De petits pas dans...

LE GRAND MONDE

Ada, un langage pour tout le monde

Par Luke A. Guest

Introduction

Faisant suite au numéro 6, nous allons continuer à apprendre les bases du langage Ada.

Types numériques (suite)

Les types entiers peuvent avoir une valeur négative, comme -10, -55, etc. Les types naturels ne peuvent accepter des valeurs qu'à partir de 0 (donc, pas de nombre négatif) et les types positifs ne peuvent prendre que des valeurs à partir de 1 (pas de valeur négative ni de zéro).

Tous ces types peuvent être utilisés ensemble dans des expressions mathématiques, mais il faut s'assurer de ne pas provoquer de débordement selon le type qui leur est affecté, sinon une erreur pourra se produire. Par exemple, si vous définissez deux variables X : Natural := 1; et Y : Integer := 2; et ensuite Y est soustrait de X puis réaffecté à X (c.-à-d. X := X - Y), cela provoquera une erreur car le résultat vaut -1 ce qui est en dehors des limites autorisées pour les types Natural.

Types booléens

Les types booléens possèdent deux valeurs, True (vrai) et False (faux), il n'y a rien d'autre qui puisse être attribué à une variable de type Boolean.

Décisions simples

Tous les langages ont la notion de valeurs booléennes car toutes les conditions en programmation reposent sur le fait qu'une chose est soit vraie soit fausse.

Tapez le code du listing 1 pour voir comment nous pouvons prendre des décisions en Ada en utilisant le type booléen.

Littéraux

Nous avons déjà vu quelques littéraux, même si nous ne savons pas de quoi il s'agit. Un littéral est un morceau de données dans le code source

```
1     with Ada.Text_IO;
2     use Ada.Text_IO;
3
4     procedure Decisions is
5        Is_Defective : Boolean := False;
6     begin
7        if Is_Defective = True then
8           Put_Line ("Defective");
9        else
10          Put_Line ("Not defective");
11       end if;
12    end Decisions;
```

Listing 1: decisions.adb

Ligne 5 : Nous définissons une variable booléenne, Is_Defective et lui donnons la valeur False.

Ligne 7 : Nous testons pour savoir si **Is_Defective = True**. Le = signifie que ce qui est à gauche est égal ou identique à ce qui est à droite. La partie entre le **if** (si) et le **then** (alors) est appelée une expression. Nous aurions aussi pu simplement mettre if **Is_Defective then** pour dire la même chose.

De même, **if Is_Defective = False** then et **if not Is_Defective then** sont similaires.

Ligne 8 : Tout ce qu'il y a entre les mots-clés **then** (alors) et **else** (sinon) est exécuté quand la condition est vraie.

Ligne 10 : Tout ce qu'il y a entre les mots-clés **else** (sinon) et **end if** (fin si) est exécuté quand la condition est fausse.

Ligne 11 : Tous les blocs **if** doivent se clôturer avec **end if**; même s'il n'y a pas de **else**.