Exercice 2.1.3 :

L’utilisateur peut saisir plusieurs valeurs à convertir.

Il saisit une valeur à convertir avec son unité de mesure (km ou mi).

Si aucune unité de mesure n’est indiquée, le programme considère la valeur en kilomètres.

A l’appui de la touche « Entrée », la valeur saisie est enregistrée dans un tableau en mémoire.

L’utilisateur peut enregistrer autant de valeurs qu’il souhaite.

Si l’utilisateur saisit la commande « go », le programme convertit toutes les valeurs mémorisées et affiche le résultat

de chaque conversion.

La commande « quit » permet de quitter le programme. Tant que cette commande n’est pas saisie, l’utilisateur peut

continuer à faire des conversions.

// Exercice 2\_1\_3

Variables :

string : userInput, unit

double : distance

string[] : userInputs

const string : go, quit

int : index

Code :

go <-- "go"

quit <-- "quit"

index <-- 0

DO

WRITE "Saisissez une distance avec son unité (\"km\" ou \"mi\") ou lancez la conversion avec \"go\" ou quitter le programme avec \"quit\""

READ userInput

IF (userInput == quit)

QuitterProgramme()

END IF

IF (userInput == go)

// Lancer la conversion de toutes les saisies

ConvertAllPreviousInputs(userInputs)

ELSE

// Vérifier la saisie (nombre, unité?)

userInput = CheckUserInput(userInput)

// Ajouter saisie dans le tableau userInputs

index <-- AddUserInputToInputs(userInput, userInputs, index)

END IF

WHILE(true)

FUNCTION:

void QuitterProgramme()

Variables :

Code :

WRITE "Bye bye"

Environnement.Exit(0)

void ConvertAllPreviousInputs(string[] inputs)

Variables :

const string : miles

string[] : input

double resultConversion

int : index

Code :

miles <-- "mi"

FOR index FROM 0 TO inputs.Length

// mettre dans input la distance et l'unité séparés en 2 éléments de tableau (input[0] distance et input[1] unité

input <-- inputs[index].Split(' ')

IF (input[0] est vide)

quitter la boucle

ELSE

IF (input[1] == miles) THEN

resultConversion < -- convertir de miles à kilometres

WRITE $"{input[0]} miles = {resultConversion} kilomètres"

ELSE

resultConversion < -- convertir de kilomètres à miles

WRITE $"{input[0]} kilomètres = {resultConversion} miles"

END IF

END IF

END FOR

Vider inputs

string CheckUserInput(string userInput)

Variables :

string[] : splitInput

const string : kilometers, emptyString

Codes :

miles <-- " km"

splitInput <-- userInput.Split(' ')

emptyString <-- ""

// Si l'entrée de l'utilisateur n'a pas d'unité, on lui ajoute l'unité kilomètre et on la renvoie, sinon on renvoie l'entrée de l'utilisateur directement

IF (splitInput .Length = 1)

return (userInput + kilometers)

ELSE

return userInput

END IF

IF (splitInput[0] n'est pas un nombre)

return emptyString

END IF

int AddUserInputToInputs(string userInput, string[] inputs, int index)

Variables :

Code :

IF (userInput n'est pas vide)

inputs[index] <-- userInput

index++

IF (index == inputs.Length)

index <-- 0

END IF

END IF

Return index