I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**

- En Pascal, on utilise le type **string**, qui n'est pas un type standard. Il est construit a partir d'un tableau de 255 caractères maximum.
- Ce type permet de manipuler des chaines de **longueur variable**.
- Il est possible de définir un sous-type, comportant moins de caractères :

- Exemple : Déclarations

```
var
s : string;
s2 : string[20];
```

39

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**

- Pour acceder a un caractere particulier de la chaine s, on ecrit simplement, comme pour *un tableau* : s[i]
- On peut aussi manipuler la chaine de manière globale, ce qui est très utile pour des affectations ou des tests .

- Exemple

```
s := 'Bonjour';
s[4] := 's';
s[6] := 'i'; { a present, s vaut 'Bonsoir' }
s := 'ok';   { a present, s vaut 'ok'}
```

Remarque : La taille de s est variable (dans l'exemple elle est de 7 caracteres au depart et de 2 ensuite).

40

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**
- **Opérateurs**

a) Operateurs de comparaison

- Il est possible de comparer des chaines de caractères avec les operateurs : =, <, >, <=, >=, <>
- L'ordre utilise est l'ordre lexicographique (utilisation du code ASCII).

- Exemple

```
s1:='salut'
s2:='bonjour'
s3:='bonsoir '
```

alors : s1 > s3 > s2

41

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**
- **Opérateurs**

- b) Concaténation
- Définition
- La concaténation est une opération qui permet d'accoler 2 ou plusieurs chaines de caractères.

- Syntaxe : Pascal
- `s:= s1 + s2 + s3... ;`

- Exemple

```
s1:='bon';
s2:='jour';
s3 := s1 + s2 ;
{ s3 vaut 'bonjour' }
```

42

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**
- **Fonctions**

a) Longueur d'une chaine

- En Pascal, la fonction **`length`** permet de connaitre la longueur d'une chaine.
- Exemple

```
s1:='salut';
s2:='bonjour';
```

- `length(s1)` vaut 5 et `length(s2)` vaut 7.

b) Position d'une sous-chaine dans une chaine

- Fonction **`pos`** :
- `pos(souschaine, chaine)`
- renvoie la position de la sous chaine dans la chaine.

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**
- **Fonctions**

c) Extraction d'une sous-chaine

- Fonction **`copy`** :

```
copy (source, debut, l)
```

- extrait la sous-chaine de la chaine source de longueur l et commençant a la position debut.

- Exemple

```
s:=copy('bonjour monsieur', 4, 4)
```

- `s` vaut alors `'jour'`

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**
- **Fonctions de codage / décodage des caractères**

a) Détermination du code d'un caractère

- La fonction **ord** renvoie le code ASCII d'un caractère donné.

- Exemple
- `ord('A')` vaut 65 et `ord('a')` vaut 97
- b) Détermination du caractère correspondant a un code ASCII
- La fonction **chr** renvoie le caractère correspondant a un code ASCII donné

- Exemple
- `chr(65)` vaut `'A'` et `chr(97)` vaut `'a'`

45

I. Notion de variable et constante

Représentation

- **CHAÎNE DE CARACTERE**
- **Fonctions de codage / décodage des caractères**

- Exemple
- On désire transformer une lettre minuscule en lettre majuscule. Soit c le caractère à transformer. On écrira alors :

```
c := chr ( ORD(c) - ord('a') + ord('A') );
```

- Explications :

Si c correspond a la lettre 'a', alors :

`ord(c) - ord('a') = ord('a') - ord('a') = 0`

donc

`ord(c) - ord('a') + ord('A') = ord('A') = 65`

et :

`chr (ord('A')) = chr(65) = 'A'`

Nous avons bien transforme 'a' en 'A'

46

I. Notion de variable et constante

- **Type scalaire : Le type énuméré**

- Définition
- Un **type énuméré** est une séquence ordonnée d'identificateurs.
- Syntaxe

```
type
identificateur = (id1, id2,..., idn) ;
```

- Exemple

```
type
couleur = (jaune, vert, rouge, bleu, marron );
semaine =(lundi,mardi,mercredi,jeudi,vendredi,
samedi,dimanche);
reponse = (oui, non, inconnu);
sexe = (masculin, feminin);
voyelle = (A, E, I, O, U);
```

47

I. Notion de variable et constante

- **Type scalaire : Le type énuméré**

- Attention, le mot réservé type ne doit être écrit qu'une seule fois.

- Remarque

- Deux types énumères différents ne peuvent contenir le même identificateur.

- Les ensembles énumérés sont donc disjoints.

- La séquence de valeurs étant ordonnée, le système connait les successeurs et prédécesseurs d'un élément :
  - mardi : prédécesseur de mercredi.
  - mercredi : successeur jeudi.

48

I. Notion de variable et constante

- **Type scalaire : Le type intervalle**

- Définition
- Un **type intervalle** est nécessairement un sous-type d'un type scalaire (standard ou énumère) déjà défini.
- Toutes les valeurs de l'intervalle sont autorisées.

- Syntaxe

```
type
identificateur = [borne inf] .. [borne sup]
```

- Exemple : Intervalle d'entiers

```
type
Decimal = 0 .. 9
Octal = 0 .. 7
Age = 0 .. 150
```

49

I. Notion de variable et constante

- **Type scalaire : Le type intervalle**

- Exemple : Intervalle de caractères

```
type
ABC = 'A' .. 'C' ;
Maj = 'A' .. 'Z' ;
```

- Exemple : Intervalle avec un type non-standard

```
type
Ouvrable = lundi .. vendredi ; {préalablement déclaré!}
WeekEnd = samedi .. dimanche ;
Lettres = 'A' .. 'Z' ;
```

50

I. Notion de variable et constante

- **Type scalaire : Le type intervalle**

- Remarque
- On ne peut pas définir un type intervalle a l'aide du type *«real»* (type non scalaire).
- L'ordre ascendant est requis : *«borne-inf»* doit être place avant *«borne-sup»* dans le type énuméré source.

- Exemple : déclarations incorrectes

```
type
Octal= 7 .. 0 ;
Ouvrable = vendredi .. lundi ;
```