Prosit 4

# Analyse du contexte

L’équipe MMH travaille sur un programme de gestion des données mais se rend compte qu’il n’est pas assez optimisé.

# Mots clés

* **Moyenne glissante :** permet de lisser une série de valeurs exprimées en fonction du temps (série chronologique). Elle permet d'éliminer les fluctuations les moins significatives.
* **Capteurs :** Organe qui élabore, à partir d'une grandeur physique, une autre grandeur physique, souvent de nature électrique, utilisable à des fins de mesure ou de commande.
* **Pointeurs :** une variable contenant une adresse mémoire
* **Langage C :** langage de programmation qui s’écrit dans un fichier source. Ensuite ce fichier doit être traduit à l’aide d’un compilateur en langage machine.
* **Complexité d’un algorithme :** une information sur son temps d’exécution liée au volume n de données à traiter.
* **Structure :** regroupement de plusieurs objets, de types différents ou non. Grosso modo, une structure est finalement une boîte qui regroupe plein de données différentes.
* **Tableau :** structure de données représentant une séquence finie d'éléments auxquels on peut accéder efficacement par leur position, ou indice, dans la séquence.
* **Bits :** Unité élémentaire d'information pouvant prendre deux valeurs distinctes, notées 0 et 1
* **Type de variable :** définie les données qui peuvent être traitées directement

# Problématique

Comment améliorer le temps d’exécution du programme et optimiser l’espace mémoire ?

# Contraintes

* Assigner une constante à chaque capteur
* Reprendre l’algorithme donné
* Espace mémoire à diminuer
* Remplacer les tableaux par des structures

# Livrables

* Fichier texte avec l’algorithme amélioré

# Généralisation

* Savoir utiliser des pointeurs
* Savoir calculer la complexité d’un algorithme
* Apprendre à manipuler les structures
* En bref : Apprendre à manipuler les données

# Pistes de solutions

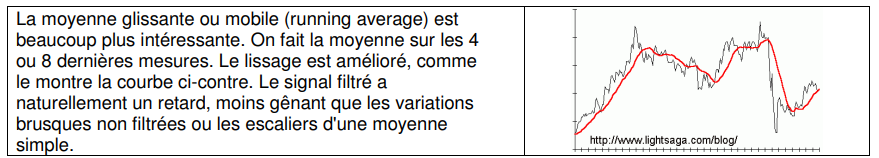
* Le temps d’exécution dépend de la machine et de l’algorithme sur laquelle on le fait tourner et la complexité dépend uniquement de l’algorithme
* Changer les types de variable pour gagner de l’espace en mémoire (ex : utiliser « float » et « % »)
* Utiliser les piles et les files
* Réduire la taille des tableaux
* Réduire le nombre d’itérations
* Factoriser le code

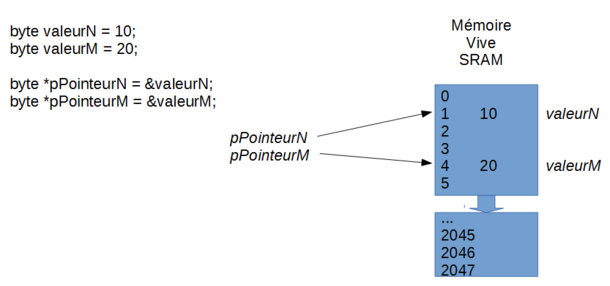
# Plan d’action

* Calculer la complexité de l’algorithme donné
* Optimiser le programme en changeant le type des variables et les tableaux en structures
* Réduire intervalle
* Factoriser le code
* Recalculer la complexité du programme après optimisation

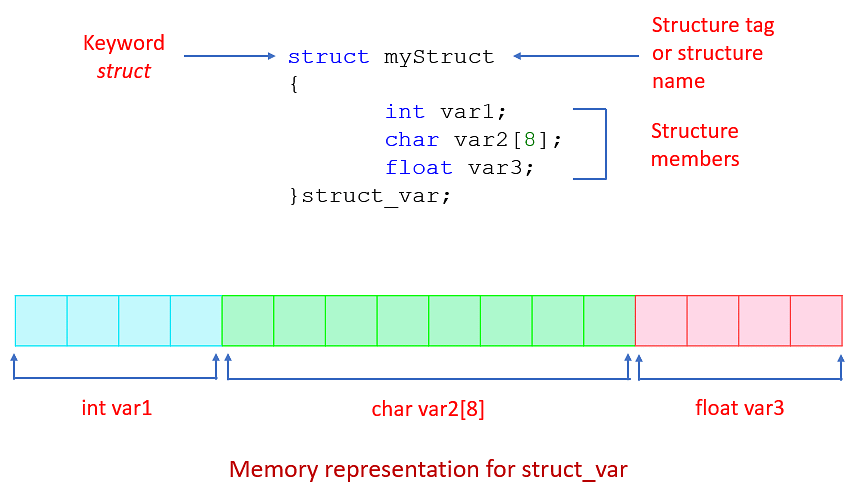
# Cours

**Moyenne glissante ou mobile :**



**Pointeurs :**

**Structures :**



# Réalisation du plan d’action

Je vais tout d’abord remplacer la moyenne instantanée par une moyenne glissante. Puis il faut changer les types de variable.

THS – Algorithme WWW

Variables globales

uint\_16 MoyA… : Pour chaque capteurs auquel on souhaite appliqué une moyenne

Librairie :

#include "Inter.h

#include "MoyGlis.h"

#include "OledPix.h"

En configuration (setup) :

SetupMoyGlis();

SetupInter();

Nb\_Capteurs : le nombre total de capteurs et outils de mesures

Indice\_max\_Tableau : représente le nombre de mesure pour la moyenne instantanée

Les tableaux seront dimensionnés par rapport à ces différentes valeurs :

Tab\_Mesures[Nb\_Capteurs - 1]

Tab\_Erreur[Nb\_Capteurs – 1]

Tab\_Moy\_Instant[NbCapteurs – 1]

Et pour chaque capteur :

Tab\_Capteur\_num\_capteur[indice\_max\_Tableau]

En exécution (loop) :

Le programme appelera successivement les fonctions :

Lecture\_Capteurs()

Maj\_Mesures(…,…)

GetMoy16(analogRead(A…)) pour chaque capteur

Les fonctions/procédures principales :

PROCEDURE LECTURE\_CAPTEURS

Lecture\_Capteurs()

Tab\_Erreur = 0

Pour chaque capteur (num\_capteur) :

Si Mesure\_capteur <> « erreur » alors Tab\_Mesures[num\_capteur] = Mesure\_capteur Sinon Tab\_erreur[num\_capteur] = 1

--------------------------------------------------

PROCEDURE MAJ\_MESURES

Maj\_Mesures(Tableau[],Erreur[])

Une instruction par tableau de capteur :

Si Erreur[num\_capteur] <> 1 alors

Tableau\_Capteur\_num\_capteur[] = Decalage(Tableau\_Capteur\_num\_capteur[], Tableau[num\_capteur])

--------------------------------------------------

Les 2 fonctions factorisées :

FONCTION DECALAGE

NouveauTableau[] = Decalage(Tableau[], Valeur)

Pour i allant de 0 à indice\_max\_Tableau - 1

Tableau[i] = Tableau [i+1]

Tableau[indice\_max\_Tableau] = Valeur

Return Tableau[]

--------------------------------------------------