Exercice Algo :

# Exercice 1 :

Ecrire un algorithme qui demande à l’utilisateur un nombre compris entre 1 et 3 jusqu’à ce que la réponse convienne.

ALGORITHME SaisirNombre

{Demander à l’utilisateur un nombre entre 1 et 3 jusqu’à que la réponse convienne}

Variables

nombre: entier {Déclaration de la variable nombre}

Début

Répéter

// Demander à l’utilisateur de saisir un nombre

afficher («Entrez un nombre entre 1 et 3 :»)

Saisir(nombre)

// Vérifier si le nombre est valide

si (nombre >= 1) et (nombre <= 3) alors

Afficher(«réponse valide»)

Sinon

Afficher(«réponse invalide»)

Fin si

Jusqu’à ce que (nombre >= 1) et (nombre <= 3)

Fin

# Exercice 2 :

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

ALGORITHME NombresSuivants

{Demander un nombre de départ et afficher les dix nombres suivants}

Constantes

(Limite : entier) 🡨 10

Variables

nombreDepart, i: entiers

Tableau

nombresSuivants: tableau d’entiers

Début

// Demander à l’utilisateur de saisir un nombre

afficher («Donnez-moi un nombre de départ:»)

Saisir(nombreDepart)

I 🡨 0 {initialisation de la variable i}

// Calculer et stocker les dix nombres suivants dans le tableau nombresSuivants

Tant que (i <= Limite) faire

nombreSuivants[i] 🡨 nombreDepart + i +1

i 🡨 i + 1

fin Tant que

// Afficher les nombres suivants

Afficher( nombresSuivants)

Fin

# EXERCICE 3 :

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) : Table de 7 : 7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 … 7 x 10 = 70

ALGORITHME TableMultiplication

{Générer une table de multiplication pour nombre donné}

Variables

nombreDepart, i: entiers

Tableau

tableMultiplication: tableau d’entiers

Chaine

Resultat: chaîne de caractères

début

// Demander à l’utilisateur de saisir un nombre

afficher («Donnez-moi un nombre:»)

Saisir(nombreDepart)

// Initialiser le tableau de la table de multiplication.

pour i <- 1 à 10 faire

tableMultiplication [i] <- nombreDepart \* i

fin pour

// Générer la chaîne de résultat

resultat 🡨 nombreDepart + «:»

pour i 🡨 1 à 10 faire

resultat 🡨 resultat + tableMultiplication [i]

si I < 10 alors

resultat 🡨 resultat + “,”

fin si

fin pour

// Afficher le résultat

Afficher(resultat)

fin

# Exercice 4 :

Lire la suite des prix (en euros entiers et terminée par zéro) des achats d’un client. Calculer la somme qu’il doit, lire la somme qu’il paye, et simuler la remise de la monnaie en affichant les textes "10 Euros", "5 Euros" et "1 Euro" autant de fois qu’il y a de coupures de chaque sorte à rendre.

Algorithme CaculerMonnaie

{Cet algorithme calcule la monnaie à rendre après achat}

Variables

prixAchetés : tableau de réels

montantPayé, coûtTotal, montantRendu :réels

coupures : tableau de réels

début

{Initialitatiin des valeurs}

prixAchetés 🡨 [20, 15, 7] {Liste des pris des arricles}

montantPayé 🡨 50 {Montant payé par le client}

coûTotal 🡨 0 {Initialisation du coût total à 0}

montantRendu 🡨 0 {Initialisation du montant rendu à 0}

coupures 🡨 [10, 5, 1] {Liste des coupures disponible}

{Calcul du coût total des articles achetés}

pour chaque prix dans prixAchetés faire

coûtToTal 🡨 coûtToTal + prix

fin pour

{Calcul du montant rendu}

montantRendu 🡨 montantPayé – coûtTotal

{Calcul des coupures à rendre}

pour chaque coupure dans coupures faire

tant que montantRendu >= coupure faire

Afficher(coupure + « Euros»

montRendu 🡨 montantRendu – coupure

fin tant que

fin pour

{Affichage du montant total rendu}

Afficher(«Montant total rendu : » + montantRendu + « Euros»)

fin

# Exercice 5 :

Écrire un algorithme qui permette de connaître ses chances de gagner au tiercé, quarté, quinté et autres impôts volontaires.

On demande à l’utilisateur le nombre de chevaux partants, et le nombre de chevaux joués. Les deux messages affichés devront être :

Dans l’ordre : une chance sur X de gagner

Dans le désordre : une chance sur Y de gagner

X et Y nous sont donnés par la formule suivante, si n est le nombre de chevaux partants et p le nombre de chevaux joués

(on rappelle que le signe ! signifie "factorielle", comme dans l'exercice 5.6 ci-dessus)

: X = n ! / (n - p) !

Y = n ! / (p ! \* (n – p) !)

NB : cet algorithme peut être écrit d’une manière simple, mais relativement peu performante. Ses performances peuvent être singulièrement augmentées par une petite astuce. Vous commencerez par écrire la manière la plus simple, puis vous identifierez le problème, et écrirez une deuxième version permettant de le résoudre.

Algorithme CalculChancesTierce

{Cet algorithme permet de connaître ses chances de gagner au tiercé , quarté et quinté}

Variables

n, p, x, y: entiers

Début

{Demander à l’utilisateur le nombre de chevaux partant et le nombre de chevaux joués

afficher(«Entrez le nombre de chevaux partants :»

saisir(n)

afficher(«Entrez le nombre de chevaux joués :»

saisir(p)

## Exercice 6 :

Un magasin de reprographie facture 2 euros les dix premières photocopies, 1.50 euros les vingt suivantes et 1 euros au-delà. Ecrivez un algorithme qui demande à l’utilisateur le nombre de photocopies effectuées puis affiche le montant correspondant.