IN200 : Les structures

Sandrine Vial
sandrine.vial@uvsq.fr

Février 2016

Structures de données

- Permet de regrouper sous un seul nom un ensemble de variables de types différents.
- Chaque variable est appelée un **champ**
- Plusieurs étapes distinctes :
  1. Définition un modèle de structure.
  2. Déclaration de variables de type structuré.
  3. Utilisation de variables de type structuré.

Définition d'un modèle de structure

```
struct nom_de_la_structure
{
    Liste des champs de la structure
};
```

Un article qui a un numéro, un prix, une quantité :

```
struct article
{
  int numero;
  float prix;
  int qte;
};
```

Déclaration et Utilisation d'une variable structurée

- Déclaration de la variable A de type structuré article :

  `struct article A;`

- Utilisation d'une variable structurée : opérateur `.`
  - `A.numero` donne accès au champ `numero` de la variable `A`
  - `A.numero` est une variable de type `int`

```
struct article A;

A.numero = 23;
A.prix = 45.67;
A.prix = A.prix * 1.1;
```

Utilisation globale

- Possibilité de copier une variable de type structuré dans une autre variable de même type.

```
struct article A, B;

B.numero = A.numero;
B.qte = A.qte;
B.prix = A.prix;
```

équivaut à

```
struct article A, B;

B = A;
```

Opérateur `typedef`

- Permet de définir des équivalences (synonymes) entre types.
  - `typedef int entier;`
  - `typedef struct article ARTICLE;`
- Déclaration de variables :

```
int n;
struct article A;
```

équivaut à

```
entier n;
ARTICLE A;
```

Une utilisation : le type POINT

- Dans `graphics.h` :

```
typedef struct point {int x,y;} POINT;
```

- Utilisation dans vos codes :

```
POINT p1, p2;

p1.x = 100;
p1.y = 200;

p2 = p1;
```

Passage de paramètres

```
int dans_tableau_cercle(POINT P, int N, POINT TCentre[100])
{
   int i;
   for(i=0;i<N;i++)
   {
      if (est_dans_cercle(P,TCentre[i],50))
         return i;
   }
   return -1;
}
int main()
{
   int k,N = 0;
   POINT p, TCentre[100];
   for(i=0;i<100;i++)
   {
       p = wait_clic();
       k = dans_tableau_cercle(p,N,TCentre);
       if (k == -1)
       {
            TCentre[N] = p;
            N = N + 1;
       }
   }
}
```

Passage de paramètres

```
int dans_tableau_cercle(struct point P, int N, struct point TCentre[100])
{
   int i;
   for(i=0;i<N;i++)
   {
      if (est_dans_cercle(P,TCentre[i],50))
         return i;
   }
   return -1;
}
int main()
{
   int k,N = 0;
   struct point p, TCentre[100];
   for(i=0;i<100;i++)
   {
       p = wait_clic();
       k = dans_tableau_cercle(p,N,TCentre);
       if (k == -1)
       {
            TCentre[N] = p;
            N = N + 1;
       }
   }
}
```

Passage de paramètres

- Si l'on déclare `POINT *p;` :
  - `p` est une variable qui contient l'adresse d'un POINT.
  - `*p` est une variable de type POINT.
  - `(*p).x` est le champ `x` de la variable `*p`

```
void Initialise(POINT *p)
{
    POINT q;

    q = wait_clic();
    (*p).x = q.x;
    (*p).y = q.y;
}
```

Passage de paramètres

- Si l'on déclare `POINT *p;` :
  - `p` est une variable qui contient l'adresse d'un POINT.
  - `*p` est une variable de type POINT.
  - `(*p).x` est le champ `x` de la variable `*p`

```
void Initialise(POINT *p)
{
    POINT q;

    q = wait_clic();
    *p = q;
}
```

Passage de paramètres

- Si l'on déclare `POINT *p;` :
  - `p` est une variable qui contient l'adresse d'un POINT.
  - `*p` est une variable de type POINT.
  - `(*p).x` est le champ `x` de la variable `*p`

```
void Initialise(POINT *p)
{
    *p = wait_clic();
}
```

Exemple

```
typedef struct point { int x; int y;} POINT;

void Initialise(POINT *p)
{
    *p = wait_clic();
}

void main()
{
  POINT p,q;

  init_graphics(300,600);
  Initialise(&p);
  Initialise(&q);
  draw_rectangle(p,q,rouge);
  wait_escape();
}
```