Type fonction et procédure

I3 - Algorithmique et programmation

Nicolas Delestre

Plan

Architecture des Systèmes d'Information

Introduction 1 / 2

Quelques constats, quelques questions ?

1. Algorithme sur les tris : que l'on trie en ordre croissant ou en ordre décroissant, l'algorithme est à 99,99% le même. Comment éviter de dupliquer du code ?
2. Quelque soit la fonction mathématique f de $\mathbb{R} \rightarrow \mathbb{R}$, l'algorithme de calcul de l'intégrale par la méthode des trapèzes, ou l'algorithme de la résolution de $f(x) = 0$ à ϵ près par un algorithme dichotomique ne changent pas. Comment rendre le code de résolution, indépendant de la fonction mathématique ?
3. On doit séparer la logique métier (LM) de l'IHM. Mais comment synchroniser la LM et l'IHM, de façon à ne pas être obligé de changer la LM lorsque l'on change l'IHM ?

Architecture des Systèmes d'Information

Introduction 2 / 2

Réponses

- Il faudrait pouvoir passer des fonctions ou des procédures en paramètre
  1. On doit passer la fonction de comparaison en paramètre, c'est-à-dire la fonction qui est vrai lorsque deux éléments du tableau doivent être échangés
  2. On doit passer la fonction f en paramètre
  3. L'IHM (qui est le programme principal) passe à la logique métier une fonction qui lui permettra d'être prévenue lorsque la logique métier changera

Questions

- Les paramètres formels ont un type, qu'est ce que le type d'une fonction ou d'une procédure ?
- Comment on utilise un paramètre formel qui est une fonction ou une procédure ?
- Comment on spécifie qu'un paramètre effectif est une fonction ou une procédure ?

Architecture des Systèmes d'Information

Remarques

- On répond à ces questions dès l'analyse
- Il faut donc pouvoir exprimer ces besoins au niveau de :
  - l'analyse. Comment représenter cela dans l'analyse descendante ?
  - la conception préliminaire. Comment représenter cela dans la signature d'une fonction ou d'une procédure ? Quel est le type de passage de paramètre ?
  - la conception détaillée. Comment le pseudo-code permet-il d'utiliser une fonction ou une procédure ?
  - le développement. Le langage permet-il cela et comment ?

Analyse

- Dans l'analyse descendante, nous allons supposer que l'on a des types représentant des opérations
- Les identifiants de ces types vont représenter :
  - des actions, donc commencer par des verbes (par exemple Comparer, Afficher, etc.)
  - des calculs, donc commencer par un nom le représentant(par exemple FonctionDeRDansR)

Exemple

FonctionRDansR
Reel
Reel
NaturelNonNul

Conception préliminaire

Déclaration du type

- Pour déclarer un type fonction (ou procédure), il suffit de reprendre lors de la déclaration la signature de la fonction (ou procédure) en omettant le nom

Exemple

Type FonctionDeRDansR = **fonction**(x : **Reel**) : **Reel**

Paramètre formel et passage de paramètre

- On déclare un paramètre formel de type fonction (ou procédure) comme tout type de paramètre
- Le passage de paramètre est très souvent un passage de paramètre en entrée

Exemple

fonction integrale (f : FonctionDeRDansR, a,b : **Reel**, n : **NaturelNonNul) : Reel**

⌊**précondition(s)** a<b

Conception détaillée 1 / 2

Utilisation du paramètre formel

- On utilise le paramètre formel comme tout autre fonction (ou procédure)

Exemple

fonction integrale (f : FonctionDeRDansR ; a,b : **Reel** ; n : **NaturelNonNul**) : **Reel**

⌊**précondition(s)** $a \leq b$

Déclaration x1, Δ, somme : **Reel**

debut

somme $\leftarrow$ (f(a)+f(b)/2)))
$\Delta \leftarrow$ (b - a)/n
x $\leftarrow$ a
pour i $\leftarrow$2 **à** n-1 **faire**
x $\leftarrow$ x + Δ
somme $\leftarrow$ somme + f(x)
finpour
somme $\leftarrow$ somme * Δ
retourner somme

fin

Conception détaillée 2 / 2

Utilisation du paramètre effectif

- On utilise comme paramètre effectif l'identifiant d'une fonction (ou procédure) préalablement déclarée et de même type

Exemple

```
fonction g (x : Reel) : Reel
debut
    retourner 2*x
fin
...
y ← integrale(g,1,10,100)
...
```

En Pascal 1 / 5

Déclaration, utilisation

- On déclare un type fonction (ou procédure) comme pour la conception préliminaire (avec les mots clés `function` et `procedure`)
- On déclare et utilise un paramètre formel comme dans les conceptions préliminaire et détaillée
- On utilise l'identifiant d'une procédure préalablement déclarée comme paramètre effectif en le préfixant du symbole @

En Pascal 2 / 5

Exemple : methodesNumeriques.pas

```
unit methodesNumeriques;

interface
type
   TFonctionDeRDansR = function(x : Real) : Real;

function integrale(f : TFonctionDeRDansR;
                   a,b : Real;
                   n : LongWord) : Real;
```

En Pascal 3 / 5

Exemple : methodesNumeriques.pas (suite)

```
implementation

function integrale(f : TFonctionDeRDansR;
                   a,b : Real;
                   n : LongWord) : Real;
var
   i : LongWord;
   x,delta : Real;
begin
   integrale:=(f(a)+f(b))/2;
   delta:=(b-a)/n;
   x:=a;
   for i:=2 to n-1 do
   begin
      x:=x+delta;
      integrale:=integrale+f(x)
   end;
   integrale:=integrale*delta
end;
```

En Pascal 4 / 5

Exemple : test.pas

```
program test;

uses methodesNumeriques;

function g(x : Real) : Real;
begin
   g:=cos(x)
end;

procedure testIntegrale();
begin
   writeln('integrale(g,0,2*PI,100) = 0 a 0.1 pres ',
           abs(integrale(@g,0,2*PI,100))<0.1)
end;

begin
   testIntegrale()
end.
```

En Pascal 5 / 5

Exercice

- Ajouter à l'unité `methodesNumeriques` une fonction qui permet de résoudre l'équation $f(x) = 0$ entre a et b (tel que $a < b$) à ϵ près

Tri croissant ou décroissant 1 / 3

Rappel tri à bulles (tri croissant)

```
procédure triABulles (E/S t :Tableau[1..MAX] d'Entier,E
nb :Naturel)
    Déclaration  i : Naturel
                 estTrie : Booleen
debut
    repeter
        estTrie ← VRAI
        pour i ←1 à nb-1 faire
            si t[i]>t[i+1] alors
                echanger(t[i],t[i+1])
                estTrie ← FAUX
            finsi
        finpour
    jusqu'a ce que estTrie
fin
```

Tri croissant ou décroissant 2 / 3

Définition des comparateurs

Type ComparerDeuxEntiers = **fonction**(a,b : **Entier**) : **Booleen**

fonction plusPetit (a,b : **Entier**) : **Booleen**
debut
    retourner a<b
fin

fonction plusGrand (a,b : **Entier**) : **Booleen**
debut
    retourner a>b
fin

Tri croissant ou décroissant 3 / 3

Nouvelle version du tri à bulles

procédure triABulles **(E/S** t :**Tableau[**1..MAX**] d'Entier**, **E** nb :**Naturel**, entiersAEchanger : ComparerDeuxEntiers**)**

```
    Déclaration   i : Naturel
                  estTrie : Booleen
debut
    repeter
        estTrie ← VRAI
        pour i ←1 à nb-1 faire
            si entiersAEchanger(t[i],t[i+1]) alors
                echanger(t[i],t[i+1])
                estTrie ← FAUX
            finsi
        finpour
    jusqu'a ce que estTrie
fin
```

Exercice

- Quelle fonction faut-il utiliser comme paramètre effectif pour faire un tri en ordre croissant ?

Jeu de nim 1 / 12

Règle de jeu (version simplifiée)

« . . .Une version basique de ce jeu utilise un seul tas d'objets. Chaque joueur à tour de rôle enlève 1, 2 ou 3 objets. Le vainqueur est celui qui peut jouer en dernier. »

Objectif

- Écrire un programme qui permet à deux joueurs de jouer au jeu de nim en précisant au lancement du jeu combien d'objets il y aura en début de partie. Le sous-programme qui permet de jouer au jeu doit être indépendant des caractéristiques de l'IHM (texte, graphique, etc.). Cette version du jeu sera en mode texte.

Jeu de nim 2 / 12

Analyse générale

- Logique métier : sous programme qui prend le nombre d'objets en entrée et qui, après affichage de l'état du jeu, demande successivement au joueur le nombre d'objets à prendre (après chaque prise, il indique aussi combien d'objets il reste). Il retourne ensuite celui qui a gagné.
- IHM : une fonction pour demander combien d'objets un joueur veut prendre et une procédure pour afficher le nombre d'objets.

Architecture des Systèmes d'Information

Jeu de nim 3 / 12

Conception préliminaire

Constante NB_JOUEURS = 2
Constante NB_OBJETS_A_PRENDRE_MIN = 1
Constante NB_OBJETS_A_PRENDRE_MAX = 3

Type Joueur = 1..NB_JOUEURS
Type Coup = NB_OBJETS_A_PRENDRE_MIN..NB_OBJETS_A_PRENDRE_MAX
Type ObtenirCoup = **fonction**(joueurCourant : Joueur, nbObjetsRestant : **NaturelNonNul**) : Coup
Type AfficherEtatJeu = **procédure**(**E** nbObjets : **Naturel**)

fonction jouer (obtenirCoup1, obteniCoup2 : ObtenirCoup, afficher : AfficherEtatJeu, nbObjets : **Naturel**) **:** Joueur
⌊**précondition(s)** nbObjets>0

Jeu de nim 4 / 12

Conception détaillée

```
fonction jouer (obtenirCoup1, obtenirCoup2 : ObtenirCoup, afficher : AfficherEtatJeu,
nbObjets : Naturel) : Joueur
    ⌊précondition(s)   nbObjets>0
debut
    joueurCourant ← 1
    tant que nbObjets>0 faire
        afficher(nbObjets)
        si joueurCourant=1 alors
            coup ← obtenirCoup1(joueurCourant,nbObjets)
        sinon
            coup ← obtenirCoup2(joueurCourant,nbObjets)
        finsi
        nbObjets ← nbObjets-coup
        si nbObjets>0 alors
            joueurCourant ← autreJoueur(joueurCourant)
        finsi
    fintantque
    retourner joueurCourant
fin
```

Jeu de nim 5 / 12

jeuDeNim.pas

```
1  unit jeuDeNim;
2
3  interface
4
5  const
6     NB_JOUEURS = 2;
7     NB_OBJETS_A_PRENDRE_MIN = 1;
8     NB_OBJETS_A_PRENDRE_MAX = 3;
9
10 type
11    TJoueur = 1..NB_JOUEURS;
12    TCoup = NB_OBJETS_A_PRENDRE_MIN..NB_OBJETS_A_PRENDRE_MAX;
13    TObtenirCoup = function(joueurCourant : TJoueur;
14                            nbObjetsRestant : Word) : TCoup;
15    TAfficherEtatJeu = procedure(nbObjets : Word);
16
17
18 function jouer(obtenirCoupJoueur1,
19                 obtenirCoupJoueur2 : TObtenirCoup;
20                 afficher : TAfficherEtatJeu;
21                 nbObjets : Word) : TJoueur;
```

Jeu de nim 6 / 12

jeuDeNim.pas (suite)

```
22 implementation
23
24 function autreJoueur(joueur : TJoueur):TJoueur;
25 begin
26    autreJoueur:=(joueur mod 2)+1
27 end;
28
29 function jouer(obtenirCoupJoueur1,
30                 obtenirCoupJoueur2 : TObtenirCoup;
31                 afficher : TAfficherEtatJeu;
32                 nbObjets : Word) : TJoueur;
33  var
34     joueurCourant : TJoueur;
35     coup : TCoup;
36  begin
37     joueurCourant:=1;
38     while (nbObjets > 0) do
39     begin
40        afficher(nbObjets);
```

Jeu de nim 7 / 12

jeuDeNim.pas (fin)

```
41          if joueurCourant=1 then
42              coup:=obtenirCoupJoueur1(joueurCourant,nbObjets)
43          else
44              coup:=obtenirCoupJoueur2(joueurCourant,nbObjets);
45          nbObjets:=nbObjets-coup;
46          if nbObjets>0 then
47              joueurCourant:=autreJoueur(joueurCourant)
48      end;
49      jouer:=joueurCourant
50  end;
51
52  end.
```

Jeu de nim 8 / 12

jeuDeNimTxtHumainHumain.pas

```
1  program jeuDeNimTxtHumainHumain;
2
3  uses jeuDeNim,minMax;
4
5  function obtenirCoupJoueur(joueurCourant : TJoueur;
       nbObjetsRestant : Word) : TCoup;
6  begin
7     repeat
8        writeln('C''est au joueur ',joueurCourant,' de jouer (
             nombre compris entre ',NB_OBJETS_A_PRENDRE_MIN,' et ',
             min(NB_OBJETS_A_PRENDRE_MAX,nbObjetsRestant),') : ');
9        readln(obtenirCoupJoueur);
10    until (obtenirCoupJoueur>=NB_OBJETS_A_PRENDRE_MIN) and (
          obtenirCoupJoueur<=min(NB_OBJETS_A_PRENDRE_MAX,
          nbObjetsRestant));
11 end;
```

Jeu de nim 9 / 12

jeuDeNimTxtHumainHumain.pas (suite)

```
12 procedure afficherEtatDuJeu(nbObjets : Word);
13 begin
14    writeln('Il reste ',nbObjets,' objet(s) a prendre')
15 end;
16
17 function nbObjetsAuDepart() : Word;
18 begin
19    repeat
20       writeln('Combien d''objets au depart (superieur a 0):');
21       readln(nbObjetsAuDepart);
22    until nbObjetsAuDepart>0
23 end;
```

Jeu de nim 10 / 12

jeuDeNimTxtHumainHumain.pas (fin)

```
24 var
25    quiAGagne :TJoueur;
26 begin
27    quiAGagne:=jouer(@obtenirCoupJoueur,@obtenirCoupJoueur,
          @afficherEtatDuJeu, nbObjetsAuDepart());
28    writeln('C''est le joueur ',quiAGagne,' qui a gagne')
29 end.
```

Jeu de nim 11 / 12

Exemple d'exécution

```
$ ./jeuDeNimTxtHumainHumain
Combien d'objets au depart (superieur a 0):
10
Il reste 10 objet(s) a prendre
C'est au joueur 1 de jouer (nombre compris entre 1 et 3) :
3
Il reste 7 objet(s) a prendre
C'est au joueur 2 de jouer (nombre compris entre 1 et 3) :
2
Il reste 5 objet(s) a prendre
C'est au joueur 1 de jouer (nombre compris entre 1 et 3) :
3
Il reste 2 objet(s) a prendre
C'est au joueur 2 de jouer (nombre compris entre 1 et 3) :
2
C'est le joueur 2 qui a gagne
```

Jeu de nim 12 / 12

Exercice

- Dans l'article Wikipédia sur le jeu de nim, il est indiqué que « Pour cet exemple, la stratégie est de laisser à chaque fois - si on le peut - un nombre d'objets multiple de 4 : ce sont les positions gagnantes. On constate alors que l'adversaire ne pourra pas en faire autant. »
- Développez une nouvelle version du jeu qui permet à un humain de jouer contre l'ordinateur