04/01/2021

Algorithme

Exercices

Lesur Rudy

Préambule

Afin de réaliser ces **exercices** de mise en œuvre, vous devez parfaitement connaître les bases de l'algorithmie :

- La notion de **variable**.
- Les **structures de contrôle**.
- Les **structures itératives**.
- Les **tableaux**.
- Les **procédures** et **fonctions**.
- Les **paramètres** et **retour** de fonctions.

Objectifs

A l'issue de la réalisation de ces exercices, vous saurez mettre en œuvre les solutions de problèmes informatiques exprimés en algorithmie et **pseudocode** et aurez assimilé tous les **fondamentaux** de la **programmation procédurale**.

Méthodologie

Dans un premier temps, à travers ces exercices, vous écrirez d'abord les solutions exprimées dans le **langage algorithmique** ou **pseudocode** puis les ferez valider par votre formateur.

Pour chaque solution, fournissez un jeu d'essais et vérifiez le bon comportement de vos algorithmes.

Exercice N°1

- Ecrivez un algorithme en **pseudocode** permettant le calcul de la somme des *N* premiers nombres entiers.
- La variable *N* sera saisie par l'utilisateur grâce à une fonction *lireEntier()* qui permet de :
    - saisir un entier au clavier et …
    - renvoyer cette valeur en retour.

- Elle s'utilise donc de la manière suivante :

```
Variables
    N : entier // Déclaration de la variable N de type entier

N :=lireEntier() // Lecture d'un entier et affectation de la variable N
…..
```

- Par exemple : A l'issue de la saisie de la valeur 6 et après validation, la variable N vaut 6.
- Vous mettrez en œuvre cet algorithme en générant **3 solutions** en **pseudocode** : une pour chaque structure itérative, **TANT QUE, REPETER** et **POUR**.
- L'affichage du résultat se fera ainsi pour chaque solution :

```
écrire (« Le résultat avec tantque est : », resultat);
```

Exercice N°2

- Ecrivez un algorithme en **pseudocode** permettant le calcul de la factorielle d'un **entier** *X*.
- La variable *X* sera saisie selon le même procédé que ci-dessus.
- Vous écrirez cet algorithme avec les 2 structures itératives **TANT QUE, REPETER**
- Les affichages seront réalisés avec la méthode que ci-dessus.

```
écrire (« La factorielle de », X, « avec tantque est : », resultat)
```

Exercice N°3

- Ecrivez un algorithme en **pseudocode** permettant la résolution d'une équation du second degré $ax^2 + bx + c = 0$.
- On saisira les **entiers** *a*, *b* et *c*. avec la fonction *lireEntier().*
- La(les) solution(s) seront de type **réel**.
- Fournissez une seule solution avec les **structures itératives** que vous jugerez pertinentes.

Aide :

Il faut d'abord vérifier que a est différent de 0 sinon erreur.
Ensuite il faut calculer le discriminant que je vais appeler D. Ce discriminant est égal à :

$b^2 - 4ac$.

Si D = 0, il y a une seule solution : le réel : $-\frac{b}{2a}$

Si D > 0, il y a 2 solutions réelles distinctes : $\frac{-b-\sqrt{D}}{2a}$ et $\frac{-b+\sqrt{D}}{2a}$

Si D < 0, pas de solution réelle.

Exercice N°4

- Ecrivez un algorithme en **pseudocode** permettant le calcul d'un nombre *X* élevé à la puissance *Y*.
- Les **entiers positifs** *X* et *Y* seront saisis toujours avec la fonction *lireEntier().*
- Ecrivez une fonction *entier puissance(entier x, entier y)* qui admet deux paramètres *x* et *y* de type **entier** et qui renvoie le résultat de *x* élevé à la puissance *y*.
- La fonction *puissance* sera appelée au sein du programme principal. La valeur retournée sera ensuite affichée pour produire le résultat.

Exercice N°5

- Ecrivez un algorithme en **pseudocode** permettant la recherche **dichotomique** d'une variable entière *X* dans un tableau d'entiers *TabEntiers* de taille TAILLE_MAX **déjà trié.**
- Les principes de la recherche **dichotomique** consistent à diviser par deux l'espace de recherche par deux, tant que l'on n'a pas trouvé *X*, et à y rechercher la valeur souhaitée, après avoir changé les bornes de l'espace de recherche.
- Naturellement, ce principe ne fonctionne que si le tableau est déjà trié.
- Demandez des informations complémentaires à votre formateur si besoin.
- Utilisez les structures **itératives** et de contrôle que vous jugerez nécessaires.

- Déclarez un tableau de 10 entiers. Rangez-y 10 valeurs négatives, positives ou nulles dans **l'ordre croissant**.
- Ecrivez une fonction *entier rechercherEntier (entier tab[] , entier x)* qui recherche la **position** de la variable *x* dans le tableau *tab* en mettant en œuvre la recherche dichotomique.
- Si *x* est trouvée dans *tab*, la fonction renvoie le rang du tableau auquel elle se trouve.
- Si *x* n'existe pas dans le tableau, la fonction renvoie -1 au programme appelant, indiquant ainsi l'absence de cette valeur.

-2	-1	0	13	24	37	44	56	99	117

Dans le tableau ci-contre, en recherchant la valeur 44, la fonction *rechercheEntier(…)* retourne la valeur 7 : c'est le rang dans le tableau où elle a été trouvée.

Le résultat produira l'affichage suivant :

```
44 se trouve au rang 7 dans le tableau
```

NB : On peut connaître la **taille** et donc le **nombre d'éléments d'un tableau** en utilisant la **notation pointée** suivi du mot-clé **taille.**

Par exemple la valeur de *tab.taille*, appliquée au tableau ci-dessous est **10**.

Exercice N°6

- Ecrivez un algorithme en pseudocode permettant de **trier un tableau d'entiers**.
- Au sein du module principal, un tableau de taille TAILLE sera déclaré et rempli avec des entiers négatifs, positifs ou nuls, sans ordre particulier et transmis dans une procédure *trierTableau.*
- *trierTableau* va donc recevoir en unique argument le tableau d'entiers à trier. On utilisera la **méthode de tri par permutation**.

Rappel : tri par permutation

On compare deux éléments (le 1er et le second) et on le permute pour avoir en premier le plus petit des deux.
Puis on compare le nouveau premier avec le 3 ème … ainsi de suite.

Quand on a comparé le 1er avec le dernier, on recommence MAIS cette fois en partant du second.
On compare donc le second avec le troisième et ainsi de suite.

Le tri est terminé quand on ne fait plus aucune permutation.