IN100 : Cours 6 – Les Procédures

Sandrine Vial
sandrine.vial@prism.uvsq.fr

Octobre 2014

Les procédures

- Permet de structurer le code
- Permet de nommer une partie du code et de pouvoir faire appel à cette partie à différentes étapes du code.

Les procédures : un exemple

```
POINT a,b,c;

a.x = 50; a.y = 20;
b.x = 100; b.y = a.y;
c.x = 70; c.y = 100;
draw_line(a,b,bleu);
draw_line(b,c,vert);
draw_line(c,a,orange);
```

```
a.x = a.x + 100;
b.x = b.x + 100;
c.x = c.x + 100;
draw_line(a,b,bleu);
draw_line(b,c,vert);
draw_line(c,a,orange);
```

Les procédures

Une procédure est une fonction qui **modifie l'affichage ou qui peut avoir des effets sur les variables globales**, ...

```
void Nom_fonction(paramètres)
{
.... //Corps de la fonction
}
```

Les procédures : le corps

Le corps d'une procédure contient :

- Des déclarations de variables locales : utilisables uniquement dans le corps de la procédure.
- Des instructions : elles portent sur les variables locales, les variables globales et sur les paramètres.

Exemple

```
void dessine_quadrillage()
{
   POINT p1,p2;
   p1.y = 0;  p2.y = 400; p1.x = 0;
   while (p1.x < 600)
   {
        p2.x = p1.x;
        draw_line(p1,p2,white);
        p1.x = p1.x + 20;
   }
   p1.x = 0; p2.x = 600; p1.y = 0;
   while(p1.y < 400)
   {
        p2.y = p1.y;
        draw_line(p1,p2,white);
        p1.y = p1.y + 20;
   }
}
```

Les procédures : les paramètres effectifs

- Appel de la procédure en lui donnant en paramètre des valeurs.

  `Nom_Fonction(5,6);`
- Attention : les paramètres effectifs doivent "correspondre"" aux paramètres formels :
  - le *i*ème paramètre formel prend la valeur du *i*ème paramètre effectif.
  - Mêmes types.

Les procédures : un exemple

```
void dessine_triangle(POINT p1, POINT p2, POINT p3)
{
   draw_line(p1,p2,bleu);
   draw_line(p2,p3,vert);
   draw_line(p3,p1,orange);
 }

int main()
{
  POINT a,b,c;

  a.x = 50; a.y = 20;
  b.x = 100; b.y = a.y;
  c.x = 70; c.y = 100;
  dessine_triangle(a,b,c);
  a.x = a.x + 100;
  b.x = b.x + 100;
  c.x = c.x + 100;
  dessine_triangle(a,b,c);
}
```

Mécanisme de passage de paramètres

- A l'appel de la procédure, les paramètres effectifs sont évalués (ont une valeur).
- Les paramètres formels sont alloués avec comme valeur initiale, une **COPIE** des valeurs des paramètres effectifs
- A l'intérieur de la procédure, on travaille donc sur une copie des valeurs.

Les procédures

```
void decale_100(POINT p1, POINT p2)
{
    p1.x = p1.x + 100;
    p2.x = p2.x + 100;
    draw_rectangle(p1,p2,rouge);
}
int main()
{
POINT a,b;

  init_graphics(600,300);

  a.x = 50; a.y = 20;
  b.x = 100; b.y = 120;
  draw_rectangle(a,b,bleu);
  decale_100(a,b);
  wait_clic();
  draw_rectangle(a,b,noir);
  wait_escape();
  return 1;
}
```

Les procédures

```
void echange(int a, int b)
{
    int c;
    write_int(a); write_text(" "); write_int(b); writeln();
    c = a;
    a = b;
    b = c;
    write_int(a); write_text(" "); write_int(b); writeln();
}

int main()
{
int a,b;

  init_graphics(600,300);
  a = 10;
  b = 20;
  write_int(a); write_text(" "); write_int(b); writeln();
  echange(a,b);
  write_int(a); write_text(" "); write_int(b); writeln();
  wait_escape();
  return 1;
}
```

Les variables globales

- Les variables locales :
  - Déclarées au sein d'une fonction
  - Utilisables uniquement dans la fonction dans laquelle elles ont été déclarées
  - Durée de vie : celle de la fonction
- Les variables globales :
  - Déclarées en dehors de toute fonction
  - Utilisables dans n'importe quelle fonction
  - Durée de vie : celle du programme

Les variables globales

```
int a,b;
void echange()
{
    int c;
    write_int(a); write_text(" "); write_int(b); writeln();
    c = a;
    a = b;
    b = c;
    write_int(a); write_text(" "); write_int(b); writeln();
}

int main()
{

  init_graphics(600,300);
  a = 10;
  b = 20;
  write_int(a); write_text(" "); write_int(b); writeln();
  echange();
  write_int(a); write_text(" "); write_int(b); writeln();
  wait_escape();
  return 1;
}
```

Les variables globales

Avantage

On peut utiliser/modifier une variable globale n'importe où dans le code

Inconvénients

- On peut utiliser/modifier une variable globale n'importe où dans le code
- Les procédures sont moins réutilisables.

Une solution : les variables globales

Et si une variable globale porte le même nom qu'une variable locale ?

```
int a;

void modifie(int b)
{
   int a;

   b = b + 10;
   a = b * 2;
}

int main()
{
   int b;

   a = 10;
   b = 30;
   modifie(b);
   modifie(a);
}
```