

Rappel

Structures

- Plus haut, nous avons vu les tableaux qui sont une sorte de regroupement de données de même type. Il serait aussi intéressant de regrouper des données de types différents dans une même entité.
- Nous allons donc créer un nouveaux type de données (plus complexes) à partir des types que nous connaissons déjà: *les structures*.
 - Une structure permet donc de rassembler sous un même nom, des informations de type différent. Elle peut contenir des données entières, flottantes, tableaux, caractères, pointeurs, structure, etc... . Ces données sont appelés les membres de la structure.
 - **Exemple**: la carte d'identité d'une personne: (nom, prenom, date_de_naissance, lieu_de_naissance, quartier, etc...).

Rappel

Structures: déclaration

Pour déclarer une structure, on utilise le mot clé `struct`. Syntaxe:

```
struct nomStructure {  
    type_1 nomMembre1 ;  
    type_2 nomMembre2 ;  
    ...  
    type_n nomMembreN ;  
}
```

Exemple :

```
struct Personne {  
    int age;  
    double poids;  
    double taille;  
};
```

Une fois la structure déclarée, on pourra définir des variables de type structuré.

Exemple:

```
Personne massamba, mademba;
```

Massamba pourra accéder à son âge en faisant `massamba.age`.

Rappel

Structures: déclaration

Exemple

```
#include<iostream>

using namespace std;

struct Personne
{
    int age;
    double poids;
    double taille;
};

int main(){
    Personne massamba;
    massamba.age=25;
    massamba.poids=90,5;
    massamba.taille=185,7;
    cout << " Massamba a " << massamba.age << " ans, il pèse " <<
    massamba.poids << " kg et il fait " << massamba.taille << " cm de long ." <<
    endl;
    return 0;
}
```

Rappel

Structures: initialisation

Dans l'exemple précédent, nous avons attribué une valeur champ après champ. Ce qui peut s'avérer long et peu pratique.

Il est en fait possible d'initialiser les champs d'une structure au moment de son instantiation grâce à l'opérateur {}.

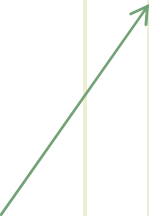
Exemple:

person.hh

```
#ifndef __PERSON_HH__
#define __PERSON_HH__

struct Personne
{
    int age;
    double poids;
    double taille;
};

#endif
```



```
#include <iostream>
#include "person.hh"
using namespace std;
int main()
{
    Personne massamba={25, 90.5, 185.7};

    cout << " Massamba a "
    << massamba.age << " ans, il pèse "
    << massamba.poids << " kg et il fait "
    << massamba.taille << " cm de long ."
    << endl;

    return 0;
}
```

Rappel

Structures et tableau

Une structure peut contenir un tableau. De ce fait, un espace mémoire lui sera réservé à sa création.

Exemple:

person.hh

```
#ifndef __PERSON_HH__
#define __PERSON_HH__

struct Personne
{
    char nom[20];
    int age;
    double poids;
    double taille;
};

#endif
```

```
#include<iostream>
#include "person.hh"
using namespace std;
int main()
{
    Personne m={'Massamba', 25, 90.5, 185.7};

    cout << m.nom <<" a "
    << m.age << " ans, il pèse "
    << m.poids << " kg et il fait "
    << m.taille << " cm de long ."
    << endl;

    return 0;
}
```

Rappel

structures imbriquées

Il est possible de créer des tableaux contenant des instances d'une même structure.

Exemple:

person.hh

```
#ifndef __PERSON_HH__
#define __PERSON_HH__

struct date {
    int jour;
    int mois;
    double annee;
};

struct Personne
{
    char nom[20];
    date date_de_naissance;
    double poids;
    double taille;
};

#endif
```

```
#include <iostream>
#include "person.hh"
using namespace std;
int main()
{
    Personne m[2]={
        {"Massamba", {8,8,2008}, 25.0, 185.7),
        {"Mafatou", {5,5,2010}, 30.6, 175.3)
    };

    cout << m[0].nom << " est né en "
    << m[0].date_de_naissance.annee << " , il
    pèse "
    << m[0].poids << " kg et il fait "
    << m[0].taille << " cm de long ."
    << endl;

    return 0;
}
```

Rappel

structures et fonctions

Une structure peut être passer à une fonction.

Exemple:

```
#include<iostream>
using namespace std;

struct date
{
    int jour;
    int mois;
    double annee;
};

struct Personne
{
    char nom[20];
    date date_de_naissance;
    double poids;
    double taille;
};

// la suite →
```

```
void saisirUser(Personne &p){
    cout << "Tapez le nom : ";
    cin >> p.nom;
    // ...
    cout << "Tapez la taille: ";
    cin >> p.taille;
}

int main(){
    Personne p;
    cout << "SAISIE DE P" << endl;
    saisirUser(p);

    cout << p.nom << " est né en "
    << p.date_de_naissance.annee << " , il pèse "
    << p.poids << " kg et il fait "
    << p.taille << " cm de long ."
    << endl;

    return 0;
}
```

Rappel

fonctions membres

On peut ajouter une fonction dans une structure.

Exemple:

```
#include<iostream>
#include<cmath>
using namespace std

struct date
{
    int jour;
    int mois;
    double annee;
};

struct Personne
{
    char nom[20];
    date date_de_naissance;
    double poids;
    double taille;
    double inMas(double p, double t);
};

double Personne::inMas(double p, double t)
{
    return p/pow(t;2);
}

// la suite →
```

```
void saisirUser(Personne &p){
    cout << "Tapez le nom : ";
    cin >> p.nom;
    // ...
    cout << "Tapez la taille: ";
    cin >> p.taille;
}

int main(){
    Personne p;
    cout << "SAISIE DE P" << endl;
    saisirUser(p);

    cout << p.nom << " est né en"
    << p.date_de_naissance.annee << " , il pèse "
    << p.poids << " kg, il fait "
    << p.taille << " cm de long et son IMC est de : "
    << p.inMas(p.poids, p.taille)
    << endl;

    return 0;
}
```


Rappel

Structures

EXERCICES D'APPLICATIONS

□ Application 24:

Soit la structure suivante:

```
struct point {  
    char a ;  
    int x, y ;  
}
```

Ecrire un programme faisant appel à une fonction recevant en argument l'adresse d'une structure de type point et qui renvoie une structure de même type correspondant à un point de même nom et de coordonnées opposées. Afficher les deux points.

□ Application 25:

En considérant la structure de type point de l'application 24, écrire pour chaque cas de figure, un programme appelant une fonction **afficher** qui prend en argument une structure de type point en le transmettant par:

- Par valeur
- Par adresse
- Par référence

La fonction affichera le point et ses coordonnées comme suit: « *le point A de coordonnées x=5 et y=7* ».