Info 2 PC

Introduction

- L'informatique est un outil puissant (cf cours du S1)
- L'informatique est omniprésent dans le domaine scientifique
- Apprenez à le maîtriser pour vous faciliter la vie et réussir vos études et vos projets

Objectifs de l'UE

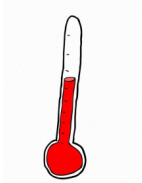
- Vous permettre de :
  - manipuler des données structurées
  - lire et stocker les données dans des fichiers
  - comprendre comment l'informatique peut servir en physique et en chimie
  - maîtriser des outils informatiques pour résoudre des problèmes et ou réaliser des expérimentations scientifiques

Nos outils

- Le langage algorithmique
- Le langage C++
- Les ordinateurs des salles de TP
- Une plate-forme Arduino + une carte mémoire

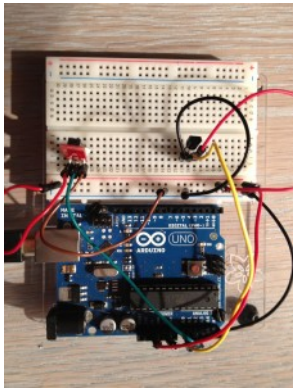
- Quelques composants électroniques

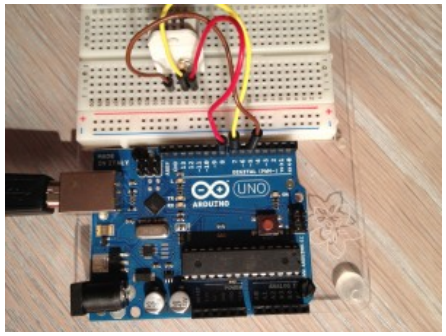
Le projet de l'UE

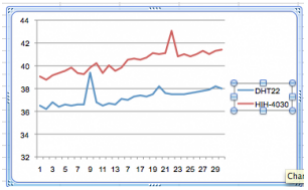


- Réaliser une plate-forme de mesures de la température qui stockera les données dans un fichier d'une carte mémoire
- Analyser les données collectées et stockées dans le fichier de la carte mémoire
- Suivant l'avancement, on envisagera des manipulations sur le fichier, pour en créer un nouveau intégrant les aspects temporels. L'objectif serait de pouvoir le traiter sous Excel/Calc pour en faire des représentations graphiques

En résumé on fera







Organisation de l'UE

- Un seul enseignant : moi :-)
- 6 séances de cours
  - Présentation des nouveaux concepts algorithmiques (enregistrements, fichiers, tri) et d'Arduino
- 9 séances de TD
  - Manipulation des nouveaux concepts sur des exercices
    - Début semaine 3
- 4 séances de TP
  - Programmation et manipulation d'Arduino

Les TPs commenceront en semaine 4

MCC :
- 1 écrit !

Pour vous motiver

- Quelques idées de ce qu'il est possible de réaliser avec Arduino :
  - http://hacknmod.com/hack/top-40-arduino-projects-of-the-web/
  - http://www.instructables.com/id/Arduino-Projects/
  - http://www.theinebriator.com/
  - http://blog.makezine.com/2012/05/09/wifi-rc-car-has-camera-and-force-feedback/
- Et le must : c'est abordable !
  - Les personnes désireuses de faire des projets autour du sujet peuvent venir me voir

... si on (re)venait à l'algorithmique ?

Structure d'un algorithme

```
Algorithme Nom_de_l_algorithme
{
    Bloc de déclarations
    Séquence d'instructions
}
```

Le bloc de « Déclarations » contient tous les objets utilisés par l'algorithme, ainsi que leur type (nature).

Tout objet utilisé doit avoir été déclaré.

De nouveaux types : les types composés (enregistrements)

- Fréquemment, le concepteur d'algorithmes a besoin de types plus adaptés aux données qu'il veut manipuler.
- Par exemple, s'il veut faire des manipulations sur des informations concernant la carte grise d'un véhicule, idéalement, il souhaiterait disposer d'un **type** approprié pour faciliter l'accès aux différentes parties la constituant. *Prenons une carte grise « simplifiée » : elle contient le nom et prénom du propriétaire du véhicule, la marque et le modèle du véhicule, son immatriculation, son année de mise en circulation, sa puissance fiscale.*

Rappels sur les types et les déclarations

- Types numériques : `entier` ou `réel`
- Types alphanumériques : `caractère` ou `chaîne de caractères`
- Type `logique` (booléen) : VRAI ou FAUX
- Déclaration :

```
variable Nom_var : type
```

exemple :

```
variable Nom_etudiant : chaîne de caractères
         An_naissance, Annee_inscription : entier
         Inscrit : logique
```

Les types composés (enregistrements)

- Malheureusement, un tel type n'existe pas de base dans le langage algorithmique (ni dans le langage C++). D'ailleurs on constate que les informations qu'il définit sont complexes car de natures différentes : certaines textuelles (nom, prénom, marque, modèle, immatriculation), certaines numériques (année de mise en service, puissance fiscale)
- Afin de permettre au développeur de manipuler ces données complexes, la langage algorithmique introduit la possibilité de définir des enregistrements.

Les types composés (enregistrements)

- Un enregistrement est une structure qui consiste en un nombre fixe de champs (éléments) de différents types.
- Le nom de la structure servira à nommer le nouveau type.
- Dans la structure, chaque champ est défini par :
  - un type (simple ou composé)
  - un identificateur

Les types composés (enregistrements)

- La syntaxe est :

```
structure nom_du_type_composé
{
    nom_du_champ1 : type1
    nom_du_champ2 : type2
    ...
    nom_du_champn : typen
}
```

- Exemple :

```
structure carte_grise
{
    nom : chaine de caractères
    prenom : chaine de caractères
    marque : chaine de caractères
    modele : chaine de caractères
    immatriculation : chaine de caractères
    annee_mise_en_service : entier
    puissance_fiscale : entier
}
```

Définition et utilisation d'enregistrements

- Une fois, la structure d'un enregistrement (un type composé) ayant été définie, il est possible de déclarer des variables de ce type et de les utiliser. L'accès aux champs se fait au travers de l'opérateur « . » comme nous l'illustrons ci-dessous.

```
Algorithme Exemple_utilisation_enregistrement
{
    variable CG_voit1 : carte_grise
    CG_voit1.nom ← "DUPONT"
    CG_voit1.prenom ← "Louis"
    CG_voit1.marque ← "Renault"
    CG_voit1.modele ← "CLIO 2"
    CG_voit1.annee_mise_en_service ← 1910
    CG_voit1.puissance_fiscale ← 4
}
```

`CG_voit1` est une variable dont le nom définit un enregistrement de type `carte_grise`

Définition et utilisation d'enregistrements

```
structure triangle
{
    cote1 : reel
    cote2 : reel
    cote3 : reel
}

Algorithme Testons_des_triangles
{
    variable t : triangle

    Afficher("Saisir les valeurs des 3 cotés du triangle")
    Saisir(t.cote1,t.cote2,t.cote3)
    Si ((t.cote1 = t.cote2) ET (t.cote2 = t.cote3)) alors
    {
        Afficher("Ce triangle est équilatéral.\n")
    }
    sinon
    {
        // faire ici les autres tests
    }
}
```

Définition et utilisation d'enregistrements

- On peut affecter un enregistrement à un autre de même type.

```
Algorithme Exemple1
{
    variable t1, t2 : triangle

    Afficher("Saisir les valeurs des 3 cotés du triangle")
    Saisir(t1.cote1,t1.cote2,t1.cote3)

    t2 ← t1
    Afficher("Les valeurs des cotés du premier triangle sont : ",
t1.cote1,t1.cote2,t1.cote3,"\n")
    Afficher("Les valeurs des cotés du second triangle sont : ",
t2.cote1,t2.cote2,t2.cote3,"\n")
    Afficher("Ces cotés sont identiques pour les deux triangles.\n")
}
```

Définition et utilisation d'enregistrements

- Il est également possible de procéder à la copie champ par champ

```
Algorithme Exemple2
{
    variable t1, t2 : triangle

    Afficher("Saisir les valeurs des 3 cotés du triangle")
    Saisir(t1.cote1,t1.cote2,t1.cote3)

    t2.cote1 ← t1.cote1
    t2.cote2 ← t1.cote2
    t2.cote3 ← t1.cote3
    Afficher("Les valeurs des cotés du premier triangle sont : ",
t1.cote1,t1.cote2,t1.cote3,"\n")
    Afficher("Les valeurs des cotés du second triangle sont : ",
t2.cote1,t2.cote2,t2.cote3,"\n")
    Afficher("Ces cotés sont identiques pour les deux triangles.\n")
}
```

Traduction en C++

- Définition d'un type composé :

```
struct nom_structure
{
    Définition des éléments...
};
```

- Accès à un élément de la structure :

```
variable.nom_élément
```

Initialisation en C++

- Exemples :

```
struct date
{
    int jour;
    int mois;
    int annee;
} d1 = {1,1,1996}; // d1 : variable de type structure

date d2 = {12,4,1996}; //d2 : variable de type structure

date d3;
d3.jour = 23;

cin >> d3.mois;
```

Et si on voyait comment utiliser ces nouveaux types pour des tableaux et avec des sous-programmes ?

Exemple de tableau d'enregistrements

- Écrire un algorithme qui demande à l'utilisateur de saisir les coordonnées planaires de points définissant 10 segments et affiche les coordonnées points formant le plus grand des segments ainsi que sa taille.

```
structure Point
{
  X,Y : réel
}
structure Segment
{
  P1, P2 : Point
}
```

Exemple de tableau d'enregistrements

```
fonction SaisiePoint(E i : entier) : Point
{
     variable P : Point
     Afficher("Saisissez les coordonnées du point ", i)
     Saisir(P.X,P.Y)
     retourne P
}
fonction SaisieSegment(E i : entier) : Segment
{
     variable S : Segment
     Afficher("Segment n°", i)
     S.P1 ← SaisiePoint(1)
     S.P2 ← SaisiePoint(2)
     retourne S
}
```

Exemple de tableau d'enregistrements

```
fonction longueurSegment(E S: Segment) : réel
{
     retourne √((S.P2.X-S.P1.X)²+(S.P2.Y-S.P1.Y)²)
}
```

Exemple de tableau d'enregistrements

```
Algorithme Exo_Sur_Segments
{   constante entier TAILLE ← 10
    variable v : tableau de TAILLE Segment
             i, iPlusLong : entier
             PlusLong : réel
    Afficher("Nous allons vous demander de saisir ", TAILLE, " segments.")
    Pour i de 0 à TAILLE-1
        v[i] ← SaisieSegment(i+1)
    iPlusLong ← 0
    PlusLong ← longueurSegment(v[iPlusLong])
    Pour i de 1 à TAILLE-1
        Si (PlusLong < longueurSegment(v[i])) alors
        {
            iPlusLong ← i
            PlusLong ← longueurSegment(v[iPlusLong])
        }
    Afficher("Le segment [(",v[iPlusLong].P1.X,",",v[iPlusLong].P1.Y,"),
(",v[iPlusLong].P2.X,",",v[iPlusLong].P2.Y,")] est de tous les segments saisis celui
ayant la plus grande longueur. Elle est de ", PlusLong,".\n")
}
```

Exemple de tableau d'enregistrements en C++

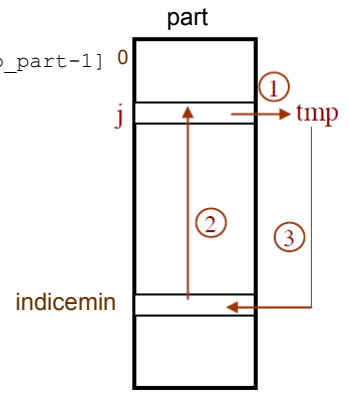
- Saisir le record et le N° des athlètes en utilisant un tableau d'enregistrements, puis le trier dans l'ordre croissant des records.
- Afficher le N°, le record et le rang de chaque athlète en donnant le même rang aux ex aequos.

Exemple de vecteur d'enregistrements en C++

```
#include <iostream>

using namespace std;

struct athlete
{
    int num;
    float rec;
};

int main()
{
    const int nb_part=15;
    athlete tmp,part[nb_part]; // part est un tableau du type athlete
    int i,j,indicemin,rang;
    float min;

    // saisie du tableau
    for (i = 0; i < nb_part; i++)
    {
        cout << "N° et record ?" << endl;
        cin >> part[i].num >> part[i].rec;
    }
```

Exemple de vecteur d'enregistrements en C++

```
// tri du vecteur (tri « rapide »)
for (j = 0; j < nb_part-1; j++)
{
    // on cherche le minimum dans la partie [j,nb_part-1]
    min = part[j].rec;
    indicemin = j;

    for (i = j+1; i < nb_part; i++)
        if (part[i].rec<min)
        {
            min = part[i].rec;
            indicemin = i;
        }

    // on échange si nécessaire
    if (indicemin != j)
    {
        tmp = part[j];
        part[j] = part[indicemin];
        part[indicemin] = tmp;
    }
}
```



Exemple de vecteur d'enregistrements en C++

```
    // affichage
    cout << "rang participant record" << endl;
    cout << "1 " << part[0].num << " " << part[0].rec << endl;

    rang = 1;
    for (i = 1; i < nb_part; i++)
    {
        if (part[i].rec>part[i-1].rec)
            rang = i+1; //pour les records égaux

        cout << rang << " " << part[i].num << " " << part[i].rec << endl;
    }

}
```