Secteur Tertiaire Informatique

Filière « Etude et développement »

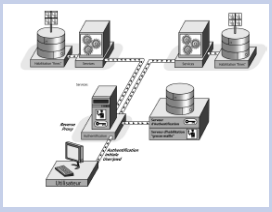
Ecrire un algorithme – partie 2

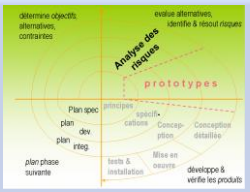
Programmation procédurale – série 2

Apprentissage

Mise en situation

Evaluation





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 aurez assimilé tous les **fondamentaux** de la **programmation procédurale**.

Méthodologie

Cette série d'exercices est la suite de « *Exercices programmation procedurale serie 1*-».

Elle vous permet de **revoir** et **compléter** les bases de la programmation procédurale que vous avez déjà étudiées.

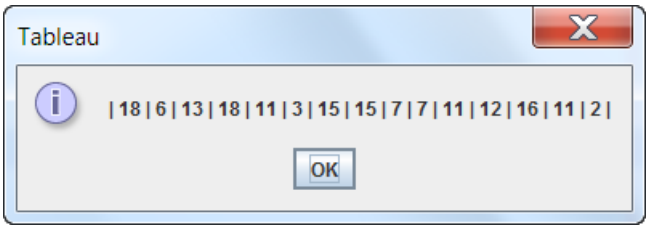
Vous écrirez les solutions exprimées dans le **langage algorithmique** puis coderez ces solutions dans le langage indiqué par votre formateur en utilisant votre EDI habituel.

Quand vous aurez terminé les exercices de ce support, vous transmettrez vos travaux à votre formateur.

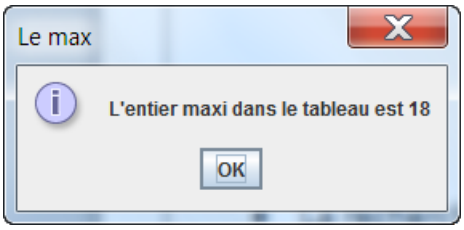
Les copies d'écran qui sont proposées sont issues de solutions écrites en **Java**.

Exercice N° 1

- Ecrivez un algorithme en pseudocode qui permet de remplir aléatoirement un tableau d'entiers de TAILLE éléments dans le module principal puis qui affiche la valeur entière maximum de ce tableau d'entiers.
- Par exemple, le tableau rempli :



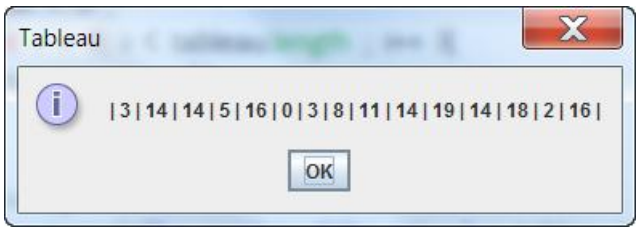
- La recherche de l'entier maximum présent dans le tableau sera confiée à une fonction *lireMaxTabEntiers* qui recevra en argument le tableau et qui renverra l'entier maxi lu dans ce tableau.



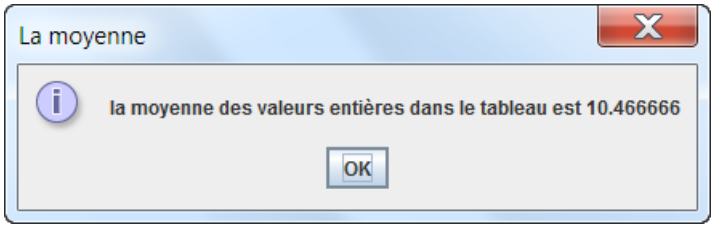
Codez votre pseudocode.

Exercice N°2

- Ecrivez un algorithme en pseudocode permettant de remplir un tableau d'entiers dans le module principal puis, grâce à la fonction *calculerMoyenne*, de calculer et afficher la moyenne des valeurs entières de ce tableau d'entiers transmis en argument.



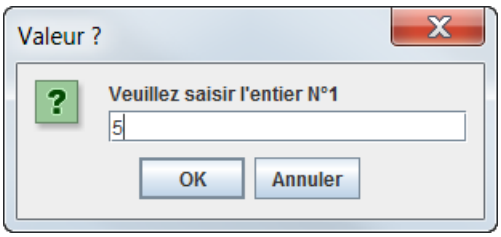
- La fonction *calculerMoyenne* renverra cette moyenne sous la forme d'un réel et sera affichée dans le module appelant :



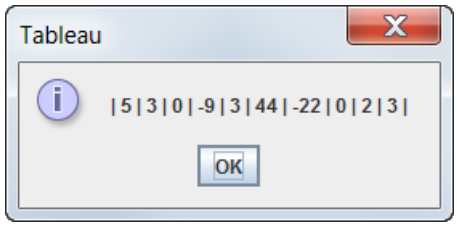
- Codez votre pseudocode.

Exercice N°3

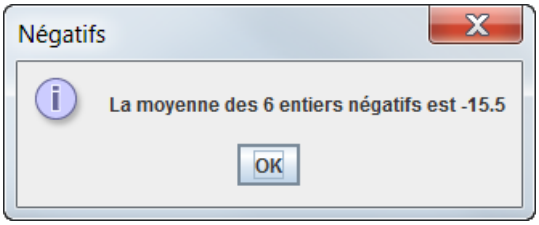
- Ecrivez un algorithme en pseudocode permettant dans le module principal la saisie de TAILLE_MAX valeurs entières négatives, positives ou nulles au sein d'une boucle.
- Par exemple :



- Rangez chacune de ces valeurs dans un tableau *tableauEntiers de TAILLE_MAX* valeurs.
- Affichez ce tableau par l'appel à la méthode *afficherTableauBoiteMessage*.



- Ecrivez une procédure *analyserTableau* qui calcule et affiche le nombre d'entiers positifs et leur moyenne, le nombre d'entiers négatifs et leur moyenne et enfin le nombre de valeurs nulles. Cette procédure reçoit évidemment le tableau d'entiers et ne renvoie rien.
- Par exemple, pour les valeurs négatives :



- Codez votre pseudocode.

Exercice N°4

- Ecrivez un algorithme en pseudocode permettant le rangement, dans un tableau «*TabMultiplication*», des tables de multiplication de 1*1 jusqu'à 9*9.
- Le nombre des colonnes (10) et le nombre des lignes (20) seront consignés dans deux constantes LIGNES et COLONNES.
- Le tableau, une fois rempli dans le module principal, devra permettre de générer, à la suite d'un dialogue, une réponse de la forme :

« Le produit de : » , X, « par : » , Y « est : », PRODUIT

- Les variables entières X et Y ayant fait, au préalable, l'objet d'une saisie interactive par :

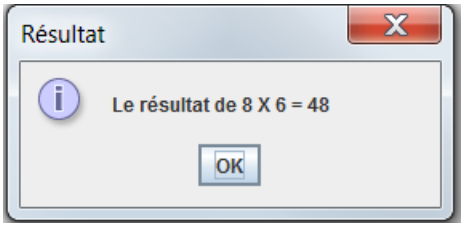
- Toute saisie incorrecte produira l'affichage suivant :

- Le tableau devra être rempli selon le format suivant :

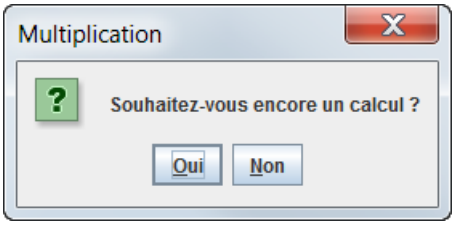
	1	2	…	8	9
1	1	2	…	8	9
2	2	4	…	16	18
…	…	…	…	…	…
8	8	16	…	64	72
9	9	18	…	72	81

- Vous utiliserez les structures de contrôle **itératives** les mieux adaptées.
- Naturellement, vous exploiterez le tableau pour extraire la valeur située à l'intersection des ligne/colonne correspondant aux opérandes saisies.

- La réponse sera de la forme :



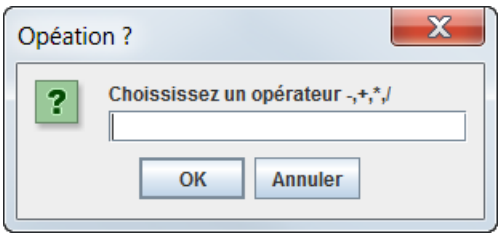
- Après un calcul et la production de son affichage, vous proposerez à l'utilisateur le dialogue suivant :



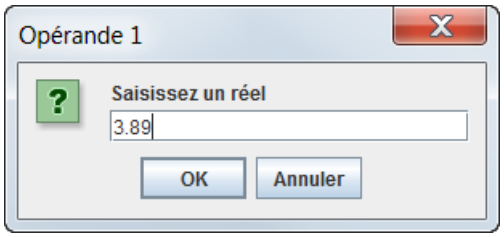
- Codez votre pseudocode.

Exercice N° 5

- Ecrivez un algorithme en pseudocode permettant de réaliser une calculatrice pour les 4 opérations arithmétiques suivantes : + , - , * , / .
- Dans le module principal, on proposera à l'utilisateur un menu l'invitant à saisir :
    1. Le caractère associé à l'opération choisie. (une des quatre opérations arithmétiques)



    2. Les 2 valeurs entières pour les lesquelles il souhaite réaliser l'opération. Cette dernière a fait l'objet d'une saisie dans l'étape précédente.



- Ecrivez une procédure (une fonction) '*calculer*' qui admet les trois paramètres suivants :

Reel calculer(**Char** operateur, **Reel** opérande1, **Reel** opérande2)

- On voit que la fonction *calculer(…)* reçoit les paramètres nécessaires pour effectuer une opération arithmétique :
    1. Le premier est le caractère correspondant à l'opération : ce sera '+' ou '-' ou '*' ou encore'/'.
    2. Le deuxième sera le premier opérande pour le calcul.
    3. Le troisième sera le second opérande pour le calcul.

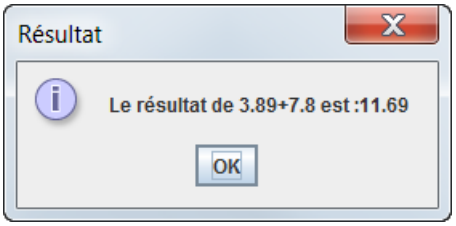
- La fonction *calculer()*, après avoir effectué le calcul, retournera celui-ci au module appelant. Ce résultat sera de type réel.

- Par exemple, en **Java,** la fonction aura donc la forme :

```
public static double calculer( char operateur,double var1, double var2)
```

- Vous mettrez en œuvre toutes les dialogues nécessaires, ainsi que les contrôles après saisies, requis pour une utilisation standard de la calculatrice.

- Pensez à vérifier et à contrôler toute tentative de division par zéro.

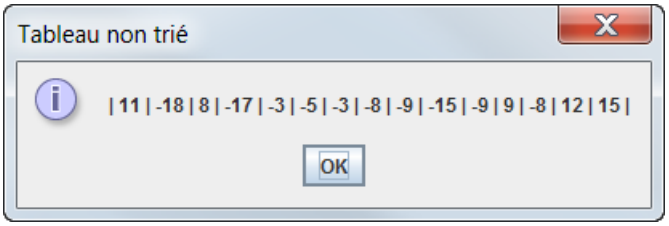


- Codez votre pseudocode.

- Vous améliorerez le code ultérieurement en le sécurisant par la **mise en œuvre des exceptions** visant à gérer toute tentative de division par zéro.

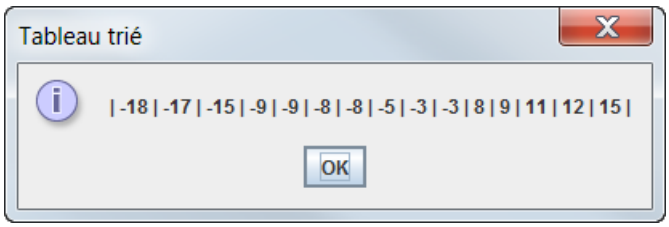
Exercice N°6

- Ecrivez un algorithme en pseudocode permettant de **trier un tableau d'entiers**.
- Au sein du module principal, ce tableau sera déclaré et rempli avec des entiers négatifs, positifs ou nuls, sans ordre particulier dans le module principal 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** (voir explication par votre formateur).

- Ecrivez le code de la fonction *trierTableau(…)* dans la classe *Utilitaires*, similairement aux solutions proposées précédentes.



- Ecrivez également une procédure permettant d'afficher dans une boîte de dialogue le tableau comme dans les exemples précédents.

- Vous utiliserez cette procédure pour afficher le tableau d'entiers non trié, puis ce même tableau trié après l'appel à *trierTableau()*.

- Lorsque vous coderez votre pseudocode, écrivez un module pour remplir aléatoirement le tableau d'entiers.

- Ajoutez un booléen à la méthode de tri pour trier selon un ordre croissant ou décroissant en fonction de la valeur de ce booléen.

Version	Date	Auteur(s)	Action(s)
1.0	05/11/14	M. Coulard	Création du document
1.1	03/12/14	C. Perrachon	Modification 1ère et dernière page
1.2	05/01/15	C. Perrachon	Modifications et commentaires en mode révision

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Michel Coulard – Formateur Evry
Chantal Perrachon – IF Neuilly sur Marne

Date de mise à jour : 05/01/15