Vade mecum pseudo-code V 1.0

Types

- Entier (int) : 3
- Réel (float) : 2.75
- Chaîne de caractère (string) : "hello!"
- Booléen (boolean) : true
- Tableau (array) : {500, 200, 100, 50, 20, 10, 5, 2, 1}

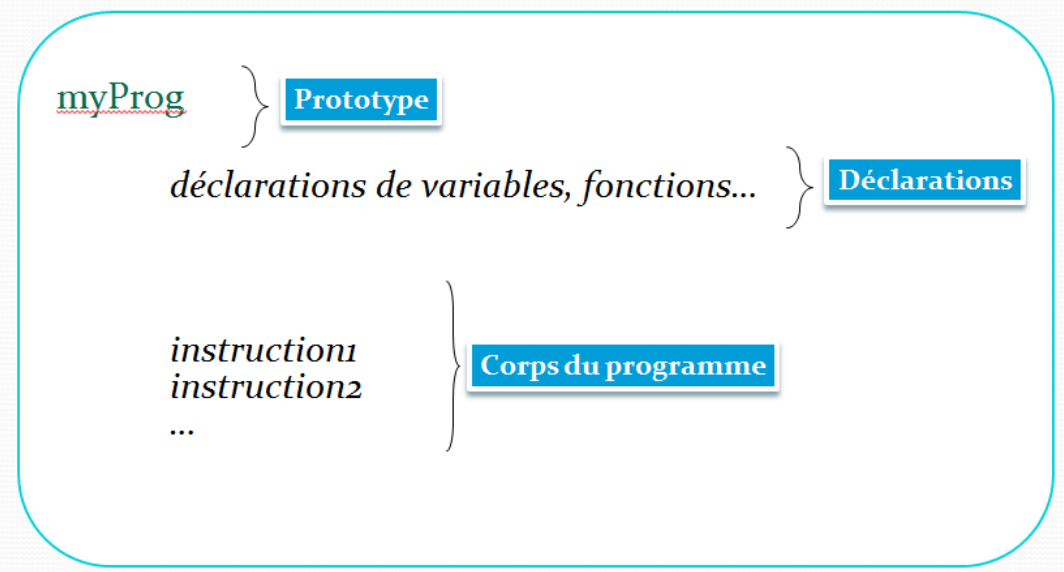
Opérateurs

- Numériques : +,/,*,-, **div** (division entière), **%** (modulo)
- Comparatifs : =, ≠, >, <, >, ≤, ≥
- Logiques: **or** (ou inclusif), **and**, **not**, **xor** (exclusif)
- De chaines : + (concaténation), **size** (taille)
- De tableaux : **size** (nombre d'éléments)

Entrées / Sorties

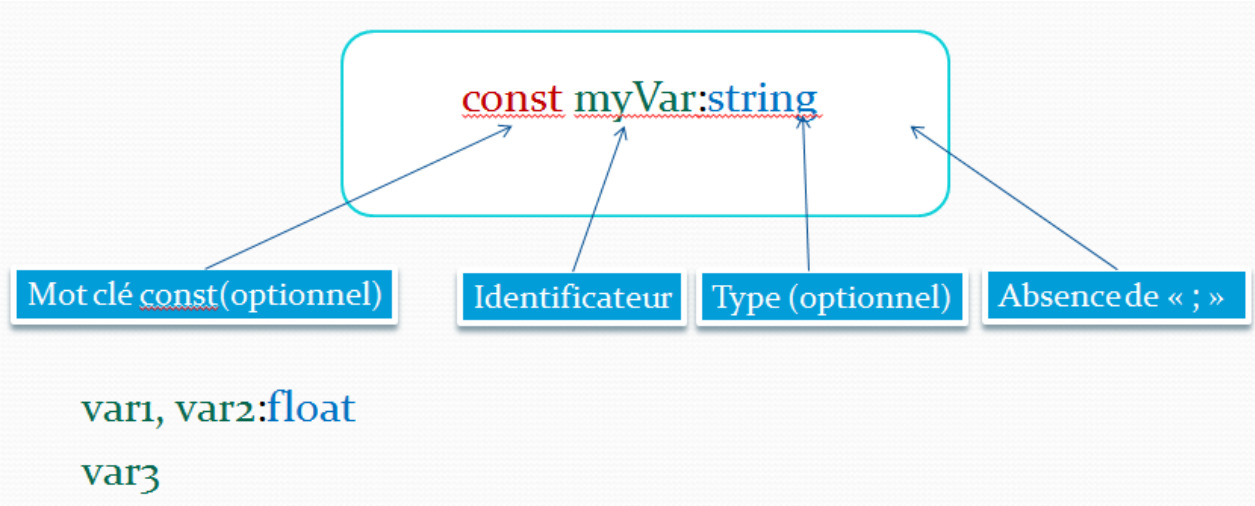
- Lire :
  - read var1
- Ecrire :
  - write var1, var2…
- Afficher :
  - print var1
  - print "result : ", var1
  - print "tab : ", {1, 4, -1}

Structure générale

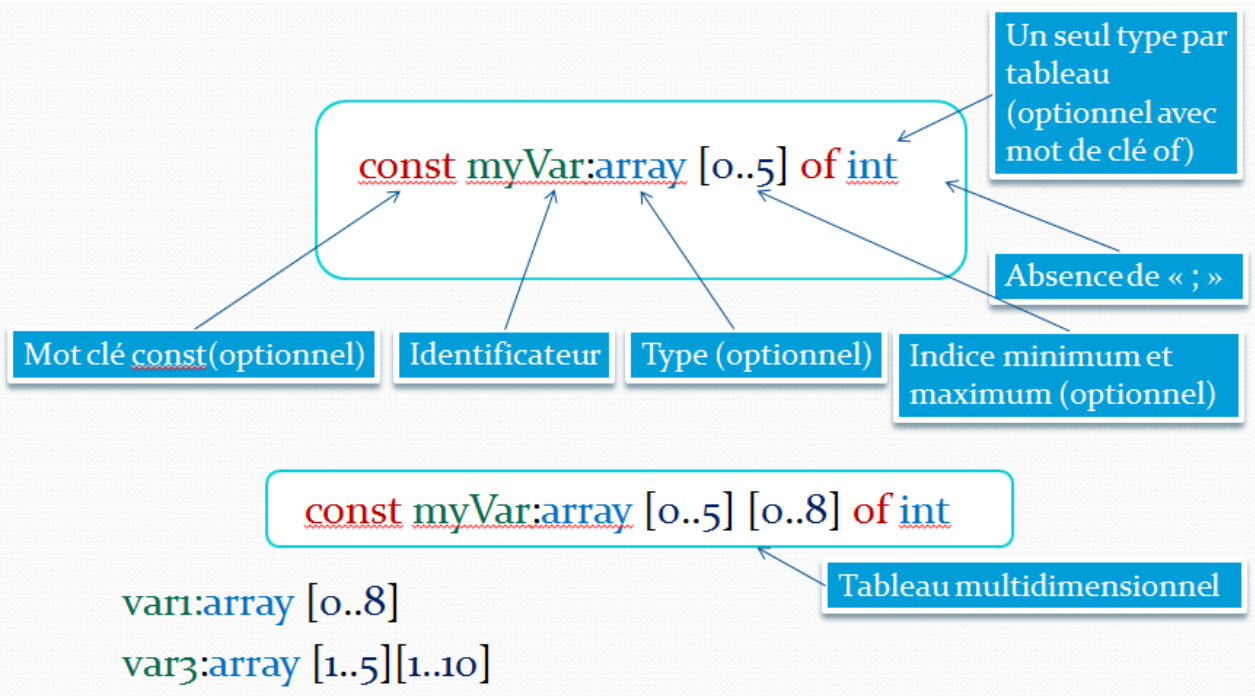


Déclaration

Simple

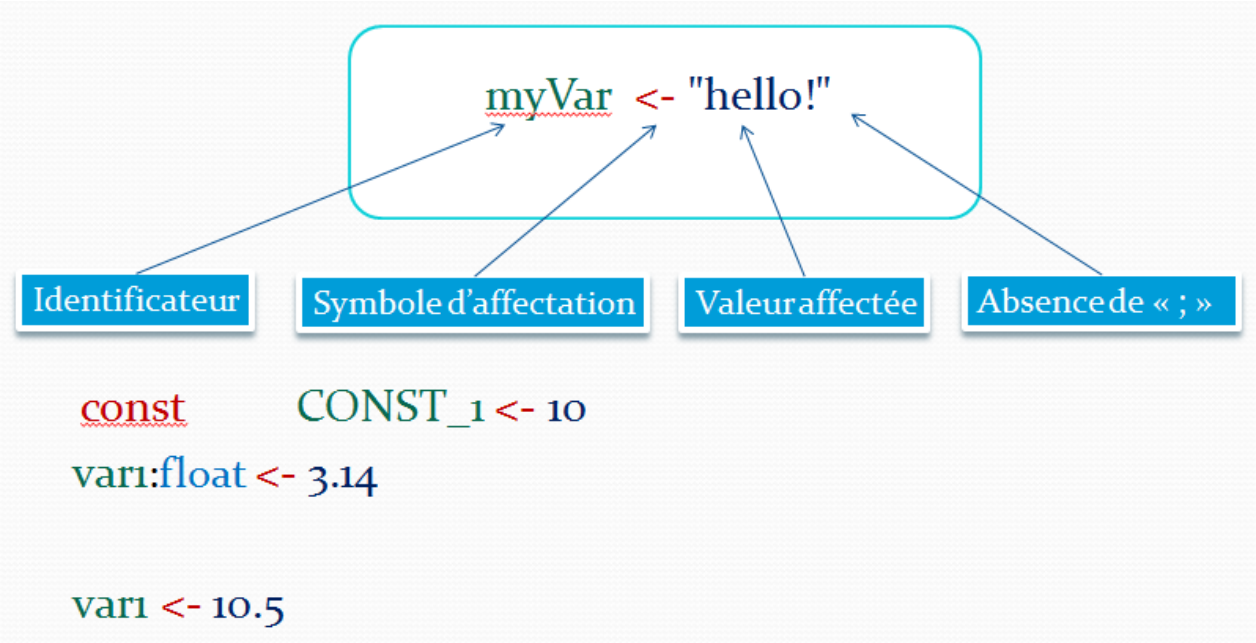


Tableau

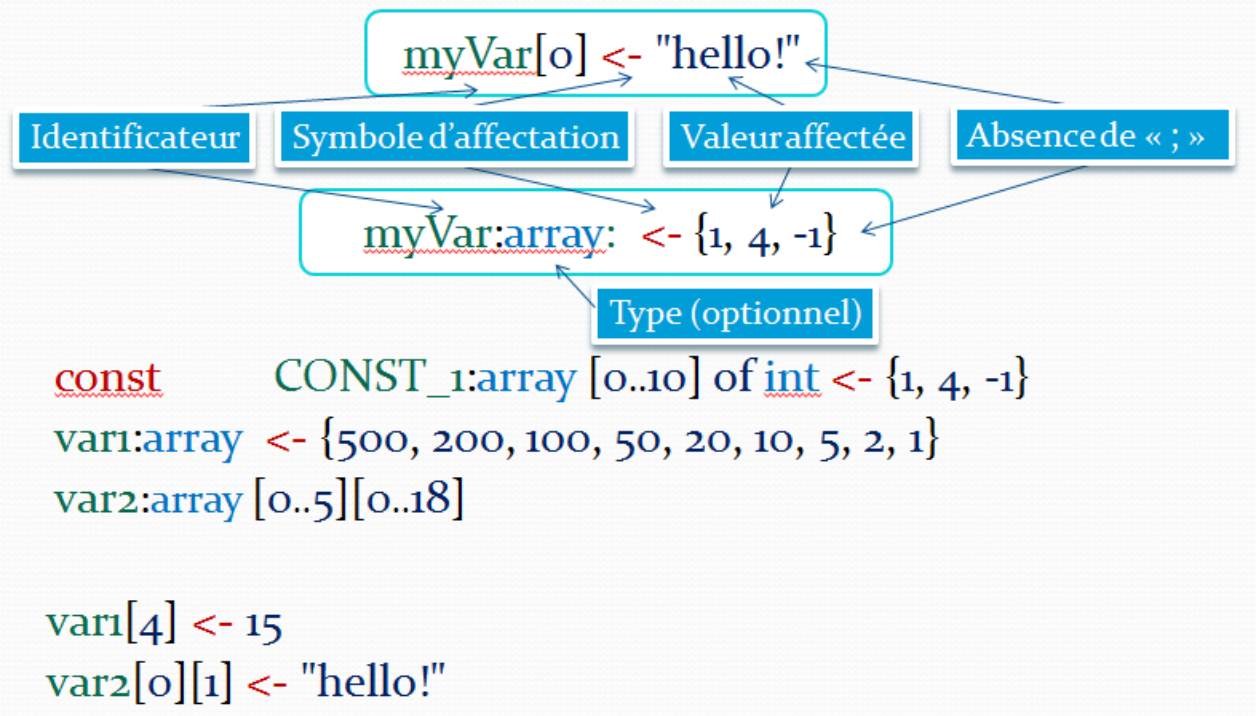


Affectation

Simple



Tableau



Structures conditionnelles

```
if condition
        instruction1
        instruction2
        ...
else
        instruction3
        instruction4
        ...
```

```
switch expression
        valeur1 : instruction1
        valeur2 : instruction2
                  instruction3
        valeur3,
        valeur4 : instruction4
        ...
else
        instruction5
        ...
```

Répétition (boucles)

De 0 à n fois

```
while condition do
        instruction1
        instruction2
        ...
```

De 1 à n fois

```
do
        instruction1
        instruction2
        ...
while condition
```

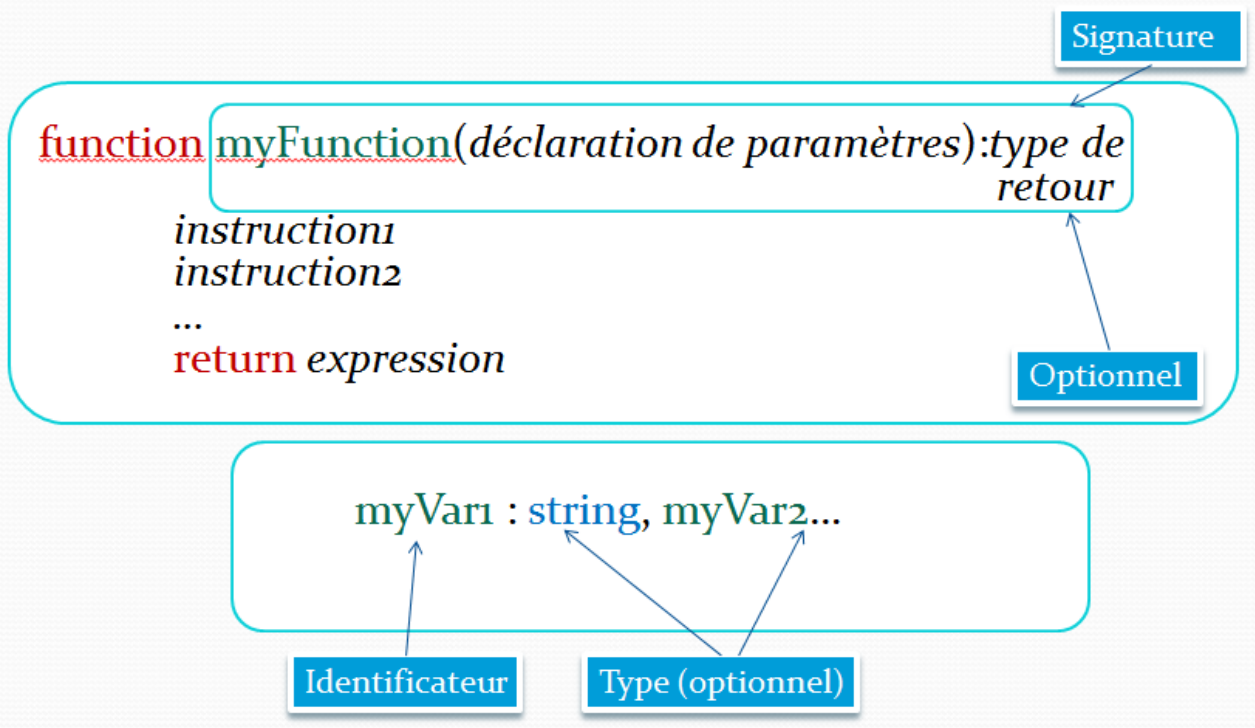
Avec un pas de 1

```
for x <- i to j do
        instruction1
        instruction2
        ...
```

Avec un pas de k
k entier

```
for x <- i to j with step k do
        instruction1
        instruction2
        ...
```

« Fonction » / paramètres



Bonne pratiques

- **Une seule page** : généralement 20 – 25 lignes
- Laisser des **lignes blanches** régulièrement
- Avoir une bonne **indentation** :
  - myProg
  - myVar
  - MY_CONST
- Choisir une **langue** : l'anglais
- Choisir une **convention** d'écriture
- **Factoriser** le code (modularité et règle D.R.Y)
- Eviter les zones troubles
- Eviter les cas particuliers
- Eviter les commentaires (qui sont de même type que les **commentaires C/C++**)
- **Pas de trivialité**. Eviter :
  - Les déclarations des variables locales (on laisse les initialisations)
  - Les Indices de début et de fin des boucles triviales
  - Les types de certaines variables ou paramètres
- **Pas de détail** sur les morceaux de code usuels :
  - Entrée et sorties simples
  - Structures ou méthodes courantes
  - Fonctions de comparaison standards