Programmation structurée

Procédures et fonctions en DELPHI

Ricco Rakotomalala
Université Lumière Lyon 2

Pourquoi le découpage du programme en procédures et fonctions ?

Problème : A mesure que la taille du programme augmente, (1) il faut une meilleure **organisation** pour en maintenir la lisibilité ; (2) des parties du code peuvent être **réutilisées** à plusieurs endroits.

Bannir le copier /coller parce que la maintenance du code devient impossible

Le programme principal doit être simple c.-à-d. se résumer – autant que possible – à des appels aux procédures et fonctions

Ecriture d'une fonction

Idée

Associer dans la même structure un bloc de code associé à une fonctionnalité identifiée

Forme générique

Nom_paramètre : type_de_données
(si plusieurs, séparés par des « ; »)

```
function nom_de_fonction(arguments…) : type_valeur_de_retour;
var… définition des variables locales à la fonction
Begin
  … instruction(s);
  nom_de_fonction:= valeur à renvoyer;
End;
```

On peut également utiliser le mot-clé « result » c.-à-d.
result:= valeur à renvoyer;

Un exemple : organisation du code et appel de la fonction dans le programme principal

ProjetTVA.dpr

ProjetTVA

```
program ProjetTVA;

{$APPTYPE CONSOLE}

uses SysUtils;

//****définition de la fonction ****
function calcPttc(ht: double; tva: integer): double;
//variable locale à la fonction
var ttc: double;
begin
 //selon le code de tva
 if (tva = 1)
  then ttc:= ht * 1.196
  else ttc:= ht * 1.33;
 //renvoyer les résultats
 result:= ttc;
end;

//**** variables globales du prog. principal ****
var pht, pttc: double;
    code: integer;
//**** début du prog. principal ****
begin
 //saisie
 write('Pht = '); readln(pht);
 write('code tva (1 ou 2) = '); readln(code);
 //appel de la fonction
 pttc:= calcPttc(pht,code);
 //affichage
 writeln('pttc = ', pttc);
 readln;
end.
```

6: 1 Insertion Code

Remarquez :

1. Organisation globale du code (où placer la fonction, où l'appeler)
2. La correspondance entre les paramètres (pas de correspondance sur les noms, mais les types doivent être cohérents)
3. Le séparateur de paramètres pour la définition et pour l'appel de la fonction

La procédure

Idée

Une procédure une fonction qui ne renvoie pas de valeurs. Sert par ex. pour les affichages, pour les écritures dans les fichiers, ou la manipulation de tableaux.

On peut quand même renvoyer des valeurs en « rusant » avec le mode de passage des paramètres (par variable ou par référence)

```
procedure nom_de_procedure(arguments…);
var… définition des variables locales à la fonction
Begin
  … instruction(s);
End;
```

Passage de paramètres par valeur

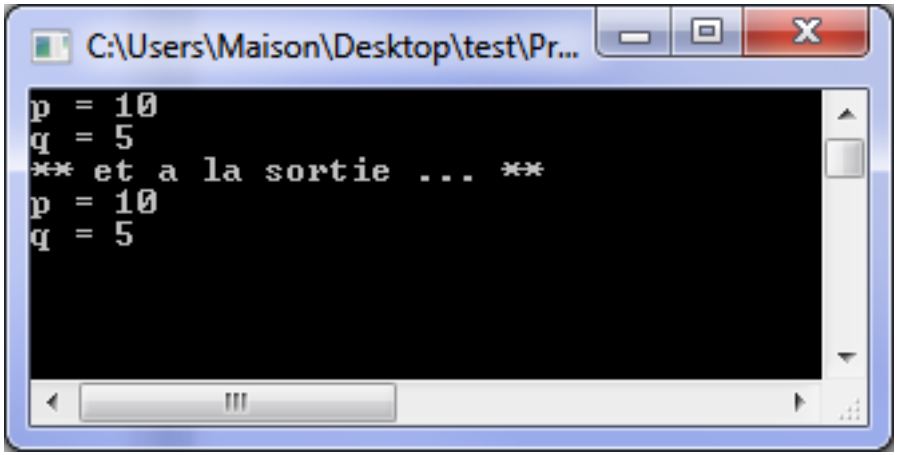
ProjetEchange.dpr

ProjetEchange

```
program ProjetEchange;

{$APPTYPE CONSOLE}

uses SysUtils;

//procedure
procedure echange(a,b: integer);
var tmp: integer;
begin
 tmp:= a;
 a:= b;
 b:= tmp;
end;

var p,q: integer;
begin
 write('p = '); readln(p);
 write('q = '); readln(q);
 echange(p,q);
 writeln('** et a la sortie ... **');
 writeln('p = ',p);
 writeln('q = ',q);
 readln;
end.
```

1: 1 Insertion Code

Le mode de passage par défaut est le passage par valeur c.-à-d. les modifications locales à la fonction ne sont pas répercutées dans le prog. principal.



Passage de paramètres par variable (par référence)

ProjetEchange.dpr

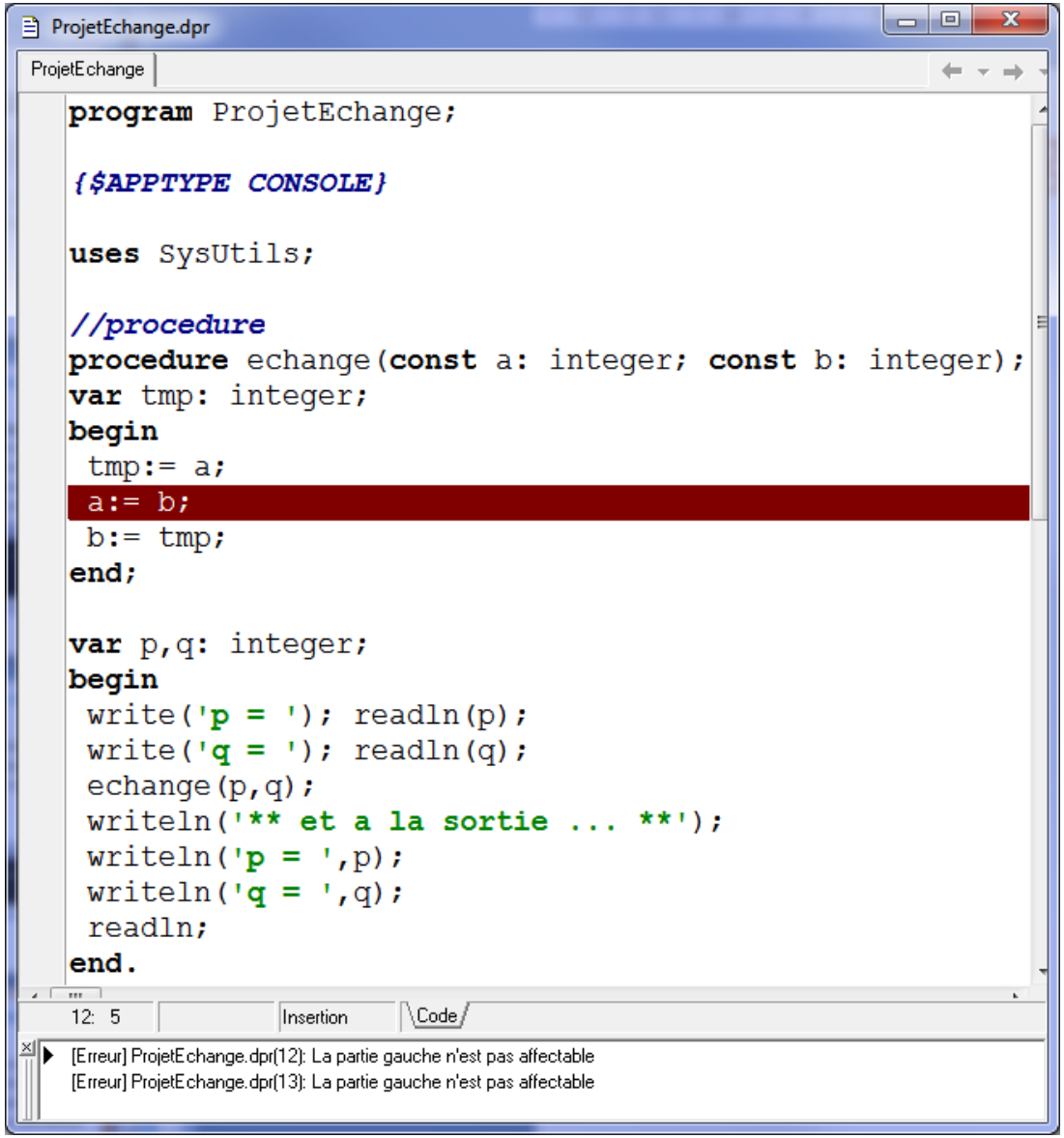
ProjetEchange

```
program ProjetEchange;

{$APPTYPE CONSOLE}

uses SysUtils;

//procedure
procedure echange(var a: integer; var b: integer);
var tmp: integer;
begin
 tmp:= a;
 a:= b;
 b:= tmp;
end;

var p,q: integer;
begin
 write('p = '); readln(p);
 write('q = '); readln(q);
 echange(p,q);
 writeln('** et a la sortie ... **');
 writeln('p = ',p);
 writeln('q = ',q);
 readln;
end.
```

8: 1 Insertion Code

Pour que les modifications locales à la fonction soient répercutées, il faut utiliser le mot clé « **var** ».

Passage de paramètres par constante



Avec « **const** », les modifications locales sont interdites.

Quel intérêt ? Rapidité car les paramètres ne sont pas dupliqués en mémoire.

Ex. Calculs sur les grands tableaux, la duplication locale est gourmande en ressources machines.

Appels entre fonctions

```
program ProjetTVACroise;

{$APPTYPE CONSOLE}

uses SysUtils;

//****définitions fonctions ****

function calcTVA(codeTva: integer): double;
begin
 if (codeTva = 1)
  then result:= 1.196
  else result:= 1.33;
end;

function calcPttc(ht: double; tva: integer): double;
//variable locale à la fonction
var ttc: double;
begin
 //appel de la fonction
 ttc:= ht * calcTva(tva);
 //retour des résultats
 result:= ttc;
end;

//**** variables globales du prog. principal ****
var pht, pttc: double;
    code: integer;
//**** début du prog. principal ****
begin
 //saisie
 write('Pht = '); readln(pht);
 write('code tva (1 ou 2) = '); readln(code);
 //appel de la fonction calcTVA
 writeln('niveau de TVA utilise = ', calcTVA(code));
 //appel de la fonction
 pttc:= calcPttc(pht,code);
 //affichage
 writeln('pttc = ', pttc);
 readln;
end.
```

1. En cas de changement de la législation (niveaux de TVA différents), les modifications portent sur un seul endroit du programme : la fonction calcTVA().
2. On peut appeler une fonction dans une autre fonction. Ex. calcTVA est appelé dans calcPttc

FIN…

Les mêmes concepts sont – à peu de choses près – présents dans tous les langages de programmation…