Types ordinaux

1 Types ordinaux

1.1 Définition

Type *ordinal* (ou *scalaire* ou *discret*) = type tel que

1. l'ensemble des valeurs est totalement ordonné ;
2. existence d'un plus grand et d'un plus petit élément ;
3. chaque valeur possède un successeur (sauf la plus grande), et un prédécesseur (sauf la plus petite).

Exemples Voici quelques types ordinaux prédéfinis en Free Pascal.

booléens BOOLEAN

caractères CHAR

entiers CARDINAL, INTEGER

Exemples de types non ordinaux

réels REAL

chaînes de caractères STRING

1.2 Fonctions et procédures sur les types ordinaux

1. Fonctions sur les types : **low** et **high**. Elles donnent respectivement les valeurs minimales et maximales du type.

```
low(BOOLEAN) = false
high(BOOLEAN) = true

low(CARDINAL) = 0
high(CARDINAL) = 4294967295
```

2. Opérateurs de comparaison de deux valeurs d'un même tye ordinal : =, <>, <, <=, >, >=
3. Fonctions : **succ**, **pred**, **ord**. Les deux premières fonctions s'appliquent à des expressions d'un type ordinal et donne la valeur de ce type qui suit pour **succ**, qui précède pour **pred**. La fonction **ord** donne un entier donnant le rang ordinal dans le type de la valeur passée en paramètre.

```
succ(123) = 124
pred(123) = 122
ord(123) = 123

succ('A') = 'B'
pred('A') = '@'
ord('A') = 65

succ(false) = true
ord(false) = 0
```

La fonction **succ** ne peut s'appliquer à la plus grande valeur d'un type ordinal. De même la fonction **pred** ne peut s'appliquer à la plus petite valeur. Toute tentative d'appel à l'uune ou l'autre de ces fonctions dans une situation extrême se traduit en général par l'arrêt de l'exécution du programme avec le message `Runtime error 201` .

4. Procédures : **inc**, **dec**. Ces deux procédures s'appliquent à toute variable de type ordinal. L'instruction **inc**(n) est équivalente à l'instruction n := **succ**(n), et l'instruction **dec**(n) est équivalente à l'instruction n := **pred**(n).

   $\{n = 123\}$
   inc(n);
   $\{n = 124\}$

   $\{c =' B'\}$
   dec(c);
   $\{c =' A'\}$

   Ces deux procédures ne peuvent être appelées avec des variables ayant une valeur extrême du type ordinal.

1.3 Retour sur les boucles pour

Tout type ordinal peut être le type d'un indice de boucle pour.

1.4 Définitions de types ordinaux

1.4.1 Déclaration de types

```
type < nom > = < definition > ;
```

1.4.2 Type intervalle

On peut définir des types intervalle à partir de tout type ordinal.

```
type
    CHIFFRE = 0..9;
    LETTRE  = 'A'..'Z';
```

Les intervalles sont des sous-ensembles de l'ensemble des valeurs d'un type ordinal. À ce titre toute opération possible sur le type de base est possible sur le type intervalle.

1.4.3 Type énuméré

```
type
    JOUR = (LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI, SAMEDI, DIMANCHE);
```

Les valeurs intervenant dans l'énumération sont des nouvelles valeurs identifiées par des noms (identificateurs). Ces noms ne sont pas des chaînes de caractères (pas d'apostrophes entourant les noms). La valeur ordinale du premier élément de l'énumération est 0.

high(JOUR) = DIMANCHE
succ(LUNDI) = MARDI
ord(LUNDI) = 0

Mises à part les fonctions et procédures s'appliquant à tout type ordinal, il n'existe aucune fonction et procédure prédéfinie pour les types énumérés. En particulier, les procédures d'entrée/sortie (en particulier **readln** et **write**(**ln**)) ne peuvent pas être utilisées.

On peut aussi créer des intervalles de valeurs énumérées.

```
type
    WEEKEND = SAMEDI..DIMANCHE;
    JOUROUVRE = LUNDI..VENDREDI;
```

2 Instruction case

Syntaxe de l'instruction

```
case < expression > of
  < valeur > : < instruction >;
  < valeur > : < instruction >;
      ...
else
  < instruction >;
end {case};
```

L'expression < *expression* > doit être d'un type ordinal, et les valeurs < *valeur* > doivent être des constantes, des énumérations ou des intervalles de valeurs du même type que l'expression.

La partie **else** d'une instruction **case** est facultative.

Exemple 1

n est une variable de type CARDINAL

```
case n of
  1 : writeln('lundi');
  2 : writeln('mardi');
  3 : writeln('mercredi');
  4 : writeln('jeudi');
  5 : writeln('vendredi');
  6 : writeln('samedi');
  7 : writeln('dimanche');
end {case};
```

Exemple 2

n est une variable de type CARDINAL

```
case n of
  0..9 : writeln('inférieur à 10');
  10..100 : writeln('compris entre 10 et 100');
else
  writeln('très grand');
end {case};
```

Exemple 3

c est une variable de type CHAR

```
case lowercase(c) of
  'a','e','i','o','u','y' : writeln('voyelle');
else
  writeln('autre chose');
end {case};
```

Attention

n est une variable de type CARDINAL

```
case n of
  5 : writeln('cinq');
  0..9 : writeln('inférieur à 10');
end {case};
```

Les deux cas ne sont pas exclusifs. Lorsque n vaut 5, les deux cas peuvent s'appliquer. Le compilateur FREE PASCAL refuse les instructions **case** dont les cas ne sont pas exclusifs, et le signale en affichant le message `Error : duplicate case label`.

3 Exercices

Exercice 1 : Écrire un programme qui affiche à l'écran les valeurs extrêmales des types ordinaux CARDINAL, INTEGER, BOOLEAN, CHAR, CHIFFRE, LETTRE.

Exercice 2 : Réalisez les fonctions **succ** et **pred** pour les caractères en utilisant les procédures **inc** et **dec**.

Exercice 3 : Écrire une procédure d'affichage des valeurs du type JOUR.

Exercice 4 : Réalisez la fonction dont voici la spécification

```
// jourSuivant(j) = jour qui suit dans l'énumération
// en considérant que jourSuivant(DIMANCHE) = LUNDI
function jourSuivant(j : JOUR) : JOUR;
```

Essayez d'en trouver deux réalisations :

1. l'une avec la fonction **high** et une instruction conditionnelle,
2. l'autre sans instruction conditionnelle et sans les fonctions **high** et **low**.

Exercice 5 : Réalisez un programme qui affiche chaque caractère du type CHAR accompagné de son rang ordinal. (Seuls les caractères à partir de l'espace (rang ordinal 32) sont affichables.)

Exercice 6 : Réalisez une fonction qui transforme les caractères de type CHIFFRE en la valeur de type CARDINAL correspondante. Vous en donnerez deux réalisations :

1. l'une utilisant l'instruction **case**,
2. l'autre utilisant la fonction **ord**.

Exercice 7 : *Code de César*

Le code de César consiste à changer toutes les lettres d'un texte en la lettre décalée de trois positions de l'alphabet. Ainsi le `A` devient un `D`, le `B` un `E`, ..., le `X` un `A`, le `Y` un `B` et le `Z` un `C`.
Q 1 . Réalisez une fonction qui transforme toutes les lettres en leur correspondant selon le code de César. (Vous pourrez supposer que les lettres sont toutes des majuscules non accentuées).
Q 2 . Utilisez cette fonction pour écrire un programme codant n'importe quel message.