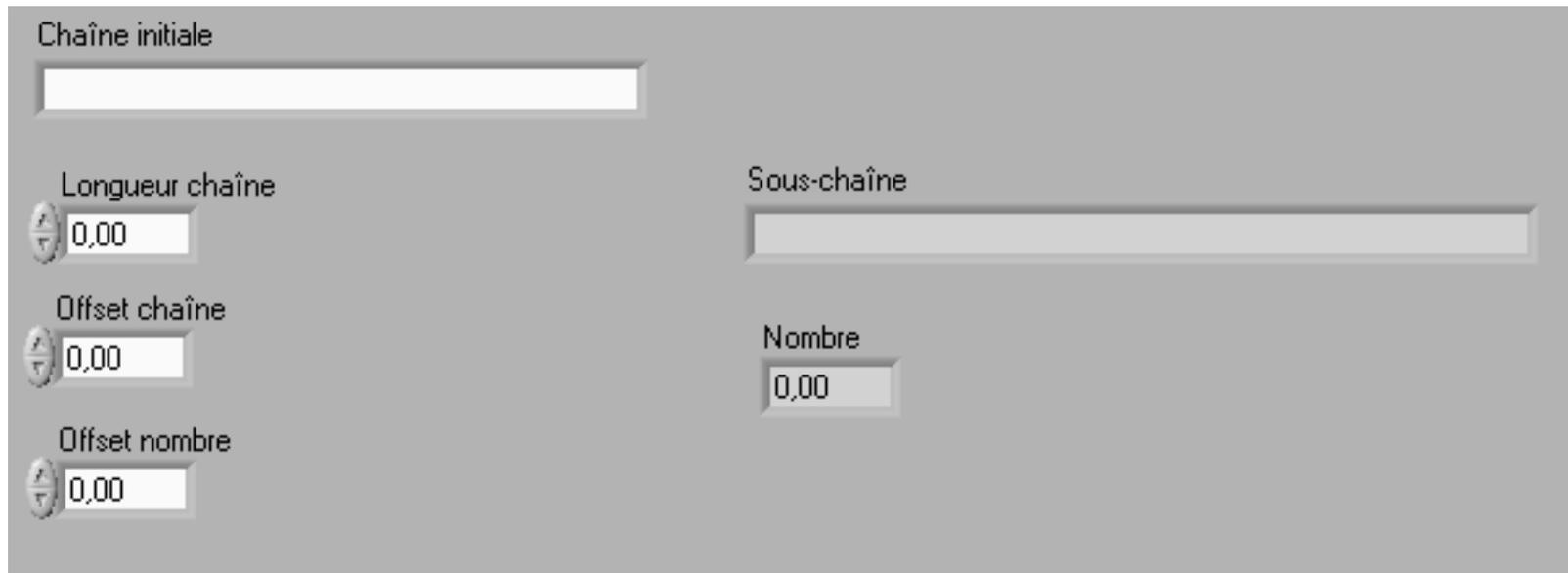
☀ Sur la face-avant créer:

- 3 commandes numériques 'Longueur chaîne', 'Offset chaîne' et 'Offset nombre'
- 1 commande chaîne 'Chaîne initiale'
- 1 indicateur chaîne 'Sous-chaîne'
- 1 indicateur numérique 'Nombre'



Extraire d'une chaîne

On obtient une face d'entrée de ce type:



Chaîne initiale

Longueur chaîne

Offset chaîne

Offset nombre

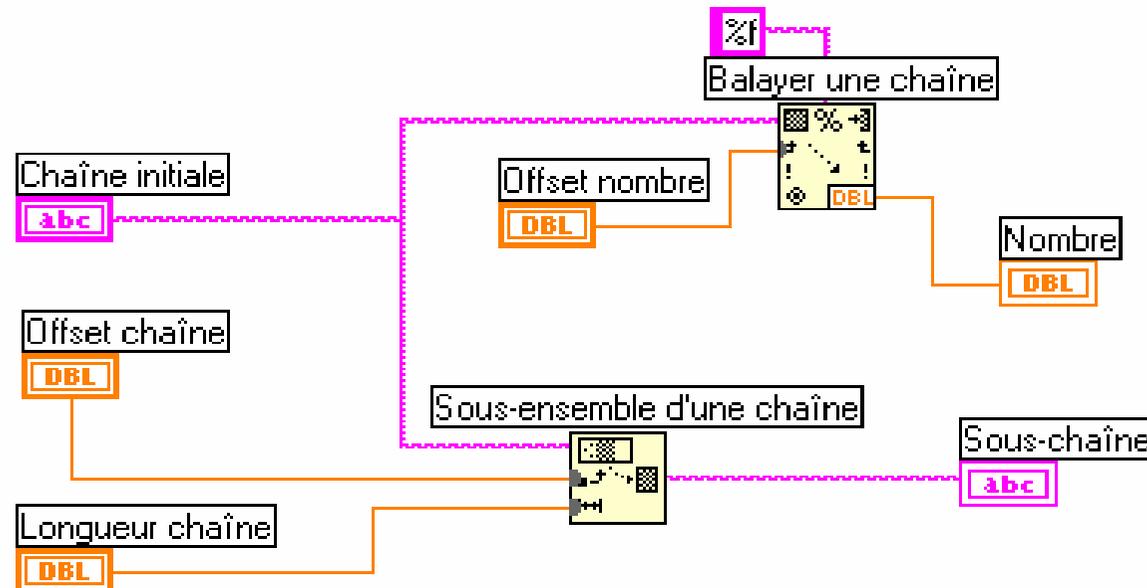
Sous-chaîne

Nombre

Extraire d'une chaîne

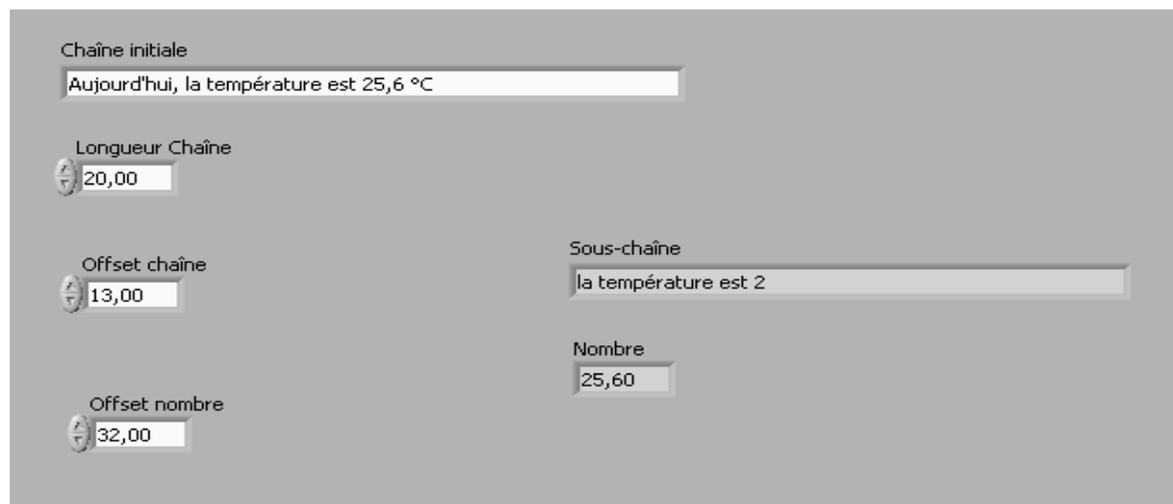
☀ Sur le diagramme créer:

- Une fonction '*Balayer une chaîne*'
- Une fonction '*Sous-ensemble d'une chaîne*'
- Faire le diagramme suivant:



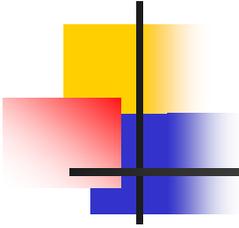
Extraire d'une chaîne

- ☀ Entrer un texte dans 'Chaîne initiale' (ex: Aujourd'hui, la température est de 25,6 °C)
- ☀ Lancer l'exécution du VI en continu
- ☀ Modifier les différents paramètres 'Longueur chaîne', 'Offset chaîne' et 'Offset nombre'



The screenshot shows a control panel with the following fields:

- Chaîne initiale:
- Longueur Chaîne:
- Offset chaîne:
- Offset nombre:
- Sous-chaîne:
- Nombre:



Extraire d'une chaîne

- ✿ On voit ici que l'extraction du nombre 25,6 dépend fortement de la valeur de 'Offset nombre'
- ✿ Pour éviter l'utilisation de cet 'offset', on peut également balayer chaque sous-chaînes existantes entre chaque espace
- ✿ Cependant, cette technique nécessite de connaître à l'avance le contenu de la chaîne

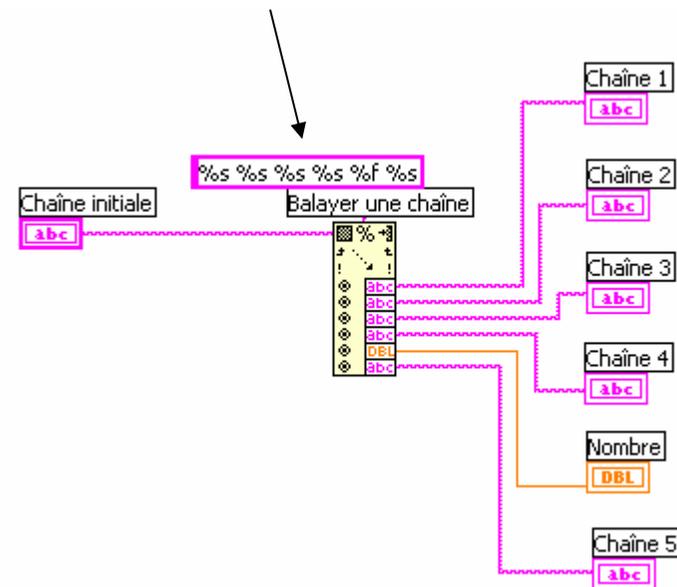
Extraire d'une chaîne

Chaîne initiale
Aujourd'hui, la température est 25,6 °C

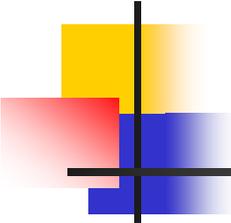
Chaîne 1: Aujourd'hui, Chaîne 2: la Chaîne 3: température

Chaîne 4: est Nombre: 25,60 Chaîne 5: °C

Cette chaîne de format définit les différents 'mots' de la chaîne



Remarque: il n'est pas nécessaire de connecter toutes les sorties à un indicateur.



Fichiers entrées/sorties

☀ Il y a trois types de fichiers:

- Fichiers ASCII (qui peuvent être lus par d'autres logiciels comme Excel, Origin...)
- Fichiers datalog (format propre à Labview et qui peut contenir de multiple formats de données)
- Fichiers binaires (les plus rapides et petits)

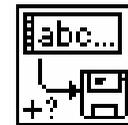
☀ Dans ce cours, nous nous limiterons aux fichiers ASCII

Fichiers E/S ASCII

☀ 5 fonctions basiques sont disponibles, chacune pouvant être trouvées dans le sous-menu '*E/S sur fichiers*' dans palette '*Fonctions*' :

- '*Écrire des caractères dans un fichier*'

écrire une chaîne dans un nouveau fichier ou ajouter une chaîne dans un fichier déjà existant



- '*Lire des caractères dans un fichier*'

Lire un nombre spécifié de caractères dans un fichier



Fichiers E/S ASCII

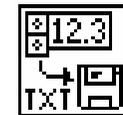
- '*Lire des lignes dans un fichier*'

Une ligne étant définie par un retour chariot ou caractère spécifique (EOF)



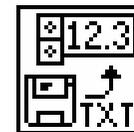
- '*Écrire dans un fichier tableur*'

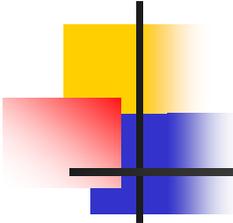
Converti un tableau 1D ou 2D en une forme lisible par un logiciel tableur (ex: excel)



- '*Lit dans un fichier tableur*'

Lit un nombre spécifié de lignes à partir d'un fichier numérique et places les données en tableaux 2D





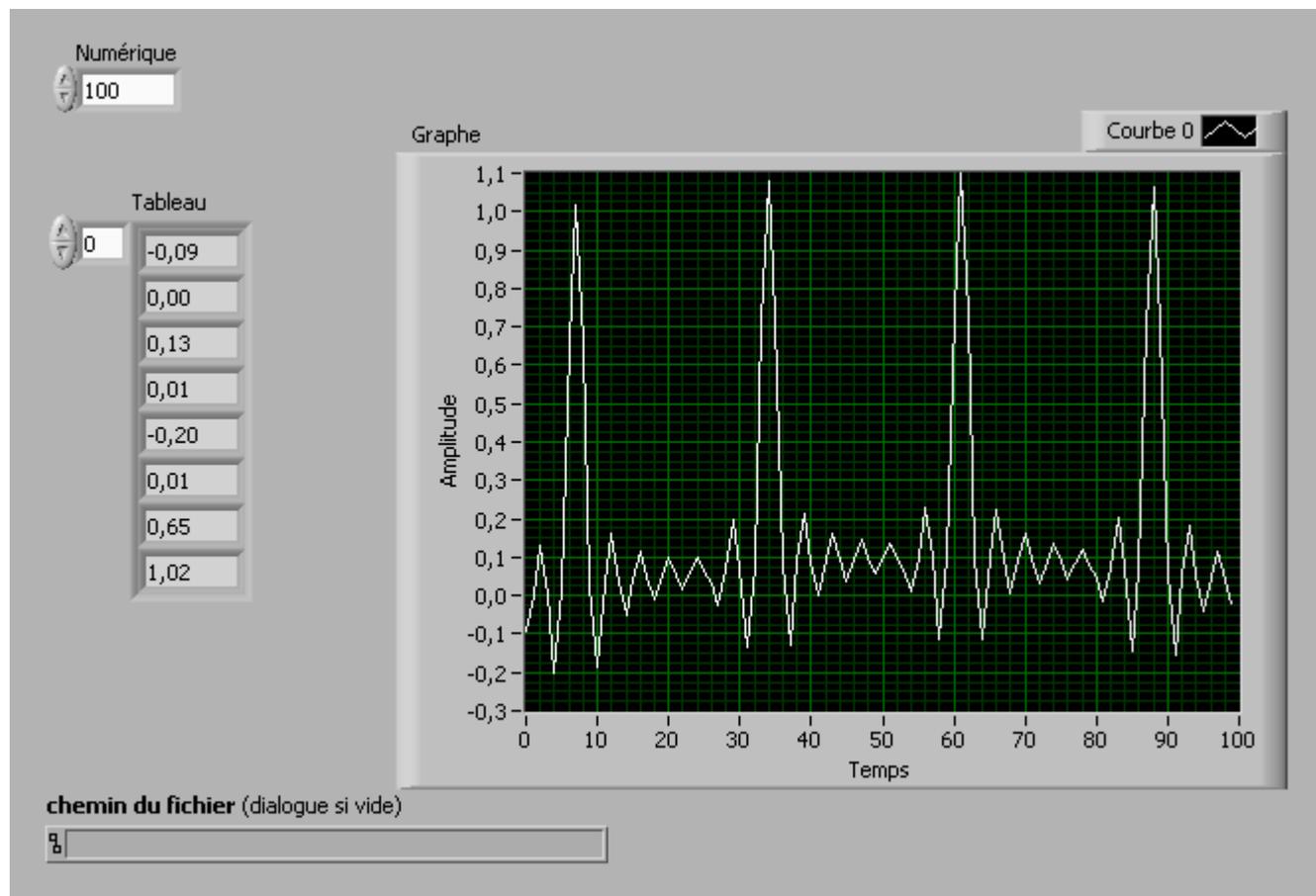
Fichiers E/S ASCII

✿ Exercice:

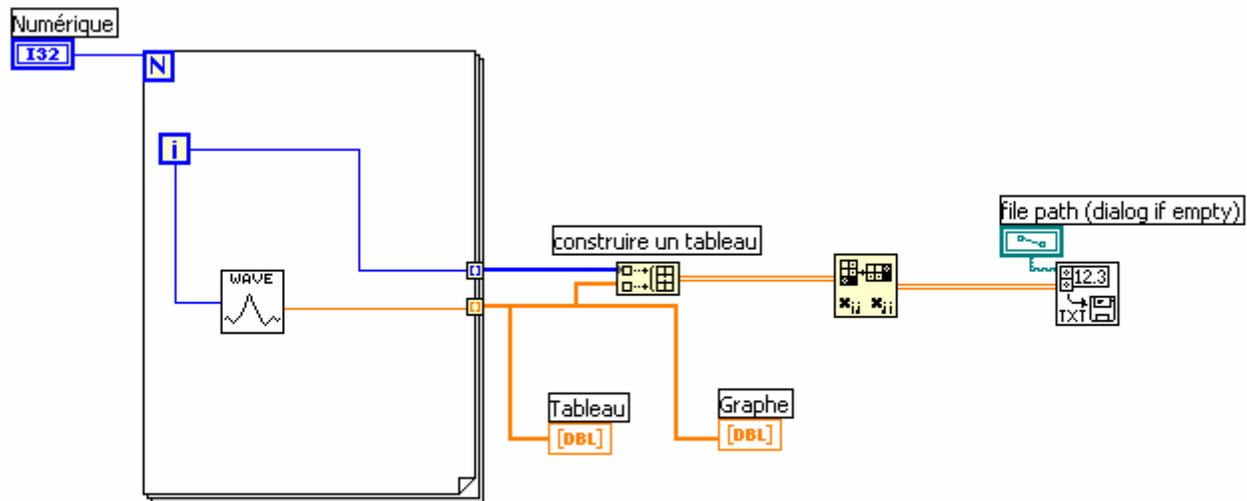
- Créer un nouveau VI
- Créer un tableau, un graphe et une commande numérique
- Utiliser la fonction '*Générer un signal*' (répertoire Labview6\activity)

L'objectif de cet exercice est de générer un signal avec la fonction, d'afficher le contenu sur un graphe (et dans un tableau 1D de 100 points), et de sauver le résultat dans un fichier tableur sur 2 colonnes: n° point | valeur de la fonction

Fichiers E/S ASCII

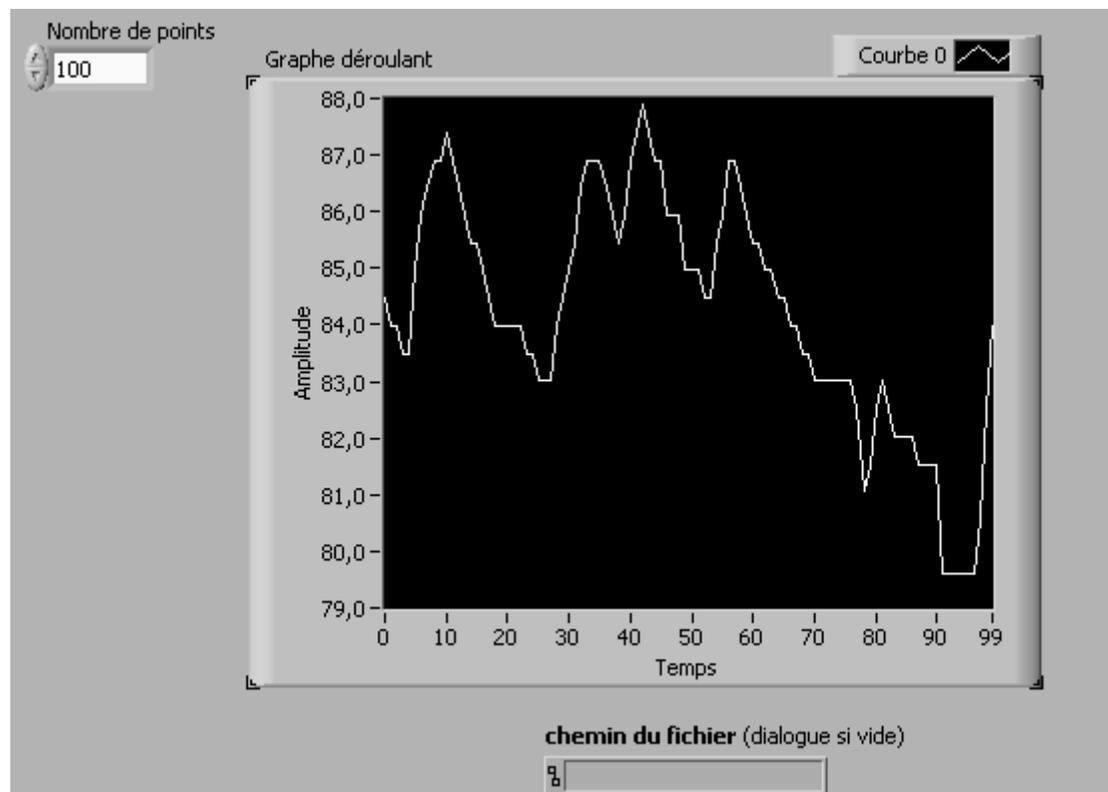


Fichiers E/S ASCII

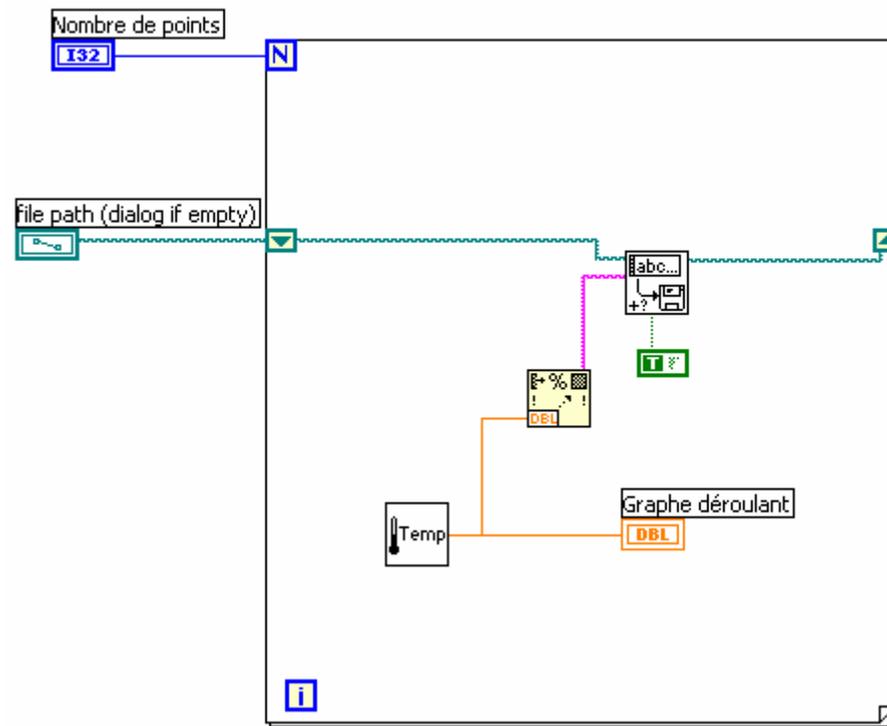


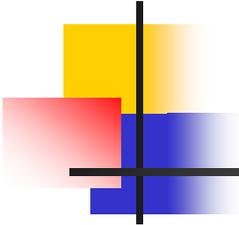
Fichiers E/S ASCII

Un autre exemple....



Fichiers E/S ASCII

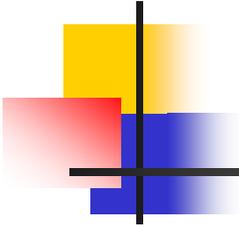




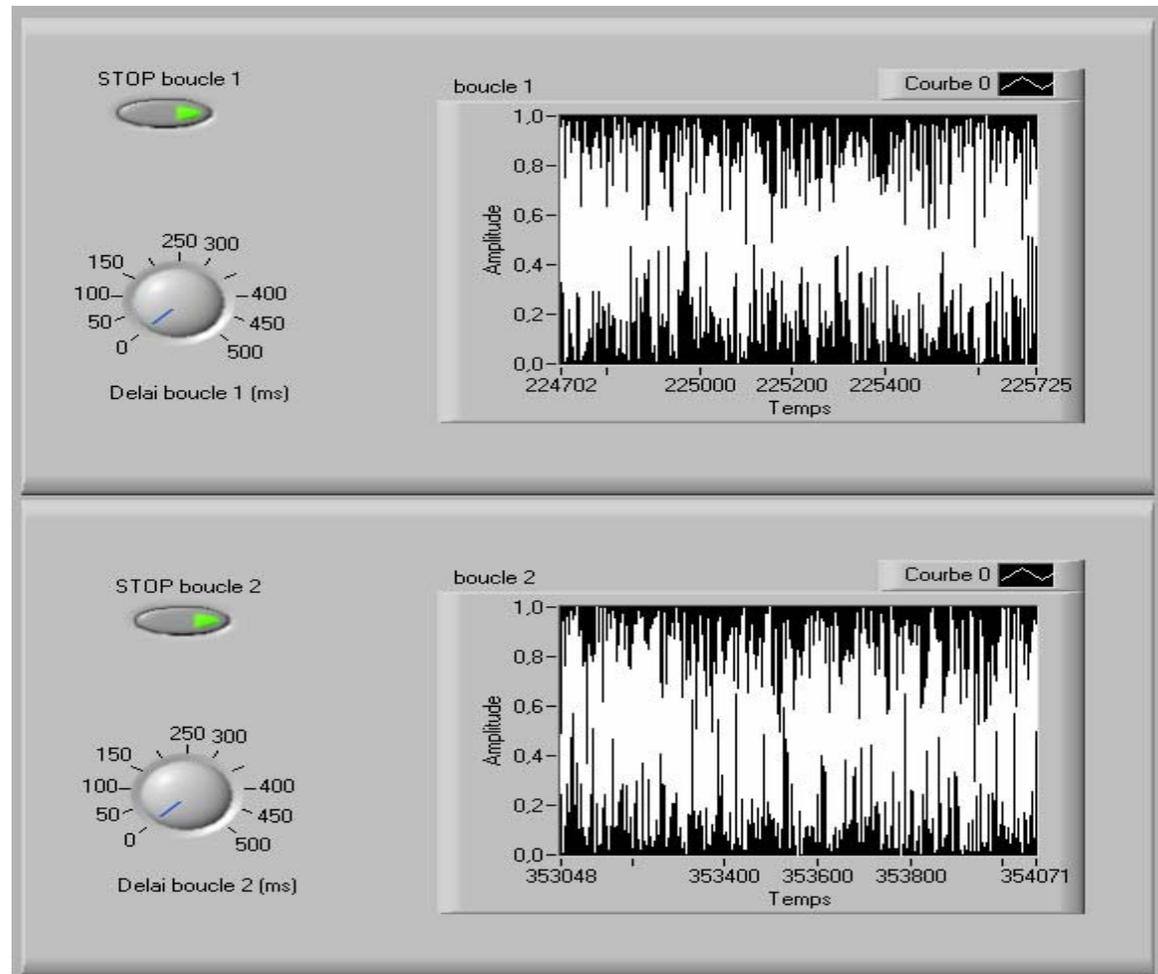
Multi-threading

Dans cette partie, nous présenterons le multifenêtrage sous Labview

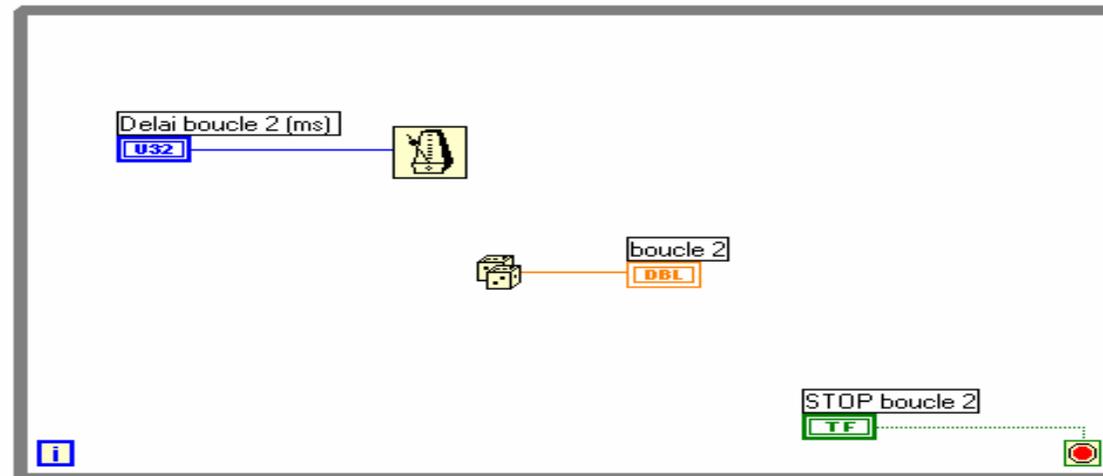
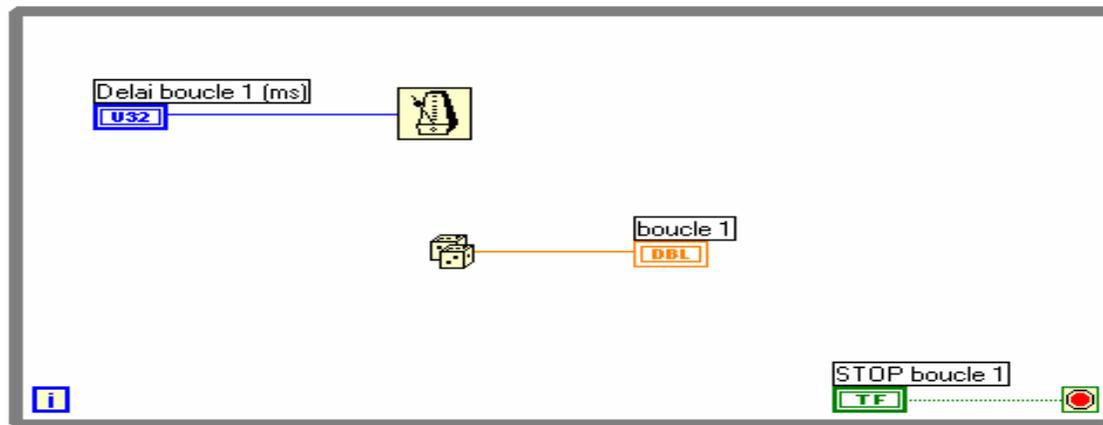
- ✿ Ouvrir le fichier '**multithread1.vi**'
- ✿ Lancer l'application

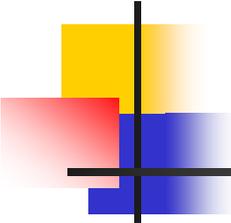


Multi-threading



Multi-threading





Multi-threading

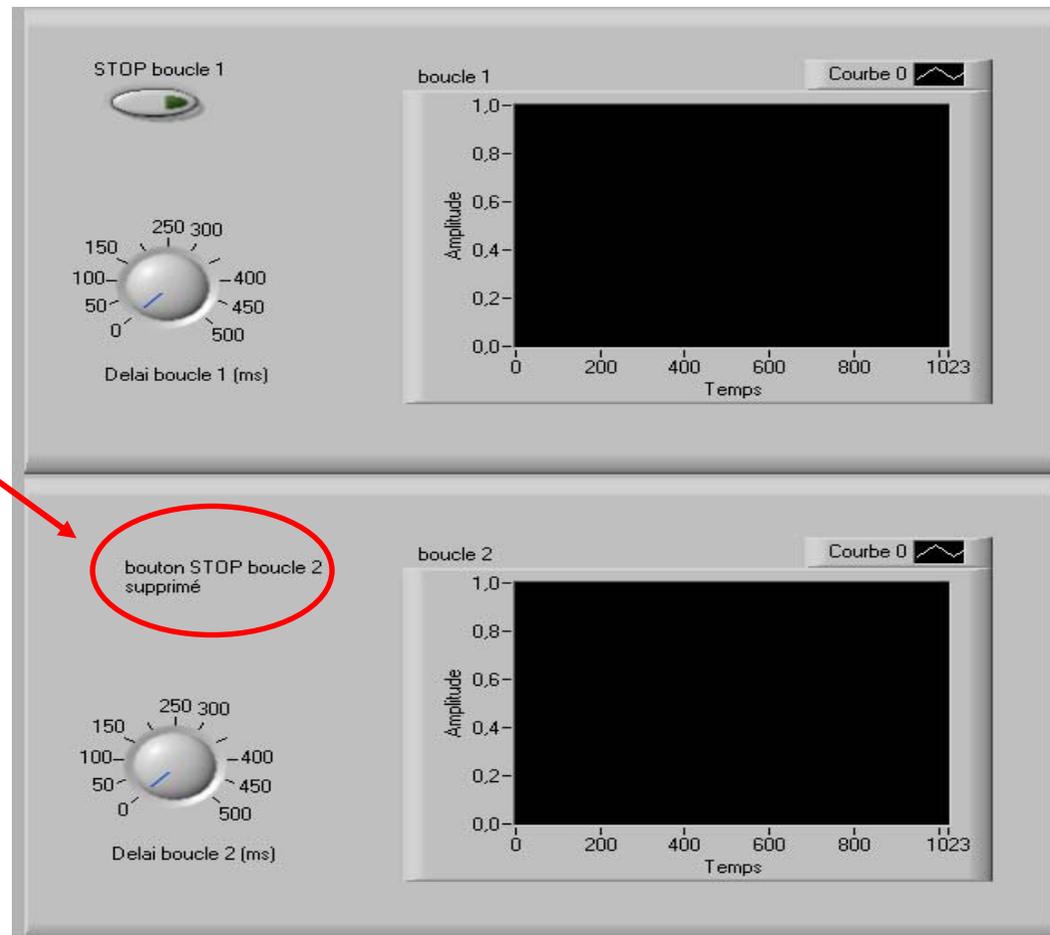
- ✿ La vitesse de chaque boucle peut être contrôlée
- ✿ Si on appuie sur un des boutons '**STOP**' :
 - La boucle concernée s'arrête sans altérer le fonctionnement de l'autre boucle
 - MAIS, il n'est pas possible de redémarrer la boucle sans arrêter l'autre...



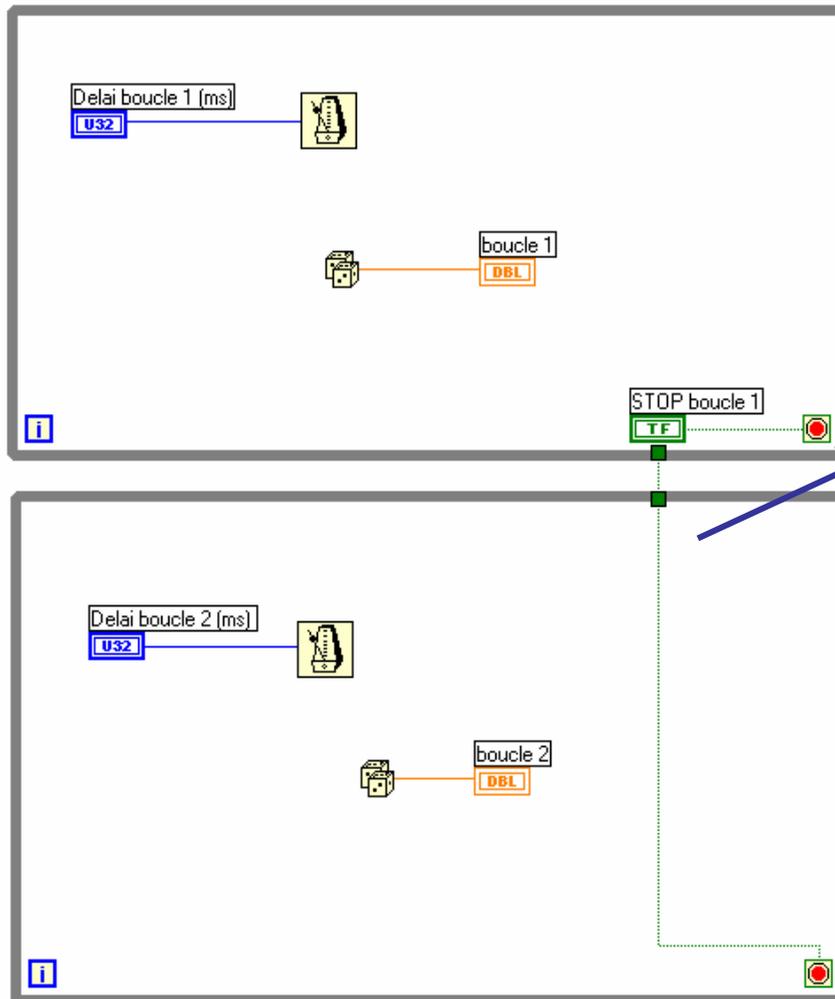
Modifions maintenant le VI de manière à avoir un seul bouton commandant l'arrêt des 2 boucles

Multi-threading

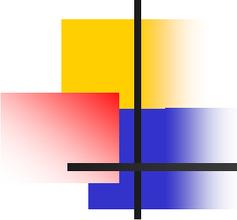
Supprimer le
2^{ème} bouton
STOP



Multi-threading



Connecter le
1^{er} bouton à la
condition de
sortie de la
2^{ème} boucle



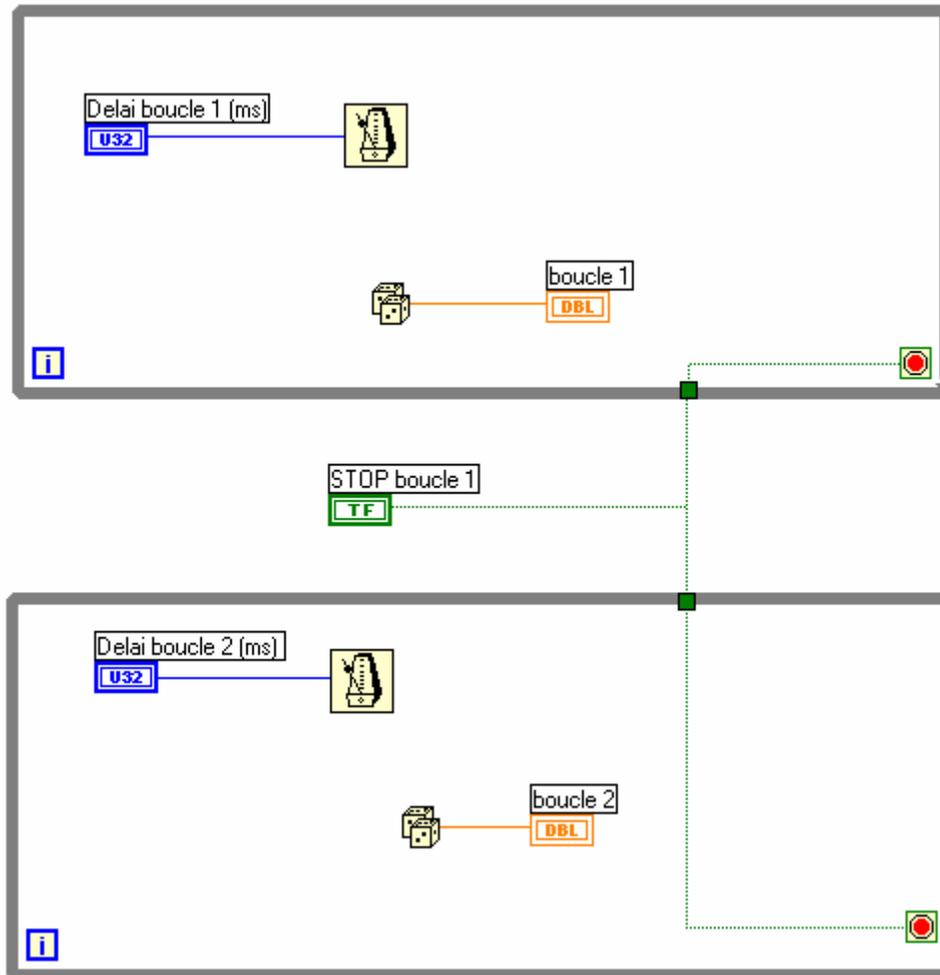
Multi-threading

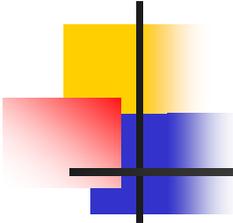
Même si cette connexion est acceptée, le programme ne s'exécute pas correctement (la boucle 2 ne démarre jamais)



Essayons une autre possibilité : placer le booléen en dehors des deux boucles

Multi-threading





Multi-threading



Les deux boucles fonctionnent, mais on ne peut pas les stopper...

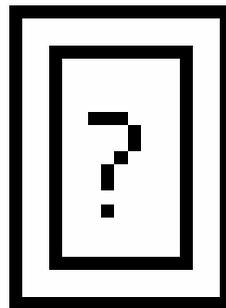
Pour résoudre ce problème, nous allons utiliser des variables locales ce qui permettra de rendre accessible la variable '**STOP bouton1**' dans les deux boucles

2 façons de créer une variable locale:

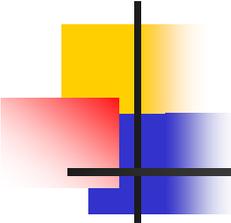
- Sous-menu '*Structures*' de la palette '*Fonctions*'
- 'Clic-droit' sur la commande ou l'indicateur de la variable

Variables locales

✿ Si on crée la variable locale à partir du sous-menu '*Structures*', celle-ci apparaîtra sous la forme suivante:



✿ Ceci indique qu'aucune variable n'a été assignée pour l'instant à la variable locale

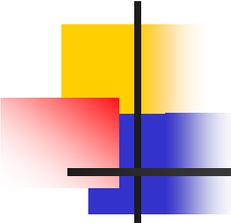


Variables locales

- ✿ Pour réaliser cette assignation, il suffit de faire un 'clic-droit' sur l'objet et de choisir la variable dans le menu '*sélectionner un élément*'
- ✿ Sélectionner un élément entraînera le fait que la variable locale prendra le nom et le type de l'élément sélectionné

Dans notre exemple, la variable locale prendra le nom '**STOP boucle 1**' et sera de type booléen:

STOP boucle 1



Variables locales

✿ Il n'y a pas de limites au nombre de variables locales que l'on peut associer à une variable

✿ Une variable locale peut être de deux types:

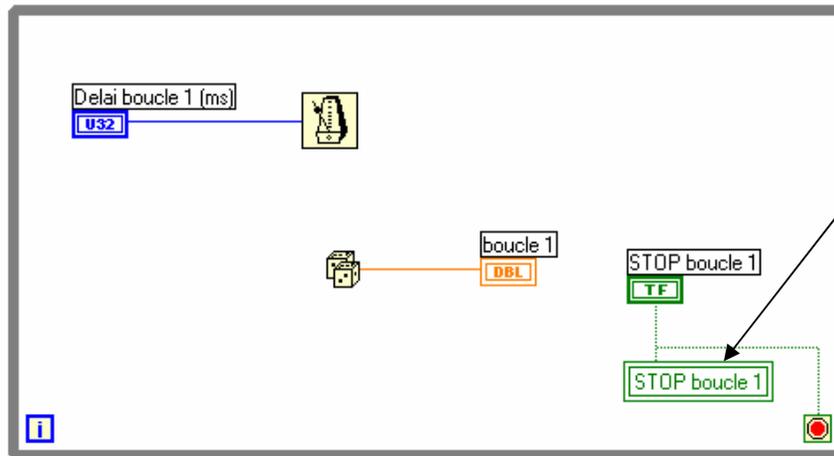
- Variable locale en lecture
- Variable locale en écriture

✿ On peut modifier le type lecture/écriture par un 'clic-droit' sur la variable locale



Reprenons maintenant notre VI 'multithread.vi'

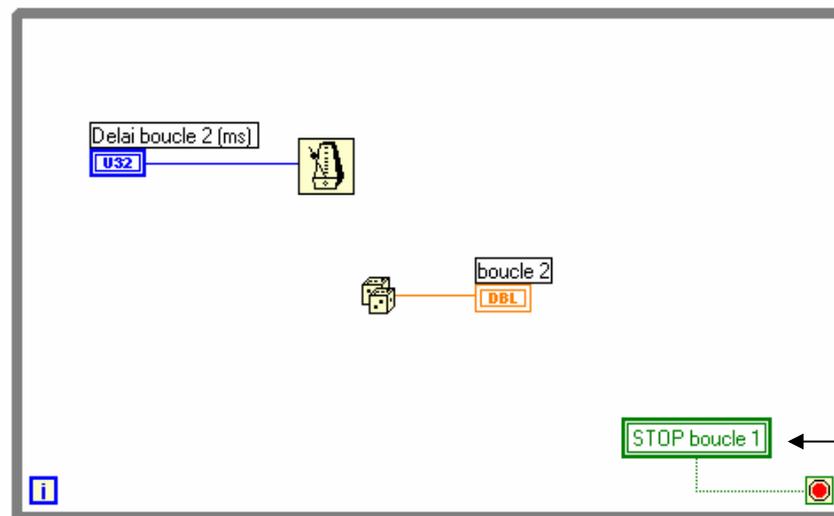
Multi-threading



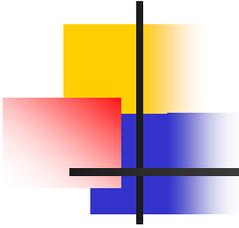
Variable locale en écriture



Les 2 boucles sont maintenant contrôlées par le même bouton STOP



Variable locale en lecture

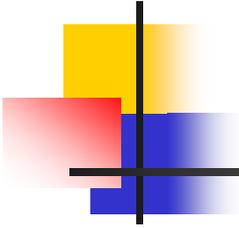


Variables globales

Nous allons maintenant introduire la notion de variables globales

✿ Une variable globale agit de façon similaire à une variable locale excepté le fait que l'on peut transférer cette variable d'un VI à un autre

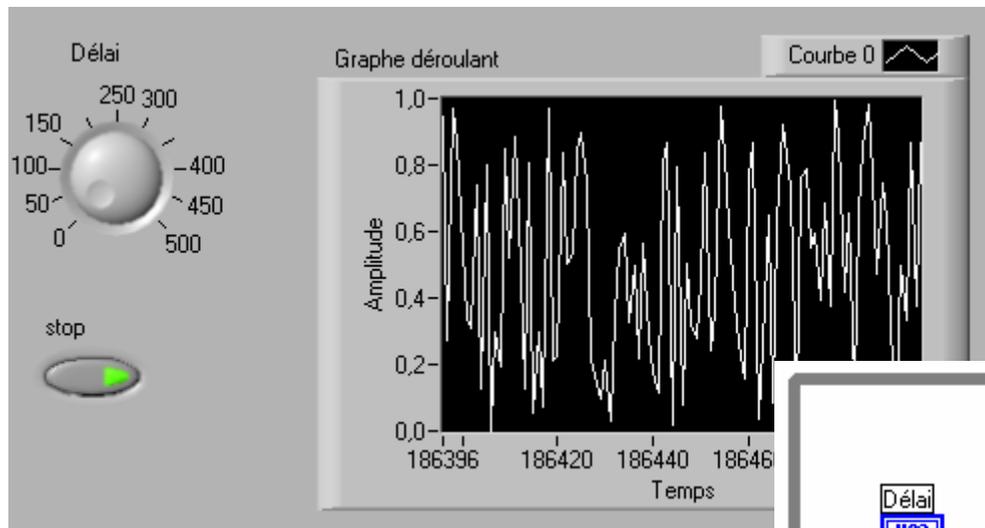
✿ Nous allons voir cette application à travers un exemple similaire au précédent



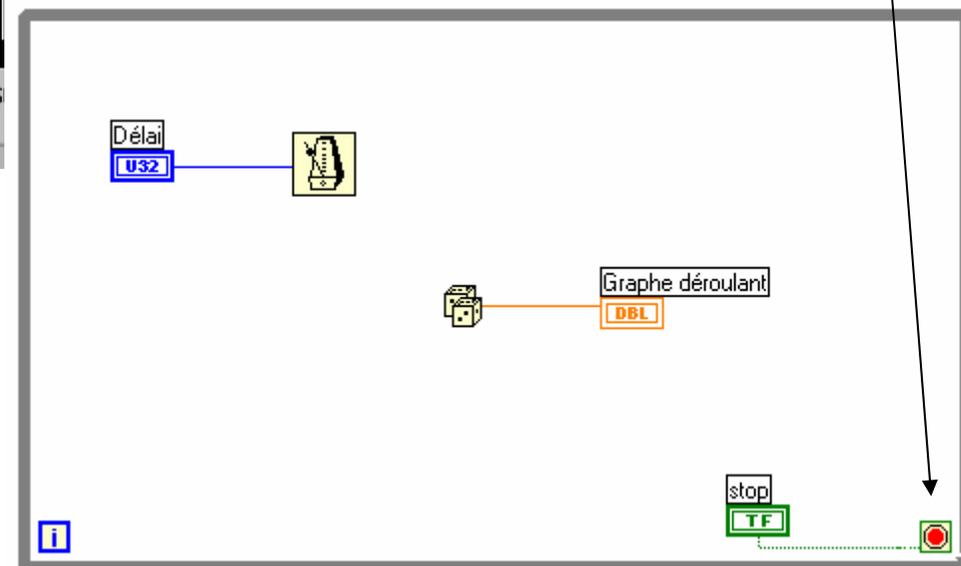
Variables globales

- ✿ Créer un nouveau VI '**boucle1.vi**'
- ✿ Sur la face d'entrée, créer:
 - un bouton-poussoir (booléen) '**stop**'
 - un graphe déroulant (échelle Y: 0...1)
 - un bouton rotatif '**délai**' (U32, échelle 0...500)
- ✿ Sur le diagramme, créer:
 - une boucle WHILE
 - un générateur de nombre aléatoire
 - une fonction '*attendre un multiple de ms*'

Variables globales



S'arrêter sur condition vraie



Variables globales

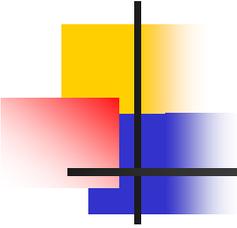
☀ Sur le diagramme, créer:

- une variable globale à partir du sous-menu '*structures*' et la placer à côté de la variable '**stop**'

- On doit avoir à l'écran le symbole suivant:



- double cliquer sur l'icône (ce qui ouvre une nouvelle fenêtre)
- ajouter un bouton-poussoir booléen '**gStop**'
- sauver le VI sous le nom '**stop_global.vi**'
- fermer le VI

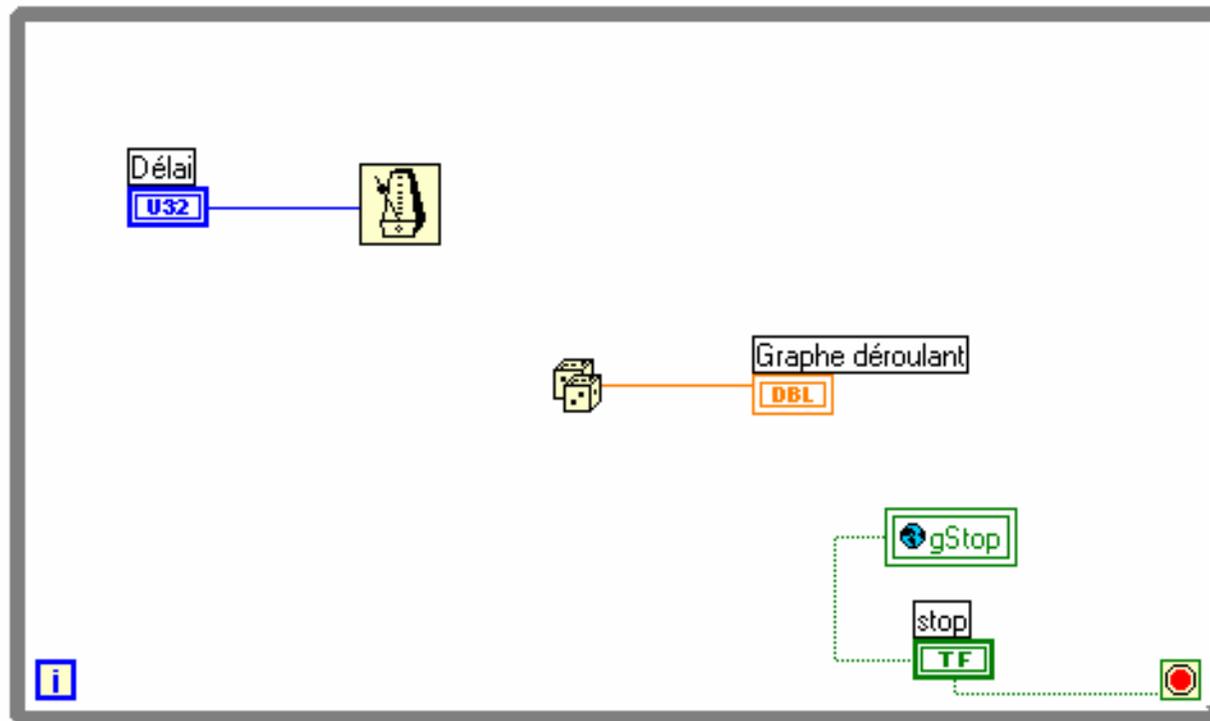


Variables globales

- ✿ Sur le diagramme de 'boucle1.vi' :
 - 'clic-droit' sur la variable globale
 - sélectionner l'élément 'gStop'
 - vérifier que la variable soit en écriture
 - connecter 'stop' à la variable globale 'gstop'

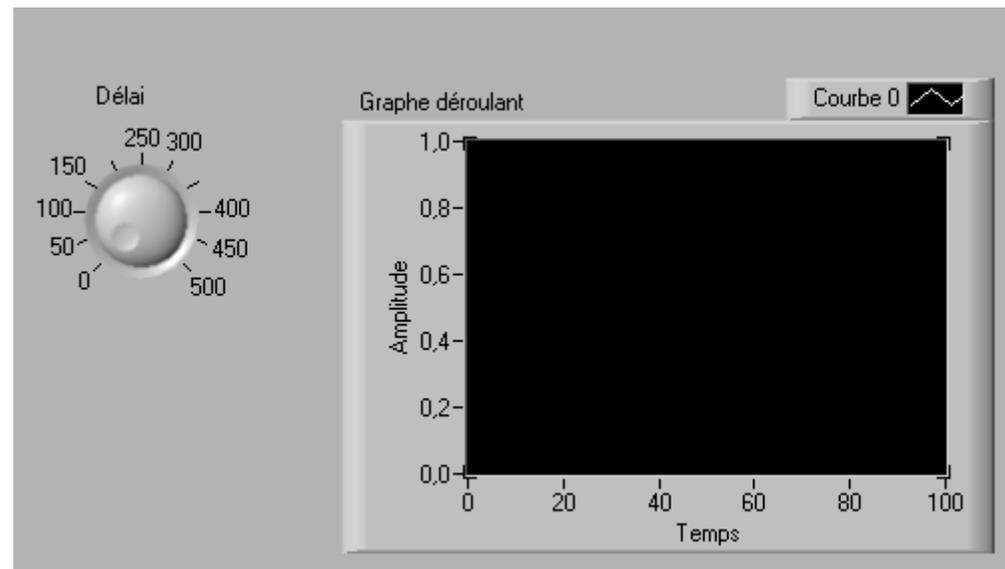
Variables globales

Diagramme de 'boucle1.vi' :



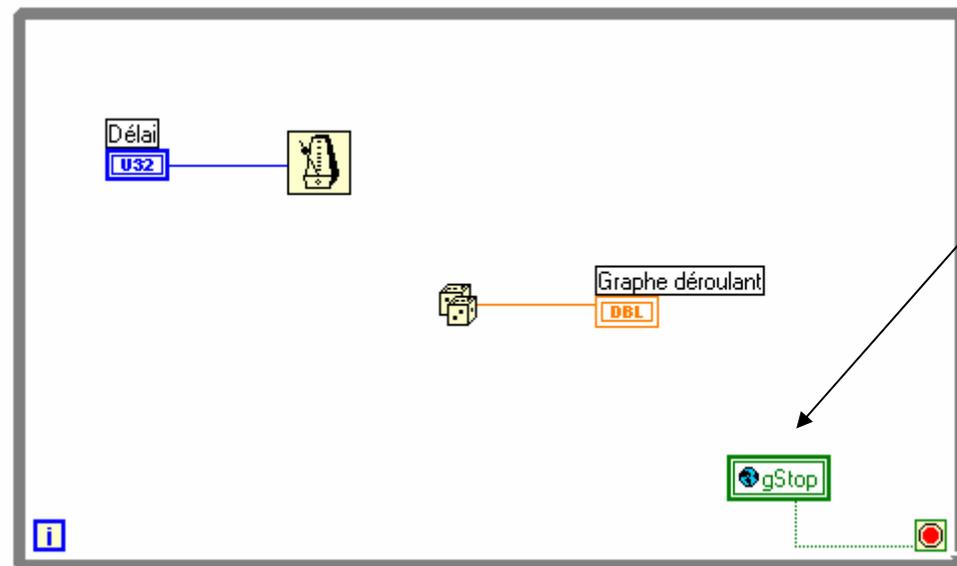
Variables globales

☀ Faire un nouveau VI 'boucle2.vi', semblable à 'boucle1.vi' sans le bouton 'stop':

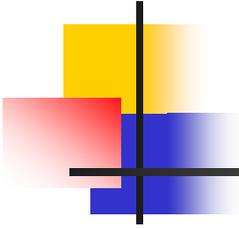


Variables globales

Diagramme de 'boucle2.vi' :



Variable globale en lecture



Variables globales

- ✿ Lancer les deux VI '**boucle1.vi**' et '**boucle2.vi**'
- ✿ comme précédemment, on peut contrôler de manière indépendante la vitesse de chaque boucle
- ✿ Si on presse sur le bouton '**stop**' de '**boucle1.vi**', on arrête l'exécution des deux Vis

Remarque: Il n'est pas possible de démarrer '**boucle2.vi**' avant d'avoir lancé '**boucle1.vi**' ... Pourquoi???