Exercices

Ces exercices vous donneront des exemples d'algorithmes répondant à une problématiques données. L'ensemble de ces algorithmes peut former une base de révision algorithmique car ce document couvre les éléments nécessaires pour construire la plupart des algorithmes procéduriaux existant.

Problèmes

Vous trouverez la liste des problèmes et une de leurs solutions algorithmiques ci-dessous.

1. Echanger le contenu de 2 variables

Problème : on souhaite échanger le contenu de `U` et de `V`.

```
A <- 25
B <- 2
```

Algorithme :

```
C <- U
U <- V
V <- C

Ne pas être mis dans une fonction
```

Python :

```
C = U
U = V
V = C
```

2. Echanger le contenu de 3 variables

Problème : on souhaite échanger le contenu de `U`, de `V` et de `Z`. C'est-à-dire de mettre la donnée de `U` dans `V`, `V` dans `Z` et celle de `Z` dans `U`.

Algorithme :

```
C <- Z
Z <- V
V <- U
U <- C

Ne pas être mis dans une fonction
```

Python :

```
C = Z
Z = V
V = U
U = C
```

3. Vérifier si un utilisateur est majeur

Problème : Vérifier si un utilisateur est majeur grâce à son âge. Mettre une variable `resultat` à `Vrai` si cela est le cas, sinon mettez la à `Faux`.

```
age <- 25
```

```
@Fonction estMajeur @Entrée age @Sortie boolean
    @DebutBloc
    @Retourner age >= 18
    @FinBloc

resultat <- @Appeler estMajeur(age)
```

Python :

```
def estMajeur(age):
    return age >= 18

resultat = estMajeur(age)
```

Algorithme :

```
resultat <- Faux
@Si age >= 18
    @DebutBloc
    resultat <- Vrai
    @FinBloc
@Sinon
    @DebutBloc
    resultat <- Faux
    @FinBloc
```

Python :

```
resultat = False
if age >= 18:
    resultat = True
else:
    resultat = False
```

ou

```
resultat <- Faux
@Si age >= 18
    @DebutBloc
    resultat <- Vrai
    @FinBloc
```

Python :

```
resultat = False
if age >= 18:
    resultat = True
```

4. Vérifier si un utilisateur est mineur

Problème : Vérifier si un utilisateur est mineur grâce à son âge. Mettre une variable `resultat` à `Vrai` si cela est le cas, sinon mettez la à `Faux`.

```
age <- 12
```

Algorithme :

```
@Fonction estMineur @Entrées baba @Sortie boolean
    @DebutBloc
    @Retourner baba < 18
    @FinBloc

resultat <- @Appeler estMineur(age)
```

Python :

```
def estMineur(baba):
    return baba < 18

resultat = estMineur(age)
```

5. Echanger les 2 premiers éléments d'un tableau

Problème : Echanger les 2 premiers éléments d'un tableau

```
tablo <- [23, 4, 2, 543, 34, ...]
```

Algorithme :

```
@Fonction echanger @Entrées tab
    @DebutBloc
    C <- tab[0]
    tab[0] <- tab[1]
    tab[1] <- C
    @FinBloc

@Appeler echanger(tablo)
```

Python :

```
def echanger(tab):
    C = tab[0]
    tab[0] = tab[1]
    tab[1] = C

echanger(tablo)
```

6. Compter le nombre d'éléments dans un tableau

Problème : Compter le nombre d'éléments dans un tableau. Mettre le compte dans `resultat`.

```
tab <- [23, 4, 2, 543, 34, ...]
```

Algorithme :

```
resultat <- 0
@PourChaque element @Dans tab
    @DebutBloc
    resultat <- resultat + 1
    @FinBloc
```

Python :

```
resultat = 0
for element in tab:
    resultat = resultat + 1
```

7. Faire la somme des éléments d'un tableau

Problème : Faire la somme des éléments d'un tableau

```
tab <- [23, 4, 2, 543, 34, ...]
```

Algorithme :

```
somme <- 0
@PourChaque element @Dans tab
    @DebutBloc
    somme <- somme + element
    @FinBloc
```

Python :

```
somme = 0
for element in tab:
    somme = somme + element
```

8. Trouver l'élément maximum d'un tableau

Problème : Trouver l'élément maximum d'un tableau comprenant des nombres de 0 à 1000. Mettre le numbre maximum dans `resultat`.

```
tab <- [...]
```

Algorithme :

```
resultat <- tab[0]
@PourChaque element @Dans tab
    @DebutBloc
    @Si element > resultat
        @DebutBloc
        resultat <- element
        @FinBloc
    @FinBloc
```

Python :

```
resultat = tab[0]
for element in tab:
    if element > resultat:
        resultat = element
```

9. Trouver l'élément minimum d'un tableau

Problème : Trouver l'élément minimum d'un tableau comprenant des nombres de 0 à 1000

```
tab <- [2, ...]
```

Algorithme :

```
resultat <- tab[0]
@PourChaque element @Dans tab
    @DebutBloc
    @Si element < resultat
        @DebutBloc
        resultat <- element
        @FinBloc
    @FinBloc
```

Python :

```
resultat = tab[0]
for element in tab:
    if element < resultat:
        resultat = element
```

10. Trouver le premier élément supérieur à 500

Problème : Trouver le premier élément supérieur à 500. S'il n'y en a pas, le resultat doit être 0

```
tab <- [23, 4, 2, 543, 34, ...]
```

Algorithme :

```
@Fonction trouverPremierSup500 @Entrée tab
@DebutBloc
    @PourChaque element @Dans tab
        @DebutBloc
        @Si element > 500
            @DebutBloc
            @Retourner element
            @FinBloc
        @FinBloc
@FinBloc
```

Python

```
def trouverPremierSup500(tab):
    for element in tab:
        if element > 500:
            return element
```

11. Copier un tableau

Problème : Copier le tableau `tab` dans un autre tableau `autre`

```
tab <- [23, 4, 2, 543, 34, ...]
```

Algorithme :

```
autre <- []
@PourChaque element @Dans tab
    @DebutBloc
    autre.ajouter element
    @FinBloc
```

Python :

```
autre = []
for element in tab:
    autre.append(element)
```

12. Copier les premiers éléments d'un tableau dont la somme fait au moins 500

Problème : Copier les premiers éléments d'un tableau dont la somme fait au moins 500

```
tab <- [23, 4, 2, 543, 34, ...]
```

Algorithme :

```
@Fonction copierElementPourSomme500 @Entrée tab
@DebutBloc
    autre_tableau <- []
    somme <- 0
    @PourChaque prix_en_cours @Dans tab
        @DebutBloc
        somme <- somme + prix_en_cours
        Ajouter prix_en_cours Dans autre_tableau
        @Si somme >= 500
            @DebutBloc
            @Stop
            @FinBloc
        @FinBloc
    @Si somme < 500
        @DebutBloc
        autre_tableau <- []
        @FinBloc
    @Retouner autre_tableau
@FinBloc
```

Python :

```
def copierElementPourSomme500(tab):
    autre_tableau = []
    somme = 0
    for prix_en_cours in tab:
        somme = somme + prix_en_cours
        autre_tableau.append(prix_en_cours)
        if somme >= 500:
            break
    if somme < 500:
        autre_tableau = []
    return autre_tableau
```