Introduction

Labview (Laboratory Virtual Instrument Engineering WorkBench) est un logiciel de développement de programmes d'application. LabView utilise un langage de programmation essentiellement ***graphique*** dédié au contrôle, à l'acquisition, l'analyse et la présentation de données. En LabView, on n'écrit pas de lignes de programme dans un langage textuel comme Pascal ou C, Basic ou Fortran. On manipule des objets **graphiques.** Ces objets graphiques représentent à la fois les variables du programme, ainsi que des fonctions qui vont réaliser des actions portant sur ces variables. La programmation en Labview consiste simplement à concevoir le traitement de l'information, organiser et relier les variables avec les fonctions au moyen de fils.

Labview est dédié à la programmation conçue pour le pilotage d'instruments électronique! Avec Labview on construit graphiquement des modules logiciels appelés des « VI » (« VI » sigle de « Visual Instruments ») au lieu d'écrire du code dans un langage informatique textuel.

Son principe de programmation est basé sur l'assemblage graphique de modules logiciels appelés «Instruments Visuels (« VI »). Le rôle d'un VI est d'acquérir des données issues par exemple de fichiers, du clavier ou encore de cartes électroniques d'Entrée/Sorties », de les analyser, et de les présenter au travers d'interfaces hommes-machines graphiques (encore appelées « face avant » par analogie avec la face avant permettant de piloter un appareil électronique). Dans Labview, ce qu'on appelle la « face avant » est donc l'interface utilisateur permettant d'exploiter, de piloter, le programme.

Présentation - Environnement du progiciel

Lorsqu'on lance Labview, on obtient les fenêtres de base suivantes:

Fenêtre Face Avant ("Front panel") et Palette de Commandes ("Control palette")

1. Le Front Panel représente l'interface utilisateur du programme VI que l'on va écrire. C'est un dessin de la face avant de l'appareillage que l'on programme.

2. La « Control Palette » n'a d'utilité que vis à vis de la fenêtre « Front Panel ». Cette palette propose un choix de divers objets graphiques portants sur des structures de données différentes. On trouvera de quoi composer des « faces avants » d'instruments, constituées d'objets numériques, booléens, chaînes de caractères, tableaux, et divers objets de visualisation graphiques,etc?

Les différentes options de la « palette de contrôle » sont (de droit à gauche et de haut en bas) :

- Les indicateurs et contrôles associés aux données numériques (thermomètres, vumètres, jauges..)
- Les indicateurs et contrôles booléens (interrupteurs, bouton poussoir, LED..)
- Indicateurs et contrôles de chaînes de caractères
- Les indicateurs et contrôles liés aux menus déroulants
- Les indicateurs et contrôles associés aux tableaux et clusters (regroupement de données sous la forme d'une seule variable)
- Les graphes : permettent de représenter différents type de graphes possibles
- etc

Pour constituer une « face avant » d'appareil, on sélectionnera un objet dans la « control palette » que l'on glissera et déposera sur la fenêtre « Front Panel ». *Chaque objet déposé sur le « front panel » aura une correspondance dans la fenêtre « Diagram » que l'on verra ci dessous*

Le Front Panel est donc la « face avant » de l'appareil que vous programmez. Elle fera donc fonction d' interface utilisateur, correspondant au programme écrit dans la fenêtre « Diagram » décrite ci dessous. On testera et on exécutera le programme à partir de cette fenêtre.

Fenêtre Diagramme et Palette de Fonctions ("function palette")

La fenêtre « Diagramme » est composée des variables associées aux objets que l'on a déposé dans le « Front Panel ». La fenêtre Diagramme contient donc le coeur du programme VI. C'est dans cette fenêtre que l'on va programmer effectivement et relier les objets entre eux au moyen de fonctions issues de la «Function palette ». La fenêtre « Fonctions » n'a de sens et d'utilité que vis à vis de la fenêtre « Diagramme »

Ces 2 fenêtres servent à construire le programme à l'aide de modules graphiques. La «function palette » contient un certain nombre d'objets graphiques. Les icônes de programmation seront toutes sortes de fonctions arithmétiques, d'autres programmes VI d'acquisition, des fonctions portants sur les tableaux de données et chaînes de caractères, ainsi que de formatage des entrées/sorties.

Pour écrire/composer son programme Labview, on sélectionnera certains composant de ce menu de fonctions, et on les glissera sur la fenêtre Diagram.

La fenêtre « Diagram » va donc contenir en quelque sorte le « code source » du programme VI. On va assembler les modules graphiques issus de la « Function palette » dans cette fenêtre pour constituer le programme VI.

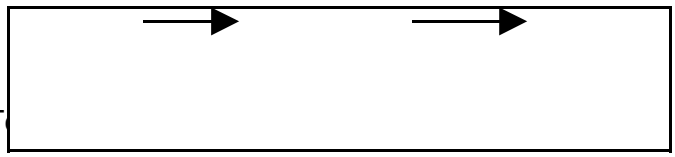
On comprend donc que pour écrire un programme VI, il n'y a donc ni syntaxe d'un langage informatique textuel, ni étape de compilation pour produire un programme exécutable, comme cela pourrait être le cas avec des langages traditionnels comme « Pascal », « C », ou encore Fortran.

Ecrire un programme VI c'est donc choisir des modules graphiques dans la « function palette », les placer sur la fenêtre « Diagram », et les connecter entre eux au moyen de « fils de liaison ». Ces fils de liaison sont obtenus par l'icône « bobine » de la « Tool palette » (fenêtre d'outils). Les fils de liaisons permettent de transmettre les données d'un composant graphique (icône) vers un autre (typiquement d'une variable vers un opérateur ou une fonction).

En Labview on réalise donc une programmation par « flux de données » entre des modules graphiques.

Remarque : Afin de faciliter la visualisation et la lecture d'u programme,

- on prendra soin à la disposition des variables et fonctions sur le diagramme
- on déposera les objets sur le « front panel » et le "diagramme", selon une séquence logique allant de la **gauche vers la droite**.



Fenêtre outils ("T

C'est un menu d'outils pour manipuler les objets graphiques que l'on a placé dans le «Diagram ». Voici ses principales fonctions

- *La main : sert à manipuler le contenu des variables de contrôle en cliquant sur les curseurs*
- *La flèche : sert sur le front panel à déplacer les objets graphiques? uniquement « déplacer »*
- *La lettre A : set à pouvoir écrire une chaîne de caractères.. un label de variable par exemple ou une valeur dans une variable*
- *La bobine : fonction ESSENTIELLE de la programmation en LabView puisqu'elle sert à relier les objets graphiques avec les fonctions de traitement*

La fenêtre Aide « Help » :

Elle fournit de l'aide « en ligne » sur les objets graphiques et les fonctions que l'on utilise. D'une manière générale il est conseillé de laisser la fenêtre d'aide en ligne ***ouverte en permanence*** ! ! ! Cette fenêtre permet d'obtenir l'aide relative à une fonction simplement en cliquant sur l'icône de la fonction. On obtient cette fenêtre par le menu *Help/Show Help.*

Création d'un VI simple

Afficher l'état d'une LED en fonction de la comparaison de 2 valeurs numériques, vert si les valeurs sont égales, rouge dans le cas contraire

1. Création de la face avant : Sur ***Front panel et Control palette*** :
   - A partir du menu « Palette », choisir, glisser et déposer 2 variables numériques simples
   - leur donner des « labels » pour nommer ces variables val1 et val2 par exemple
   - choisir une variable de type booléen « square light » dans la fenêtre « control » par exemple qui aura pour valeur vrai./faux ce qui correspondra à égal/différent.
   - lui donner un nom : LED par exemple

2. Dans la fenêtre Diagram, à l'aide du menu « Functions »
   - choisir une fonction de comparaison d'égalité
   - la glisser sur le diagram
   - à l'aide du menu « tools » choisir la bobine de fils
   - relier la sortie de la variable 1 à l'entrée 1 de la fonction de comparaison
   - relier la sortie de la variable 2 à l'entrée 2 de la fonction de comparaison
   - relier la sortie de la fonction de comparaison à l'entrée de la variable booléenne LED
   - retirer… c'est prêt ! le programme VI est terminé
   - n'oubliez pas d'enregistrer ce programme dans votre répertoire personnel en lui donnant un nom led.vi par exemple

3. Colorier la LED selon la comparaison des 2 nombres
   - On souhaite faire prendre la couleur verte à la variable LED lorsque les nombres entrés dans les variables val1 et val2 sont égaux, et rouge dans le cas contraire. Par défaut la variable booléenne est initialisée à « vrai »? Avec l'outil « pinceau » du menu « Tools » choisir une couleur verte et la déposer sur la LED, puis avec la « main » du menu tools, cliquer sur la LED pour la faire changer d'état ; en déposer une couleur rouge avec le pinceau.

4. Tester, exécuter le programme
   - Sélectionner la « main » du « Tool menu » qui permet de rentrer des nombres dans les variables numériques val1 et val2. Cliquez sur la « flèche » du « Diagram » ou du « Front panel » pour exécuter le programme

Remarque 1 : on remarque que les fils de câblage ***sont de couleurs différentes*** selon le type des données qu'ils véhiculent :

- Orange pour les données de type réel
- Bleu pour les données de type entier
- Vert pour les données de type booléens

Remarque 2 : veillez à nommer correctement vos variables. Les variables ont un label qui s'affiche au dessus. Il est bon de donner un nom sémantiquement parlant pour désigner une variable.

Remarque 3 : Un VI ne peut être exécuté que lorsque toutes les entrées ont été reliées et sont actives. LabView vérifie les connexions des éléments du diagramme en continu. Si un élément a été mal cablé LabView génére une erreur. La flèche correspondant au lancement du programme est « brisée », le programme ne peut pas s'exécuter. Pour comprendre une erreur, ou pour avoir plus d'information sur la nature d'une erreur, il suffit de sélectionner l'option « liste des erreurs » dans le menu « fenêtre », ou plus simplement de lancer l'exécution du programme en cliquant sur la flèche brisée.

Les variables

Les classes de variables :

En LabView, on distingue deux classes de variables :

- ***Les Variables de contrôle*** (control variable) : Une variable de contrôle est une variable qui « régit » le déroulement d'un programme. C'est la plupart du temps une variable d'entrée d'un programme, dont l'initialisation influe et conditionne le déroulement du programme subséquent.

- ***Les Variables indicatrices*** (indicator variable) : Une variable indicatrice est une variable qui reçoit une valeur calculée en amont, et qu'on affiche pour connaître (indiquer) un résultat

Typiquement dans l'expression $\boldsymbol{a = b + 2}$

- « a » est une variable « indicatrice » puisque elle reçoit le résultat de l'expression b+2
- « b » est une variable de contrôle
- 2 une constante

Dans l'exemple ci dessus « led.vi », on peut dire que val1 et val2 sont 2 variables de contrôle, alors que la variable booléenne LED qui ne sert juste qu' à afficher le résultat (vrai-vert/faux-rouge) est une variable indicatrice, dans le sens ou elle indique un résultat

modification de la classe d'une variable :

- Après avoir choisi une variable d'une classe quelconque, à partir du menu « Control », on a la possibilité de changer le type d'une variable en cliquant sur la variable avec le *bouton droit de la souris*. On verra à ce niveau une option « *change to indicator* » ou bien *« change to control »* permettant respectivement de définir la variable comme étant une variable de contrôle ou une variable indicatrice.

Les types des variables :

- Comme dans tous les langages de programmation classiques, les variables informatiques ont un type, qui définit leur taille en mémoire et l'usage que l'on peut en faire. Par défaut en Labview, toutes les nouvelles variables créées sont de type « réel » (couleur orange). On peut modifier le type d'une variable en cliquant dessus avec le bouton droit et en sélectionnant l'option « représentation ».
  Un sous menu s'affiche alors avec un ensemble de type de variables différents. On pourra alors choisir et modifier des variables de type entier ou réel

- Le type des données que manipule LabView est semblable à ceux des autres langages informatiques:
  - Données de type réel : de couleur orange en labview,
  - Données de type entier : de couleur bleue en labview
  - Données de type booléen : de couleur verte en labview
  - Données de type « chaîne de caractère » : de couleur violette en labview

Exemple TP2 : modifier l'exemple précédent de la LED

- Rajouter un fonction arithmétique de différence, calculant la différence des 2 valeurs val1 et val2
- Rajouter une variable « indicatrice » de label « diff » à la sortie de la fonction de différence, afin d'afficher la valeur de la différence de val1 et val2
- Rajouter une fonction de comparaison « égal à 0 » de manière à tester si la différence est égale à 0, ou pas
- Relier la sortie de cette fonction de comparaison à 0, à l'entrée de la LED

Exemple TP3 : Moyenne de 3 nombres écrits au clavier
Mode opératoire :

- Sur le front panel, insérer 3 variables de contrôle, et les nommer par un label v1,v2 v3 par exemple
- Faire la somme (prendre une fonction « somme » dans la palette des fonctions) des 2 premières variables, et ranger le résultat dans une variable indicatrice « som1 ». On peut éventuellement (pas obligatoire) ranger cette somme temporaire dans une variable indicatrice somv1v2 par exemple.
- Faire la somme de la troisième variable, avec la somme des 2 premières, et ranger le résultat dans une variable indicatrice
- Insérer une fonction arithmétique « diviser »
- Relier la somme générale à la première entrée de la fonction « diviser » (numérateur)
- « Coller » une constante 3 à la deuxième entrée de la fonction diviser. Pour coller une constante il suffit de cliquer avec le bouton droit de la souris sur une entrée de la fonction et de choisir l'option « *create constant* »
- Ranger le résultat final dans une variable indicatrice de label « moyenne »

Création d'un sous VI

Un « sous VI » est un programme VI, utilisé par un autre programme VI ; C'est l'équivalent des « sous-programmes », procédures et fonctions des autres langages de programmation comme Pascal ou «C ».
Pour créer un sous VI, il faut d'abord créer un VI ! ! Puisqu'un sous VI ne sera ni plus ni moins qu'un VI utilisé dans un autre VI ! Il fallait le dire non ?

- ***Création d'une icône originale***

Après avoir créé un VI, et l'avoir enregistré, cliquer avec le bouton droit sur l'icône du VI courant en haut à droite dans le Front Panel. Sélectionner l'option « Edit Icon »
Il s'agit en fait de créer une icône originale propre à ce VI. On obtient l'éditeur d'icônes avec lequel on peut styliser un quelconque logo. Je vous fais confiance pour trouver quelque chose à dessiner.

- ***Création des entrées et sorties du SOUS VI***

Puisque ce VI va être utilisé comme sous VI dans des contextes de programmation différents, il est nécessaire de définir quels seront ses arguments d'entrées et de sorties. Pour cela, toujours avec le bouton droit de la souris cliquer sur l'icône en haut à droite dans le Front Panel, et sélectionner l'option « Show Connector ». Il s'agit alors de définir les E/S de ce futur sous VI. On clique alors respectivement dans les cases d'entrées de l'icône du sous VI puis sur les variables d'entrées. De cette manière on associe une variable d'entrée avec une entrée de la fonction que l'on crée. On enregistre alors cette nouvelle fonction par « save as ». Elle est créée.

- ***Utilisation d'un sous VI dans un autre VI***

Pour appeler un sous VI dans un VI existant, il suffit de cliquer dans la dernière case du menu Fonctions «Select a VI ». On choisit alors le sous VI que l'on souhaite et on le place sur le diagramme, comme une tout autre fonction.

Exemple TP4 : Création - utilisation d'un sous VI calculant une moyenne de 3 nombres
Mode opératoire :

- Ouvrir le VI précédent qui calculait la moyenne de 3 nombres
- Modifier son icône (en haut à droite du front panel), en choisissant « Edit icon »
- Définissez ses arguments d'entrée et de sortie avec l'option « Show Connector »
- Sauvez le sous un nouveau nom i.e submoy3.vi par exemple
- Créer un nouveau VI, par exemple moy2
- Créez les 3 variables d'entrée et la variable de sortie résultat, comme dans l'exemple précédent
- Dans le menu Fonctions, choisissez « Select a VI » et choisissez le sous VI précédemment créé i.e submoy3.vi. glissez le sur le diagramme
- Relier les objets
- run

Les tableaux et structures de contrôle « boucle »

En informatique, les tableaux sont des ensembles de valeurs de type homogène (entier, réel, caractère), rangées de façon contiguë en mémoire. Ils représentent une portion de la mémoire physique de la machine qui est allouée (réservée) au programme en cours d'exécution. En Labview on n'a pas à réserver une taille fixe de tableau en début de programme, comme on devrait le faire avec un autre langage. Les tableaux Labview sont virtuellement illimités.

Création d'un tableau :

Menu Control /array & cluster, choisir « Array » et le déposer sur le « Front Panel »

Création du type d'un tableau :

En Labview, par défaut, le tableau créé est vide et n'a pas de type de données ! Il faut tout d'abord lui donner un type de données. Pour cela, cliquer sur le menu « Control /Numéric », choisir une variable de type numérique et la déposer sur le tableau dans le FrontPanel. Ceci signifie que le tableau sera composé de valeurs du type identique à la variable que l'on a sélectionné : valeur entière (integer) ou réelle (float)

Modification du type d'un tableau :

Si l'on souhaite modifier le type des valeurs du tableau, on peut cliquer sur le tableau avec le bouton droit de la souris et choisir l'option « Représentation ». On accède alors au choix de plusieurs types de données différents (réel double, simple, entier, etc?)

➔ A présent le tableau a un type, mais il est toujours vide.

Parcours d'un tableau, méthodes d'indexing :

Pour parcourir un tableau, c'est à dire *accéder séquentiellement à chacune de ces valeurs*, on utilise le plus souvent une boucle. Dans Labview les boucles sont situées dans le menu « Functions/Structures ».
Lorsqu'on est censé connaître le nombre d'éléments d'un tableau, on va choisir un boucle «For » (« For Loop ») qui permet de parcourir un tableau en faisant varier un index «i » jusqu'à une valeur maximale « N » représentant le nombre maximal de valeur du Tableau.

```
For (i=0 ; i<N ; i++) {
        Valeur=tab[i] ;
}
```

Lorsqu'on ne connaît pas a priori le nombre de valeurs du tableau on utilisera une boucle «while » (tant que) qui permettra de parcourir le tableau « tant que » une condition est vraie. Pour rentrer dans la boucle la condition doit être vraie. Il est nécessaire de faire évoluer la condition à l'intérieur de la boucle sous peine d'avoir une boucle infinie, qu ireste vraie tout le temps.

```
i=0 ;
While (condition==true) {
      Valeur=tab[i++] ;
      Evolution_de_la_condition;
}
```

En Labview on peut utiliser plusieurs méthodes pour accéder aux divers éléments d'un tableau.

Première méthode : Utilisation d'une boucle For « For loop » avec **Disable Indexing**

- Choisir une boucle « For » dans le menu « Functions/Structure/For Loop », la glisser sur le « Diagramme ». Le tableau reste à l'extérieur de la boucle

- Choisir une fonction Array size, et en relier l'entrée au tableau, et la sortie à la case «N » de la boucle For. Cette fonction « Array size » va retourner le nombre d'éléments du tableau, qui va nous servir de butée pour faire une boucle « For »

- Choisir une fonction « Index Array » et la mettre dans la boucle « For ».
  Cette fonction (regarder l'aide) demande 2 arguments d'entrée et retourne un argument de sortie. Le premier argument d'entrée est le tableau et le second est un index «i » de la boucle « For » qui va permettre d'accéder aux divers éléments, un à un. L'argument de sortie est l'élément «i » du tableau
  On est donc en train de programmer les boucles For classiques de Pascal ou « C »
  For i :=I to N do
  For (i=0 ; i<N ; i++)

- Relier la sortie du tableau à l'entrée de la fonction « Index array », on voir que le lien devient orange et au niveau de l'entrée du flux dans la boucle, un carré noir se dessine. C'est un «tunnel » qui va permettre d'agir sur la façon de passer les valeurs du tableau dans la boucle. En cliquant avec le bouton droit de la souris sur ce « tunnel » on peut voir « Enable Indexing » ou (dans le cas contraire) « Disable Indexing ».

- Les propriétés ***« Enable Indexing*** » et « ***Disable Indexing*** » sont importantes à connaître.
  - En choisissant « ***Enable Indexing »*** on indique qu'on « active l'indexation » automatique. Cela signifie que Labview va mettre en place l'indexation du tableau automatiquement lorsque les données franchissent la boucle : chaque élément du tableau va automatiquement passer un après l'autre dans la boucle « For ». Ce sont les valeurs Tab[i] qui rentrent dans la boucle une après l'autre.

  - En choisissant « ***Disable Indexing*** », on indique que l'on désactive l'indexation et dans ce cas, le tableau va rentrer en totalité dans la boucle. Ce sera au programmeur d'utiliser explicitement une fonction pour accéder à chaque élément du tableau. La fonctino pour accéder aux éléments d'un tableau est «index array »

- relier la variable « i » de la boucle « For », à la seconde entrée de la fonction « Index Array»
- Créer une variable indicatrice en sortie de la fonction « Index Array ». Celle ci va contenir successivement chaque élément extrait du tableau de rang « i ».

Deuxième méthode : Utilisation d'une boucle For « For loop » avec **Enable Indexing**

Dans ce cas la méthode est encore plus simple, puisqu'on profite de la propriété intrinsèque de faire passer les éléments du tableau un à un dans la boucle For. Il suffit de cliquer sur le tunnel au bord de la boucle « For » et de sélectionner « Enable Indexing ». On demande dans ce cas à ce que chaque élément du vecteur passe un à un dans la boucle. On n'a donc PLUS BESOIN de la fonction Index Array pour extraire un à un les éléments du tableau avec l'index « i » de la boucle. (dans ce cas l'index « i » de la boucle ne sert plus à rien).

Initialisation d'un tableau

Il convient par la suite de le remplir de valeurs. Pour cela, à l'aide de la flèche dans le menu «Tools » on sélectionne le tableau et on le dimensionne en tirant sur un de ses bords. Laissons apparaître 5 cases par exemple?
On peut initialiser ces 5 valeurs du tableau avec l'outil « A » (i.e saisie de chiffres au clavier) du menu « Tools ». Cliquer dans chaque case et taper une valeur. Bien sur le tableau, en dépit de ces 5 valeurs apparentes, peut contenir un nombre indéfini de valeurs. Il suffit de rentrer les valeurs séquentiellement dans chacune des cases suivantes.. Bien sur l'initialisation d'un tableau à partir du clavier est longue et fastidieuse?.Nous nous en servirons occasionnellement pour les TP, mais il est clair que le plus souvent, en situation de travail, les valeurs sont acquises à partir de fichiers ou bien) partir d'instruments d'acquisition!

La valeur située en regard du tableau, en haut à droite représente l'index de la première case apparente du tableau. (ex : si cette valeur est à 5 et le tableau contient 3 cases remplies apparentes, alors le tableau contient en totalité 8 valeurs : 5 cachées et 3 apparentes).

- On peut également initialiser un vecteur avec des valeurs aléatoires. Pour cela utiliser la fonction de génération de nombres aléatoires dans la palette de fonctions/Numerics/random Number (icône sous forme de dé)

Somme des éléments d'un vecteur : shift register

On veut additionner chaque élément d'un vecteur. Cela correspond à l'expression Pascal

For i :=1 to N do
Somme :=Somme+Tableau[i]

En Labview, il sera nécessaire d'accéder à chaque élément du tableau selon une des 2 méthodes exposées ci dessus, puis il va falloir additionner chacun des éléments à une variable qui doit contenir la somme de tous les éléments.
On appelle cette variable, une variable réentrante (i.e le genre de variable dont on se sert dans les langages textuels traditionnels pour affecter à une variable le résultat d'une opératin avec elle même: *somme=somme+x*). On crée cette variable dans la boucle « For » en cliquant sur le bord de la boucle avec le bouton droit de la souris, et en choisissant **« Add Shift Register** ».
Un petit carré orange se créé sur le bord de la boucle symbole de la variable réentrante «***shift register ».***

Cette variable « shift register » va servir à calculer la somme partielle des élements d'un tableau dans une boucle

- Créer une variable shift register sur le bord gauche de la boucle For (clic sur le bord de la boucle avec le bouton droit de la souris)
- Choisir une fonction « Somme » dans la fenêtre des fonctions, et relier sa première entrée à la sortie de la fonction Index Array vu ci dessus, et sa deuxième entrée à la variable réentrante « shift register »
- Relier la sortie de la fonction Somme à la seconde variable shift register. Celle ci contiendra en fin de parcours de tableau, la somme de tous les éléments du tableau.

Les fonctions portant sur les tableaux

- Array size : permet de connaître le nombre d'éléments d'un tableau
- Index Array : permet d'accéder séquentiellement à toutes les valeurs du tableau. Cette fonction (regarder l'aide) demande 2 arguments d'entrée et retourne un argument de sortie. Le premier argument d'entrée est le tableau et le second est un index « i » de la boucle « For » qui va permettre d'accéder aux divers éléments, un à un. L'argument de sortie est l'élément « i » du tableau

  On est donc en train de programmer les boucles For classiques de Pascal ou « C »

  For i :=1 to N do
  For (i=0 ; i<N ; i++)

Exemple TP5 : remplir un vecteur de nombres aléatoires

Créez et initialisez au clavier un tableau de valeur réelles
Sélectionnez la fonction de valeur aléatoires
Générer 5 valeurs avec une boucle For
Reliez la fonction Random avec le tableau ! comparez en choisissant « Enable Indexing » et « Disable Indexing »

Exemple TP6 : Moyenne d'un tableau de valeurs numériques

Créez et initialisez un tableau de valeurs entières
Accédez aux éléments du tableau, par un boucle For par la méthode que vous voulez (Enable Indexing ou Disable Indexing)
Calculez la somme et la moyenne de ce tableau
Créer un sous VI calculant la moyenne d'un tableau

Exemple TP7 : Calculez les statistiques de base d'un tableau de valeurs numériques

Créez et initialisez un tableau de valeurs entières
Calculez la somme et la moyenne de ce tableau
Calculez la somme des X au carré, la somme des xi au carré
Créer un sous VI calculant les statistiques ci dessus

Les structures de contrôle : boucles et tests

On retrouve dans Labview des structures de contrôles connues dans d'autres langages, tests et boucles sont représentés sous formes graphiques. Les boucles sont représentées sous forme d'une aire fermée. Tous les objets graphiques qui sont inscrits dans l'aire appartiennent à la boucle et sont donc exécutés itérativmeent!

La boucle For : « For Loop » :

elle équivaut à For **i** :=1 to **N** do

Dans Labview la boucle ***For*** est représentée sous forme d'une aire sur laquelle on va retrouver les 2 paramètres essentiels d'une boucle For : la valeur maximale **« N »** et l'indice de boucle **« i ».** Bien entendu pour faire fonctionner la boucle il faudra coller (initialiser) ces valeurs «i » et « N » avec des variables ou constantes

Nota : en choisissant la méthode « Enable indexing » pour accéder aux élements d'un tableau, il n'est pas nécessaire d'initialiser « N » et « i » de la boucle. Ces valeurs sont renseignées automatiquement par la méthode « Enable indexing ». En revanche pour la méthode Disable Indexing, il serait nécessaire d'initialiser « N » et « i »

La boucle While « While Loop »

Rappel : Les instructions contenues dans le corps de la boucle s'exécutent SI l'expression est VRAIE et TANT QUE l'expression booléenne est « vraie »

While (**<*expression*>**) do
Begin
Instruction 1 ; instruction n ;
end ;
	While (<expression>) {
	instruction 1 ; instruction n ;
	}

• En Pascal	• En C

Dans Labview la boucle ***While***, déposée sur le « Diagramme » est représentée sous forme d'une aire qui va englober un certain nombre d'objets graphiques (les instructions), et sur laquelle :

- on va retrouver en bas à droite une petite icône représentant l'expression booléenne de la boucle. Il est nécessaire d'initialiser cette expression booléenne à quelque chose de VRAI ou FAUX! au moyen d'une variable booléenne que l'on va relier
- on retrouve également en bas à gauche de l'aire de la boucle une variable préexistante «i » dont on peut se servir comme d'un compteur d'itération. Cette variable si on s'en sert, est auto-incrémentée par labview.

Pour créer cette variable booléenne qui va contrôler l'exécution de la boucle, il suffit de cliquer avec le bouton droit de la souris sur cette icône de boucle et de choisir « create control ». Cela crée une variable booléenne qui par défaut est initialisée à FAUX/OFF. ***Donc par défaut la boucle ne s'exécute PAS*** !

Sur le control Panel cette variable booléenne apparaît comme un bouton «ON/OFF » qui est à OFF (donc faux) par défaut. En cliquant sur ce bouton booléen, on le fait passer à « ON donc VRAI »?. Donc l'expression prend la valeur VRAI et les instructions contenues dans la boucle s'exécutent TANT QU'on n'a pas cliqué une nouvelle fois sur le bouton pour le faire passer à OFF/FAUX.

Exemple ?? TP : faire une boucle qui affiche la date en heure :minute :seconde, toutes les secondes jusqu'à ce qu'on clique sur un bouton et pas plus de 10secondes

- Choisir une fonction d'affichage de la date «fonction « get date/time string »
- Créer les variables adéquates
- Choisir une fonction de temporisation : fonction « wait »..l'initialiser avec une valeur de 1000 millisecondes
- Placer ces 2 fonctions dans le corps de la boucle
- Créer l'expression booléenne de fin de boucle
  -On veut que la boucle s'exécute (expression VRAIE) si le bouton booléen est sur «ON » ET le compteur de boucle est inférieur à 10

Les structures de contrôle de type « IF »

Dans labview, les tes de comparaison de type « if » sont regroupés dans la palette de fonctions dans le menu « comparisons ». On trouvera là, toutes sortes de tests de comparaisons sous forme de portes graphiques (= > < et ou etc?) à 2 entrées, auxquelles on reliera les 2 variables à comparer.

Les structures de contrôle de type « Case »

On peut sélectionner un élément parmi une liste au moyen d'une structure de contrôle de type «Case ».

- Utilisation de la structure Case :

La structure Case est comme en Pascal ou en C une structure de contrôle de programmation permettant d'effectuer UN choix parmi « n ». Il est donc nécessaire pour faire fonctionner la structure « Case » d'avoir

- une variable instanciée avec une liste de « n » choix
- une variable permettant de sélectionner un choix parmi les « n » proposés

rappel en langage textuel (en Pascal par exemple) d'une structure « case »

Syntaxe:

```
CASE <variable scalaire> OF
<choix1>          :          BEGIN
<instructions_liees_au_choix_1> END;
.....
<choix      2>    :          BEGIN
<instructions_liees_au_choix_2> END;
<choix      n>    :          BEGIN
<instructions_liees_au_choix_n> END;
ELSE
    <instructions_liees_aux_choix_differents_de
_1,2et_n>
END;
```

exemple;

```
TYPE choixmenu=1..3;    {variable scalaire de type
intervalle}

VAR bon_chiffre:choixmenu

bon_chiffre:=[1..3];
REPEAT
  BEGIN
        writeln(' Votre choix ? '); readln(choix);
        CASE choix OF
              1: sauvegarder;
              2: Imprimer;
              3: exit;
        ELSE  writeln('mauvais choix!);
  END;
UNTIL choix IN bonne  chiffre;
```

Dans Labview l'objet graphique de la structure « Case » présente un point d'interrogation sur le bord gauche. C'est l'entrée de la structure « Case ». Il faudra relier sur ce point d'interrogation une variable proposant plusieurs choix! Par défaut la structure case attend une variable de type booléen (2 cas vrai/faux). Mais, si on relie à cette entrée une variable d'un autre type, par exemple de type « Menu/list & ring », le choix portera alors sur un choix parmi les « n » du menu.

Après avoir relié une variable proposant « n » choix à la structure Case, il va falloir indiquer pour les choix « i » de 1 à « n », quelle sera la variable retournée pour chaque choix effectué !

Il faudra donc pour chaque choix de la liste, créer une constante DANS la structure. Cette constante sera la valeur renvoyée par le « case » pour CHACUNE de ses branches/choix ! Pour chaque cas du « case » on va donc initialiser la constante à la valeur qu'elle devra renvoyer à l'extérieur de la structure.
En d'autre terme la valeur de la constante à l'intérieur de la structure « case » est la valeur retournée pour un certain cas « i ».

Il faut donc créer dans l'aire de la structure autant de constantes que ce qu' il y a de choix possibles!

La majeure partie de ces actions se font en cliquant avec le bouton droit de la souris!

- Cliquer sur la barre de nombre de choix en haut de la structure case avec le bouton droit de la souris
- Choisissez « Add case after ». Cela permet de rajouter un cas dans la structure
- Pour tous les choix de la structure, créer une constante de type chaîne de caractères (par exemple) et saisir dans cette constante la chaine qui sera retournée par la structure.
- Tapez « A » par exemple dans la constante, pour le choix 1
- Puis « B » dans la constante correspondant au choix 2

Etc? pour chaque choix

Exemple : ci dessous sur les chaînes de caractères

Les chaînes de caractères

La manipulation des chaînes de caractères est très importante lorsqu'on veut faire des échanges entre le PC et un appareillage via une liaison quelconque RS232 ou GPIB. La plupart du temps ces échanges, au travers de la liaison reliant les 2 appareillages, se font par des chaînes de caractères ASCII.
Il est donc nécessaire de savoir utiliser les fonctions portant sur les chaînes de caractères.

On trouvera les variables de type chaîne de caractères dans le menu Control « strings & Table ».
On trouvera les fonctions portant sur les chaînes de caractères dans le menu « Functions » « Strings »
(NdlR : je me demande s'il fallait dire de telles évidences ! ? »

Premier exercice simple d'extraction d'une sous chaîne :

- Créer une variable de contrôle de type chaîne sur le « front panel » et inscrivez la chaîne de caractères « CANAL A 15.6°C » tapée au clavier, dans cette variable de type chaîne
- Il s'agit dans cet exercice d'extraire la sous chaîne « 15.6 » et de la convertir en valeur numérique réelle.
- Calculer la longueur de la chaîne (fonction « length ») et utiliser la fonction « substring » afin de découper la sous chaîne correspondant à la valeur numérique
- Convertir cette chaine « 15.6 » en valeur réelle avec la fonction « From Exponential/Fract/eng » ou bien la fonction « Format & Strip » dans la rubrique des fonctions portant sur les chaînes de caractères

La fonction String Subset nécessite comme arguments d'entrée : une chaîne de caractère et 2 constantes « length » et offset. La fonction va extraire la sous chaîne partant de la valeur contenue dans «offset » jusqu'à la fin de la chaîne. Ce premier exercice nécessite 2 constantes, que l'on va initialiser par une saisie au clavier. Ce qui n'est pas très rigoureux, si le format des chaînes de caractère est changé d'un programme à l'autre.

- Exemple : TP :
- Modifier l'exercice en utilisant la fonction « Match pattern » pour extraire dynamiquement des sous chaînes situées entre 2 motifs? On veut dans notre cas extraire la valeur située ENTRE le motif « A « et « °C ».

Envoi d'une chaîne du PC vers un instrument

On va dans cette exercice composer une chaîne de caractères à partir de différentes sous chaînes que l'on va concaténer. Ce cas se produit lorsqu'on veut qu'un PC envoie certaines commandes sur une ligne série vers un dispositif actionneur qui permet de définir une température. Ce dispositif possédant 4 voies A,B,C,D le problème qui se pose est alors de construire une chaîne « CANAL A 12.5°C » par exemple? avec l'identificateur du canal A,B,C et la température qui sera donnée dans une variable du programme.

- Créer une variable de type « Menu Ring » que l'on trouvera dans le menu « Control », dans la rubrique « List &Ring ». Donner le label « Liste d'entrée » cette variable. Cette variable permet de saisir plusieurs chaîne de caractères dans un objet graphique de type listbox.
- Entrez dans cette variable de type liste 4 choix ayant les valeurs : « Voie A » « Voie B » « Voie C » « voie D »
- Créer sur le Diagramme une structure de contrôle de type « Case » qui va permettre selon le choix effectué, de retourner une certaine valeur convenue !
- Relier la variable « liste entrée » au bord gauche de la structure case « point d'interrogation »
- Pour chaque cas (4 cas en tout) dans la structure, créer une constante de type chaîne et instancier cette constante. On changera de cas de la structure en cliquant sur le sommet de la structure
- Donner la valeur « A » pour le cas 1 (correspondant à l'entrée « Voie A »)

- Donner la valeur « B » pour le cas 1 (correspondant à l'entrée « Voie B »)
- Donner la valeur « C » pour le cas 1 (correspondant à l'entrée « Voie C »)
- Donner la valeur « D » pour le cas 1 (correspondant à l'entrée « Voie D »)
- Créer une variable indicateur de type chaîne hors de la structure Case et pour chaque cas relier la constante de l'intérieur de la structure, à la variable indicatrice à l'extérieur.

Créer une variable de type digital control (température) : soit une simple variable numérique soit une variable de type knob (aspect d'un bouton qu'on tourne).
Par défaut ce bouton de type Knob fournit une graduation entre 1 et 10. On peut modifier l'échelle et la graduation en cliquant sur le bouton droit et en choisissant « Text label ». Un label apparaît en dessous du bouton et on peut alors modifier l'échelle en saisissant les valeurs que l'on veut dans la zone de saisie et en terminant par «enter ». On peut rajouter autant de graduation que l'on veut par l'option « Add item after ».

Ce bouton renvoie une valeur numérique réelle, la transformer en chaîne de caractères par la fonction «To Fractionnal » que l'on trouvera dans le menu Fonctions/Strings/Conversion » Cette fonction permet de convertir un nombre réel ou entier en une chaîne de caractères, selon une précision que l'on donne.
Enfin utiliser la fonction de concaténation de chaînes pour concaténer :

- La chaîne constante « canal »
- La chaîne issue du choix de la structure case
- La chaîne correspondant à la valeur numérique choisie
- La chaîne constante °C

Afin d'aboutir à une chaîne comme Canal C 15.2 °C

Réception d'une chaîne d'un instrument sur le PC

Prenons le cas inverse de celui du paragraphe précédent. Un appareillage déporté '=(instrument de mesure) communiquant avec un PC par un liaison série sur un port RS232 envoie aussi ces données au PC sous forme de chaînes de caractères. Ces chaînes devront être traitées par le PC. On est alors confronté au problème inverse, il s'agit de repérer dans la chaîne, l'information qui nous intéresse (une valeur) et de la transformer en valeur numérique entière ou réelle.

> Exercice : Traiter la chaîne de caractères « Canal 1 temp. 17.5°C voltage 3.5volts », analyser la chaîne et extraire les 3 valeurs de cette chaîne dans 3 variables canal, temperature et voltage.

Quelques fonctions portant sur les chaînes de caractères

- Length : retourne la longueur d'une chaîne
- Concatenate : permet de concaténer plusieurs chaines de caractères dans une seule
- Subset : permet d'extraire une sous_chaîne de caractères, et l'isoler dans une autre plus courte
- Match pattern : permet de réperer un motif dasn une chaîne. La fonction retourne la sous chaîne située avant le motif,celle située après, ainsi que le motif s'il a été trouvé.

Les Graphiques

Un des avantages majeurs de Labview est de fournir des fonctions graphiques de haut niveau immédiatement utilisables par le programmeur. On trouvera ces fonctions graphiques dans le menu «Control » « Graphs »

Ces «objets graphiques permettent de représenter sans aucune difficulté un flot de données. Il suffit de connecter un vecteur de données à cet objet graphique pour visualiser immédiatement ces données comme sur un écran de contrôle.

Quels sont les principaux objets graphiques disponibles :

1. Waveform chart : le « chart » a un vecteur (tableau 1D) de données en entrée, qui représentent les valeurs successives des ordonnées « y » de la courbe. Dans ce cas on ne maîtrise pas l'abscisse « x » de la courbe tracée. Elle est prédéterminée par le numéro d'ordre des valeurs du vecteur. C'est donc un affichage chronologique
2. Waveform Graph
3. XY Graph : permet de tracer des fonction y=f(x). A l'inverse du « waveform chart », cet objet graphique permet d'afficher et contrôler des couples de valeur x,y

Exemple ? ? :TP :
Nous allons visualiser un flot de données de température issu d'un thermomètre.

- Utilisons un sous VI déjà programmé dans le répertoire Labview/Vi.lib/Tutorial.llb/Digital Thermometer.vi. Ce vi permet de simuler des saisies de températures.
- *Visualiser une valeur de température :*
  - Il suffit de relier ce vi à un objet graphique de type WaveForm Chart, pour visualiser immédiatement les données issues du thermomètre.
- *Acquérir les valeurs en continue :*
  - Incluez ces 2 objets dans une boucle de type While, dont le test d'arrêt sera effectué sous le contrôle d'un indicateur booléen « stop » que vous prendrez dans le menu « Control/Boolean » Par défaut ce bouton initialisé à STOP renvoie la valeur « faux ». Pour faire démarrer la boucle immédiatement, il faut donc envoyer la valeur « VRAI » à la boucle While. Donc il faut inverser la valeur de la variable booléenne. Inversez là donc avec une fonction « NOT » dans le menu « Fonctions/Boolean »
- *Ralentir le défilement des valeurs :*
  - Le défilement des valeurs, lié à leur rythme d'acquisition, va trop vite ? Intégrez alors une fonction de temporisation « Milliseconds to Wait » dans le menu « Function/Time Dialog »

Exemple ? ? :TP :
Nous allons générer des valeurs aléatoires, remplir un vecteur et afficher les valeurs.

- Avec une boucle For générons « n » valeurs aléatoires avec la fonction « Random »
- Relier cette fonction à un vecteur à travers la boucle : mode de transmission Enable Indexing ou Disable Indexing ? ? ?
- Relier le vecteur à un objet graphique de type WaveForm Graph
- Mettre une temporisation « Wait until next ms multiple »
- Contrôlez l'affichage des valeurs avec une boucle While que l'on arretera par un bouton ON/OFF

Utilisation d'un graphe XY : cluster & bundle

Le graphe XY s'obtient par le menu « Controls » / XY Graph. Cette fonction demande comment entrer un flux de type cluster, dans lequel les éléments des vecteurs d'entrée sont associés un à un selon leur rang :
{ {Xi,Yi}, {Xi+1,Y,i+1}, ?.. {Xi+n, Yi+n} }

Si nos valeurs sont rangées dans des vecteurs 1D, il nous faut donc transformer ces vecteurs en un cluster avec la fonction *« Index and Bundle cluster Array »*. La fonction « Index and Bundle cluster Array » crée un tableau de cluster dans lequel chaque élément est un groupage des éléments correspondants des tableaux d'entrée. Par exemple si le tableau Tab-X contient les éléments {1,2,3} et le tableau Tab-Y contient {4,5,6}, la fonction *« Index and Bundle cluster Array »* fournit le résultat { {1,4}, {2,5}, {3,6} }

Cette fonction *« Index and Bundle cluster Array »* s'obtient en cliquant par le bouton droit de la souris sur un tableau et en choisissant le menu « Array Tools ». Il suffit alors de relier les vecteurs d'entrée à cette fonction. Le flux de sortie qui est produit est directement compatible avec un affichage par l'objet graphique XY Graph.

- Exemple ? ? : TP :
- Générer 10 valeurs aléatoires avec la fonction Random

- Les placer dans un vecteur 1D de variables indicatrices Tab-X
- Pour chaque valeur xi calculer le terme y= 2.5*xi + i, ranger ces valeurs calculées dans un second vecteur Tab-Y
- On a donc 2 vecteurs d'entrée Tab-X et Tab-Y que l'on veut représenter graphiquement sous la forme Y=f(X)
- Trier le vecteur Tab-X pour avoir une distribution croissante
- Lorsqu'on à 2 vecteurs à afficher au moyen d'un graphe XY, il est nécessaire d'utiliser une fonction qui transforme les tableaux en cluster. Relier ces 2 vecteurs à la fonction *« Index and Bundle cluster Array »*
- Relier la sortie de cette fonction à l'objet XY Graph
- Ergonomie :
  - Placer ce programme dans une boucle While dont on contrôlera l'exécution avec un bouton poussoir Booléen « On/OFF »
  - Mettez une temporisation dans la boucle pour ralentir l'affichage des valeurs

Manipulation des fichiers avec Labview

Rappel sur la nature des fichiers

Les fichiers sont les structures capitales d'un système informatique dans la mesure où ils permettent une rémanence des informations sur des supports magnétiques, au delà de la durée d'exécution d'un programme et de l'alimentation de la machine.

Les fichiers servent donc à conserver des données.

Il est donc nécessaire et indispensable dans n'importe lequel des langages informatiques, de savoir utiliser les fichiers. Les opérations de base que l'on peut et doit faire sur des fichiers sont

- Ouverture
- Lecture ou écriture (i.e parcours et accès aux éléments)
- Fermeture

Rappelons

- qu'une lecture de données A PARTIR d'un fichier équivaut à un accès à la donnée sur un support magnétique (mémoire de masse) et à son transfert dans la mémoire centrale (RAM) de l'ordinateur

  données sur disque ➔lecture➔ variable en mémoire centrale

- qu'une écriture de données DANS un fichier correspond à un transfert de la donnée DEPUIS la mémoire centrale, et à son écriture sur le support magnétique

  données dans une variable en RAM ➔écriture ➔donnée sur disque

Les fichiers peuvent être de 2 types : « ***texte*** » ou « ***binaire*** »

- Les fichiers de type texte contiennent des éléments d'information codés sur un octet c'est à dire un caractère et dont la valeur fait partie de la table ASCII des caractères. Les fichiers de texte sont donc composés d'éléments de notre alphabet courant (lettres minuscules a-z, lettres majuscules A-Z, chiffres 0-9, et tous les caractères visibles du clavier, ponctuation &é»'(-è_çà)#{[|\^@]} etc? Ce sont des fichiers lisibles par un opérateur humain puisque la machine interprète les octets contenus comme appartenant à la table ascii des caractères

Exemple
Le chiffre « 1 » contenu dans un fichier de type texte serait codé sous la forme du caractère «1 ». Ce caractère est codé sur UN SEUL octet. Le caractère « 1 » dans la table ASCII des caractères est codé avec la valeur numérique « 33 » (voir ce chiffre car j'ai mis n'importe quoi ici) c'est à dire en binaire 00100001 (2^6 + 1)

- Les fichiers de type binaire contiennent des informations. Aucune interprétation de ces octets n'est faite par l'ordinateur. Ce sont des fichiers qu ne sont pas lisibles pas un opérateur humain, puisque les valeurs binaires qui seraient affichées ne correspondraient à aucun caractères de l'alphabet

Exemple
Le chiffre 1 contenu dans un fichier de type binaire serait codé sous la forme d'un nombre entier c'est à dire avec 2 ou 4 octets. Le chiffre 1 numérique est codé, sous la forme d'un entier sur 2 octets 00000000 00000001

Les fonctions portant sur les fichier dans Labview

Les fonctions portant sur les fichiers se trouvent dans le menu Functions /FILE IO

On trouve :

- *Read from spreadshit file*
- *Write to spreadshit file*

Exemple ? ? TP :
Le but est de lire un fichier comportant 2 colonnes de valeurs X,Y séparées par une tabulation

On utilisera la fonction *Read from spreadshit file*
Les arguments principaux de cette fonction sont :

1. Un booléen vrai/faux indiquant si les vecteurs lus dans le fichier doivent être transposés ou pas
2. Le file path : c'est le chemin d'accès au fichier de texte que l'on veut lire
3. Le nombre de lignes à lire : -1 pour lire toutes les lignes jusqu'à la fin de fichier
4. Un format de lecture des valeurs
5. All rows : cet argument doit être un tableau multi-dimensionné qui va recevoir toute les valeurs lues? appelons le « tabval »
6. First row : un vecteur mono dimensionné correspondant à la première ligne lue

On positionnera le booléen de la fonction « *Read from spreadshit file»* à « vrai » pour indiquer que l'on veut une transposition des colonnes du fichier en lignes?i.e les colonnes du fichier vont se retrouver en lignes dans le tableau « tabval »

On veut afficher y en fonction de x y=f(x)? la 1ère colonne du fichier est sensée représenter les valeurs X, l deuxième colonne les Y

Il faut donc extraire du tableau « tabval » chaque ligne séparément? on effectuera cette opération avec la fonction « index array» permettant d'extraire une ligne d'index « i » du tableau tabval? on placera la première ligne extraite dans un second tableau « tabx », et la seconde dans un 2ème tableau « taby »

On a vu dans le chapitre consacré aux graphiques que pour afficher des valeurs dans un «Wave Form Chart » ou un « XY Chart », il fallait que les valeurs des 2 vecteurs a afficher soient groupées dans un «bundle » de manière à associer les couples { {xi,yi}?{xn,yn} }

Il est alors nécessaire d'utiliser une fonction «bundle » que l'on trouvera dans le menu « Array tools » en cliquant sur un tableau avec le bouton droit de la souris?
On relie les 2 vecteurs TabX et TabY aux entrées de la fonction bundle, et nos valeurs seront prêtes à être affichées dans un graphiques « XY Chart »