



Centre national de télé-enseignement de Madagascar

CNTEMAD

Apprendre et réussir en toute liberté

www.cntemad.mg

cntemad@cntemad.mg
22 600 57

**LICENCE 1
EN INFORMATIQUE**

**MODULE N°03
TOME 2**

**ALGORITHMIQUE
ET
LANGAGES 1**



TABLE DES MATIERES

1. Conventions syntaxiques

1.1. Syntaxe générale d'un programme en VB.NET	1
1.1.1. Les modules	1
1.2. Les variables	2
1.2.1. La déclaration des variables	2
1.2.2. Les types élémentaires de variable	2
1.2.3. Initialisation des variables	3
1.2.4. Conversion de type de variable	3
1.2.5. Affectation de valeur	4

2. La console

2.1. Affichages	6
2.2. Saisies	6
2.3. Exemple	7
2.4. Les opérateurs et expressions	7
2.4.1. Opérateurs et expressions arithmétiques	7
2.4.2. Opérateurs relationnels	8
2.4.3. Opérateurs et expressions logiques	8

3. La structure conditionnelle

3.1. La structure alternative	9
3.2. La structure de choix multiple	12

4. Structures répétitives

4.1. La structure répétitive (Tant que ... Faire ...)	15
4.2. La structure répétitive (Répéter ... Jusqu'à ...)	17
4.3. La structure répétitive For	19

5. Les tableaux

5.1. Syntaxes	20
5.2. Les tableaux à plusieurs dimensions	20
5.3. Initialisation pendant la déclaration	21
5.4. Redimensionnement	22
5.5. Autres méthodes	23

6. Programmation modulaire

6.1.	Syntaxe	24
6.2.	Passage par adresse	24
6.3.	Passage par valeur	25
6.4.	Visibilité d'un identificateur	26
6.5.	Les fonctions définies par l'utilisateur	26

7. Série d'exercices n° 1

7.1.	Enoncés	27
7.1.1.	Instructions d'affectation et d'entrée-sortie sourie	27
7.1.2.	Instructions conditionnelles	27
7.1.3.	Instructions répétitives	27
7.1.4.	Exercice récapitulatif avec structures alternatives et répétitives	28
7.2.	Corrigés	28
7.2.1.	Programme permettant de calculer la circonference et la surface d'un cercle.....	28
7.2.2.	Programme permettant de calculer le montant TTC	29
7.2.3.	Programme permettant de calculer la somme des n premiers nombres entiers	29
7.2.4.	Programme permettant de calculer le quotient et le reste de la division de 2 entiers ..	30
7.2.5.	Programme permettant de calculer la somme des n premiers nombres entiers pairs ..	30
7.2.6.	Programme permettant de calculer la somme et la moyenne de n nombres ..	31
7.2.7.	Programme permettant d'afficher la table de multiplication d'un nombre ..	31
7.2.8.	Programme de calcul de la factorielle de n ..	32
7.2.9.	Programme permettant d'identifier les nombres parfaits ..	32

8. Série d'exercices n° 2

8.1.	Recherche d'un élément dans un tableau	33
8.1.1.	Recherche à l'aide de la position relative de l'élément	33
8.1.2.	Recherche à l'aide de la valeur de l'élément	34
8.1.3.	Recherche de l'élément le plus grand ..	35

9. Série d'exercices n° 3

9.1.	Exercices sur les tableaux à une dimension	36
9.1.1.	Exercice 1	36
9.1.2.	Exercice 2	37
9.1.3.	Exercice 3	39
9.2.	Exercices sur les tableaux à deux dimensions	41
9.3.	Exercices sur les procédures et fonctions	42

1. Conventions syntaxiques

1.1. Syntaxe générale d'un programme en VB.NET

1.1.1. Les modules

En VB.NET les programmes sont contenus dans des modules.

Les modules sont des feuilles (ou formulaires) qui contiennent plusieurs procédures. Un module commence toujours par le mot clé **Module** et finit par **End Module**.

En Visual Basic.NET, il existe deux sortes de procédures :

- Les **sub**
- Les **function**.

Notes :

- Les **Sub** commencent par le mot clé **Sub** puis finissent par **End Sub** et ne retournent aucune valeur
- Les fonctions débutent par **Function** puis finissent par **End Function** et retournent une valeur précise.
- Les lignes commentaires débutent par une apostrophe.

'Exemple de module

Module NomDeMonModule

Sub Test1()

 'Procédure 1

End Sub

Function Test_1()

 'Fonction 1

End Function

Function Test_2()

 'Fonction 2

End Function

Sub Test2()

 'Procédure 2

End Sub

End Module

1.2. Les variables

1.2.1. La déclaration des variables

On utilise le mot clé "Dim" pour la déclarer et "new" pour instancier. De plus, il vaut lui assigner un type.

Voici la syntaxe :

Dim <nom de la variable> As <Type>

1.2.2. Les types élémentaires de variable

Types élémentaires			
Boolean	2 octets	True (0) ou False (1)	Pour les conditions
Byte	1 octet	De 0 à 255	Pour les fichiers
Char	2 octets	De 0 à 65 535	Pour les caractères alphanumériques
Date	8 octets	Dates entre le 1 Janvier 0001 et le 31 Décembre 9999	Pour les dates
Decimal	16 octets	+ / - 79 228 162 514 264 337 593 543 950 335 sans séparateur décimal sinon + / - 7.9 228 162# avec 28 chiffres après le séparateur décimal	Pour les nombres décimaux
Double	8 octets	- 1,79769313486231E+308 à - 4.94065645841247E-324 (nombres négatifs) et 4.94065645841247E-324 à 1,79769313486231E+308 (nombres positifs)	Pour les grands nombres à virgules (avec double précision)
Integer	4 octets	De - 2 147 483 648 à 2 147 483 647	Pour les nombres entiers
Long	8 octets	De - 9 223 372 036 854 775 808 à 9 223 372 036 854 775 807	Pour les entiers longs
Short	2 octets	De - 32 768 à 32 767	Pour les entiers courts
Single	4 octets	De - 3.402823E-45 à - 1.401298E+38	Pour les grands nombres à virgules (avec simple précision)
String	Variable	0 à 2 milliards de caractères Unicode	Pour les chaîne de caractères

'Exemples de déclarations de variables

Dim Ma_Variable As String

Dim Nombre As Integer

Dim Compteur As Integer

'Instancier des variables

Dim Msn As New Msn_Protocol 'Msn_Protocol est une class

Dim Test As New Class

1.2.3. Initialisation des variables

Avec VB.NET, on peut initialiser (donner une valeur initiale) une variable dès sa déclaration.

'Exemple avec les variables précédentes

Dim Ma_Variable As String = " toto "

Dim Nombre As Integer = 45

Dim Compteur As Integer = 1000

Une variable peut aussi être initialisée par une fonction :

'Exemple de variable instanciée par une fonction

Dim Pi As Decimal = Fonction_pour_calculer_pi()

1.2.4. Conversion de type de variable

Dans tout langage de programmation, il peut être utile voire même nécessaire de convertir des variables d'un type dans un autre type. Cela permet d'éviter la création de nouvelles variables qui alourdiraient le programme.

CBool	Boolean
CByte	Byte
CChar	Char
CDate	Date
CDbl	Double
CDec	Decimal
CInt	Integer
CLng	Long
CObj	Object
CShort	Short
CSng	Single
CStr	String

'Exemples

Dim Pi As Decimal = 3.14 'crée la variable Pi qui vaut 3.14

Dim a As Integer = 15 'crée la variable a qui vaut 15

Dim Pi_Entier As Integer = CInt(Pi) 'retournera 3

Dim a_caractère As String = CStr(a) 'retourner " 15 " 'en chaîne de caractère

1.2.5. Affectation de valeur

<identificateur> = <expression>

Note : En VB.NET le signe « = » représente à la fois l'affectation de valeur et l'opérateur de comparaison ! ?

Exemples :

Programme calculant le périmètre et la surface d'un quadrilatère :

```
Module Module1
```

```
Sub Main()
```

```
    Dim Longeur As Double
```

```
    Dim Largeur As Double
```

```
    Dim Perimetre As Double
```

```
    Dim Surface As Double
```

```
    Longeur = 28
```

```
    Largeur = 24
```

```
    Perimetre = (Longeur + Largeur) * 2
```

```
    Surface = Longeur * Largeur
```

```
End Sub
```

```
End Module
```

Notes :

- Par convention, on appelle Sub Main() la partie principale du programme ;
- On peut fusionner les déclarations avec les affectations de valeur.

```
Module Module1
```

```
Sub Main()
```

```
    Dim Longeur As Double = 28
```

```
    Dim Largeur As Double = 24
```

```
    Dim Perimetre As Double = (Longeur + Largeur) * 2
```

```
    Dim Surface As Double = Longeur * Largeur
```

```
End Sub
```

```
End Module
```

2. La console

La console est une fenêtre du type DOS. C'est assez archaïque comme interface mais c'est la définition de la console.

Dans une console, on peut afficher uniquement du texte.

Pour créer une application console dans Visual Studio.NET, aller dans :

Fichier/Nouveau Projet/Application Console

Avec SharpDevelop, il faut aller dans :

Fichier/Nouveau/Solution/VBNet/Application Windows/Application Console

Ensuite, il faut importer l'objet System.Console avec l'instruction suivante :

Imports System.Console

2.1. Affichages

Write(<Expression chaîne de caractère>)

WriteLine(<Expression chaîne de caractère>)

2.2. Saisies

<Variable>=Read()

<Variable>=ReadLine()

2.3. Exemple

```

Imports System.Console
Module Quadrilatere
    Sub Main()
        Dim Longeur As Double
        Dim Largeur As Double
        Dim Perimetre As Double
        Dim Surface As Double
        Write("Veuillez indiquer la longueur: ")
        Longeur = ReadLine()
        Write("Veuillez indiquer la largeur: ")
        Largeur = ReadLine()
        Perimetre = (Longeur + Largeur) * 2
        Surface = Longeur * Largeur
        WriteLine("Le périmètre est de " & Perimetre)
        WriteLine("Sa surface est de " & Surface)
        Write("Press any key to continue . . .")
        ReadKey(True)
    End Sub
End Module

```

2.4. Les opérateurs et expressions

2.4.1. Opérateurs et expressions arithmétiques

Opérateur	Description
+	Additionne deux nombres
-	Soustrait deux nombres
*	Multiplie deux nombres
/	Divise deux nombres et retourne un nombre à virgule flottante (décimal)
\	Divise deux nombres et retourne un nombre entier
Mod (Modulo)	Divise deux nombres et retourne seulement le reste
^	Elève à la puissance un nombre
&	Additionne (concatène) deux chaînes

2.4.2. Opérateurs relationnels

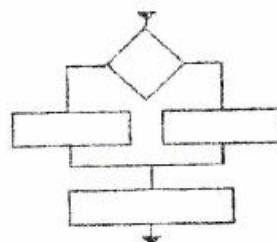
Opérateurs	Signification
>	Strictement supérieur à
<	Strictement inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à
<>	Different de
=	Egal à

2.4.3. Opérateurs et expressions logiques :

Opérateurs	Syntaxe	Signification
And (et)	Var_1 And Var_2	True si Var_1 et Var_2 sont vraies
Or (ou)	Var_1 Or Var_2	True si une des deux est vraie
Xor (ou exclusif)	Var_1 Xor Var_2	True si une et une seule est vraie
Not (non)	Not Var_1	True si Var est faux et vice versa

3. La structure conditionnelle

3.1. La structure alternative



```

If <condition> Then
    <Instructions si vraie>
Else
    <Instructions si fausse>
End If
  
```

Exemples :

Afficher le plus grand des deux nombres a et b :

```

Imports System.Console
Module Program
    Sub Main()
        Dim a As Double
        Dim b As Double
        Write("Entrez un nombre: ")
        a=ReadLine()
        Write("Entrez un nombre: ")
        b=ReadLine()
        Write("Le plus grand des deux est ")
        If a>= b Then
            WriteLine(a)
        Else
            WriteLine(b)
        End If
        ReadKey()
    End Sub
End Module
  
```

Que se passe-t-il lorsque $a=b$? Dans ce seul cas, la variable "a" est affichée. Ce cas n'a pas été explicitement prévu dans le programme. Voici par exemple comment y remédier :

```
Imports System.Console
```

```
Module Module1
```

```
Sub Main()
```

```
    Dim a As Double
```

```
    Dim b As Double
```

```
    Write("Entrez un nombre: ")
```

```
    a = ReadLine()
```

```
    Write("Entrez un nombre: ")
```

```
    b = ReadLine()
```

```
If a > b Then
```

```
    Write("Le plus grand des deux est " & a)
```

```
Else
```

```
If a = b Then
```

```
    WriteLine("Les nombres sont égaux")
```

```
Else
```

```
    WriteLine("Le plus grand des deux est :" & b)
```

```
End If
```

```
End If
```

```
.ReadKey()
```

```
End Sub
```

```
End Module
```

Ceci est une alternative imbriquée! On peut imbriquer autant d'alternatives que l'on souhaite.

Pour plus de lisibilité, on peut fusionner le premier Else avec le second If :

```
Imports System.Console
```

```
Module Module1
```

```
Sub Main()
```

```
    Dim a As Double
```

```
    Dim b As Double
```

```
    Write("Entrez un nombre: ")
```

```
    a = ReadLine()
```

```
    Write("Entrez un nombre: ")
```

```
    b = ReadLine()
```

```
If a > b Then
```

```
    Write("Le plus grand des deux est " & a)
```

```
ElseIf a = b Then
```

```
    WriteLine("Les nombres sont égaux")
```

```
Else
```

```
    WriteLine("Le plus grand des deux est :" & b)
```

```
End If
```

```
ReadKey()
```

```
End Sub
```

```
End Module
```

Il peut arriver aussi que la partie Else ne se présente pas. Cela veut dire que quand <expression_logique> est fausse, on ne fait rien.

Considérons l'algorithme suivant :

On se propose de récolter uniquement des nombres impairs. Pour cela, si le nombre est impair, on l'affiche, sinon, on ne fait rien.

```
Imports System.Console
```

```
Module Module1
```

```
Sub Main()
```

```
    Dim n As Integer
```

```
    Write("Veuillez saisir un nombre: ")
```

```
    n = ReadLine()
```

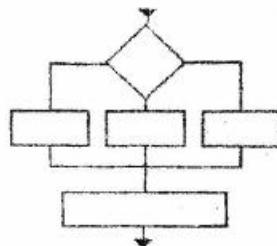
```
    If n Mod 2 = 1 Then WriteLine(n)
```

```
    ReadKey()
```

```
End Sub
```

```
End Module
```

3.2. La structure de choix multiple



```
Select Case <expression>
```

```
Case <valeur1>
```

<code effectué si expression=valeur1>

```
Case <valeur2>
```

<code effectué si expression=valeur2>

```
Case <valeur3>
```

<code effectué si expression=valeur3>

```
Case Else
```

<code effectué dans tous les autres cas>

```
End Select
```

Note : Attention si expression=valeur1 le code entre Case Valeur1 et Case valeur2 (et uniquement celui là) est effectué, puis l'exécution saute après End Select.

Exemple d'un code affichant le jour de la semaine :

Module Module1

```
Sub Main()
    Dim j As Integer
    Write("Veuillez saisir un nombre: ")
    j = ReadLine()
    Select Case j
        Case 1
            WriteLine("Lundi")
        Case 2
            WriteLine("Mardi")
        Case 3
            WriteLine("Mercredi")
        Case 4
            WriteLine("Jeudi")
        Case 5
            WriteLine("Vendredi")
        Case 6
            WriteLine("Samedi")
        Case 7
            WriteLine("Dimanche")
        Case Else
            WriteLine("Jour sans date")
    End Select
    ReadKey()
End Sub
```

End Module

Nous venons d'utiliser une expression simple après chaque Case mais on peut utiliser des expressions plus complexes.

Plusieurs clauses d'expression peuvent être séparées par des virgules.

Select Case N

Case 8,9,10

'Effectuer le code si N=8 ou N=9 ou N=10

Le mot clé To permet de définir les limites d'une plage de valeurs correspondantes pour N.

Select Case N

Case 8 To 20

'Effectuer le code si N est dans la plage 8 à 20

Le mot clé Is associé à un opérateur de comparaison (=, <>, <, <=, > ou >=) permet de spécifier une restriction sur les valeurs correspondantes de l'expression. Si le mot clé Is n'est pas indiqué, il est automatiquement inséré.

Select Case N

Case Is >= 5

'Effectuer le code si N supérieur ou égal à 5.

Vous pouvez utiliser plusieurs expressions ou plages dans chaque clause Case (séparées par des virgules). Par exemple, la ligne suivante est valide :

Case 1 To 4, 7 To 9, 11, 13, Is > MaxNumber

4. Structures répétitives

4.1. La structure répétitive (Tant que ... Faire ...)

```
While <Condition>
```

```
    <Instructions à répéter>
```

```
End While
```

Exemple : Multiplication de 2 entiers positifs en n'utilisant que l'addition.

Produit := A x B ou bien Produit:= A+A+A+ ... +A

L'addition est effectuée B fois et la valeur de la variable B est stockée dans une variable décrémentée à chaque boucle. Boucle dans laquelle on additionne la valeur de A à la valeur précédente de Produit (Produit est initialisé à 0).

```
Imports System.Console
```

```
Module Module1
```

```
Sub Main()
```

```
    Dim Produit As Integer
```

```
    Dim A As Integer
```

```
    Dim B As Integer
```

```
    Dim Compteur As Integer
```

```
    Produit = 0
```

```
    Write("Veuillez indiquer un entier positif: ")
```

```
    A = ReadLine()
```

```
    Write("Veuillez indiquer un autre entier positif: ")
```

```
    B = ReadLine()
```

```
    Compteur = B
```

```
    While Compteur <> 0
```

```
        Produit = Produit + A
```

```
        Compteur = Compteur - 1
```

```
    End While
```

```
    Write("Le produit de " & A & " par " & B & " est égal à: " & Produit)
```

```
    ReadKey()
```

```
End Sub
```

```
End ModuleEND.
```

Pourquoi a-t-on conservé la valeur initiale de B ? La valeur de B a été conservée car on en a besoin pour l'affichage du résultat.

Que se passe-t-il si malgré le message, l'utilisateur introduit un ou deux nombres négatifs ?

Vous constaterez que dans le cas où $B < 0$, on ne sort plus de la boucle ! On dit alors de l'algorithme qu'il boucle !

Il faut donc "protéger" l'utilisateur en bouclant sur l'introduction des données tant qu'elles ne sont pas positives.

```

Imports System.Console
Module Module1

Sub Main()
    Dim Produit As Integer
    Dim A As Integer
    Dim B As Integer
    Dim Compteur As Integer
    Produit = 0
    Write("Veuillez indiquer un entier positif: ")
    A = ReadLine()
    While A < 0
        Write("Veuillez indiquer un entier positif: ")
        A = ReadLine()
    End While
    Write("Veuillez indiquer un autre entier positif: ")
    B = ReadLine()
    While B < 0
        Write("Veuillez indiquer un autre entier positif: ")
        B = ReadLine()
    End While
    Compteur = B
    While Compteur <> 0
        Produit = Produit + A
        Compteur = Compteur - 1
    End While
    Write("Le produit de " & A & " par " & B & " est égal à: " & Produit)
    ReadKey()  End Sub

End Module

```

Note : Cet exemple très simple montre les 3 points à vérifier lors de l'utilisation d'une structure répétitive :

- L'initialisation des variables qui interviennent dans l'expression logique ;
- L'établissement d'une expression logique correcte, c'est-à-dire provoquant à un moment donné un arrêt du traitement ;
- Les précautions à prendre en cas d'intervention sur les variables utilisées dans l'expression logique.

4.2. La structure répétitive (Répéter ... Jusqu'à ...)

Syntaxe :

```

Do
  <Instructions>
Loop Until <expression logique>;

```

La sémantique est la suivante:

1. Le bloc <instructions> est exécuté.
2. L'expression logique est testée.
3. Dans le cas où elle est fausse, on recommence au point 1.

Dans le cas où elle est vraie, le programme poursuit son exécution après l'instruction "Until"

Remarques :

- Avec cette forme de structure répétitive, le bloc d'instructions est exécuté au moins une fois ! Cela peut être source d'erreurs dans certains cas.
- La condition exprimée par <expression logique> est l'expression contraire de celle exprimée dans la structure répétitive de type "While".

Exemple :

Ainsi pour passer de la structure While à la structure Repeat dans le programme, il suffit d'inverser les conditions d'arrêt. Compteur $\neq 0$ devient Compteur = 0 , (A<0) devient (A>=0) et (B<0) devient (B>=0).

```
Imports System.Console
```

```
Module Module1
```

```
Sub Main()
```

```
    Dim Produit As Integer
```

```
    Dim A As Integer
```

```
    Dim B As Integer
```

```
    Dim Compteur As Integer
```

```
    Produit = 0
```

```
    Do
```

```
        Write("Veuillez indiquer un entier positif: ")
```

```
        A = ReadLine()
```

```
    Loop Until A >= 0
```

```
    Do
```

```
        Write("Veuillez indiquer un autre entier positif: ")
```

```
        B = ReadLine()
```

```
    Loop Until B >= 0
```

```
    Compteur = B
```

```
    Do
```

```
        Produit = Produit + A
```

```
        Compteur = Compteur - 1
```

```
    Loop Until Compteur = 0
```

```
    Write("Le produit de " & A & " par " & B & " est égal à: " & Produit)
```

```
    ReadKey()
```

```
End Sub
```

```
End Module
```

INFO / 1

Mais dans le cas où B est inférieur ou égal à zéro l'algorithme boucle car les instructions comprises dans la boucle sont exécutées au moins une fois ! D'où la nécessité de forcer l'utilisateur à introduire deux valeurs strictement positives!

4.3. La structure repetitive For

```
For <compteur> = <borne inférieur> To <borne supérieur> <instruction>
```

```
Next <compteur>
```

Exemple : Table de multiplication de 10.

```
Imports System.Console

Module Module1

    Sub Main()
        Dim k As Integer
        WriteLine("Table de multiplication de 10")
        For k = 1 To 10
            WriteLine(k & "*10=" & (k * 10))
        Next
        ReadKey()
    End Sub

End Module
```

5. Les tableaux

5.1. Syntaxes

`Dim Tableau(3) As Integer` 'déclare un tableau de 4 entiers

On remarque que, dès la déclaration du tableau, le nombre d'éléments est bien défini et restera toujours le même.

`Dim Tableau(3) As Integer` entraîne la création des variables 'Integer' suivante :

Tableau (0)

Tableau (1)

Tableau (2)

Tableau (3)

Contenu du tableau :

0
0
0
0

soit 4 éléments

Noter que comme c'est un tableau d'entier, juste après la création du tableau les éléments sont initialisés à 0.

Le tableau commence toujours par l'indice 0

Le nombre d'éléments dans le tableau est toujours égal à l'indice de dimension + 1 (ou l'indice du dernier élément+1)

5.2. Les tableaux à plusieurs dimensions

`Dim T(2,2) 3 X 3 éléments`

Pour un tableau à 2 dimensions, le premier argument représente les lignes, le second les colonnes.

Voyons pour chaque élément du tableau le numéro de ligne et celui de la colonne: (pas le contenu des éléments ici mais leur index).

élément:0,0	élément:0,1	élément:0,2
élément:1,0	élément:1,1	élément:1,2
élément:2,0	élément:2,1	élément:2,2

Exemple :

La première ligne comporte les 3 éléments: T(0,0) T(0,1) et T(0,2)

Pour mettre 33 dans l'élément central:

Dim T(2,2) As Integer

T(1,1)=33

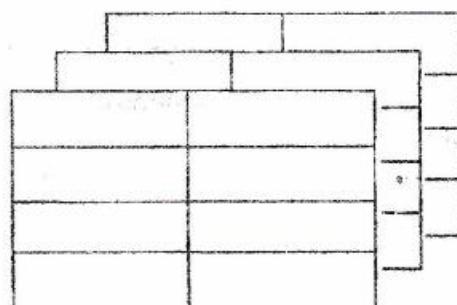
Voyons le contenu du tableau :

0	0	0
0	33	0
0	0	0

Il est possible de créer des tableaux à 3, 4, ..., n dimensions.

Exemple :

Dim T(3,1,2) crée un tableau de 4x2x3 éléments.



5.3. Initialisation pendant la déclaration

On peut initialiser un tableau (Donner une valeur aux éléments).

En effet après dimensionnement d'un tableau, il contient :

- La valeur 0 si c'est un tableau de numérique.
- Nothing si c'est un tableau de String ou d'Objet.

Dim mois(11) As String

'mois (1) contient Nothing

mois(0)= "Janvier"

mois(1)= "Février"

mois(2)= "Mars"

ou lors de sa déclaration :

Dim Mois() As String = {Janvier,Février,Mars}

On remarque ici, que le nom d'élément n'est pas indiqué; comme on initialise 3 éléments, le tableau en aura 3.

Autre syntaxe :

Dim t As String() 'déclaration

t = New String(1) {"One", "Two"} ' on affecte au tableau un nouveau tableau de String contenant "One" et "Two".

Dim R(,) as Integer = {{0, 1}, {1, 2}, {0, 0}, {2, 3}}

On déclare et on initialise en même temps un tableau à 2 dimensions, remarquez qu'on rentre les éléments 2 à 2.

(Équivalent à R(0,0)=0 R(0,1)=1 R(1,0)=1 R(1,1)=2 ...)

5.4. Redimensionnement

Redim permet de redimensionner un tableau (modifier le nombre d'éléments d'un tableau existant), si on ajoute **Preserve** les anciennes valeurs seront conservées.

Attention, on ne peut pas modifier le nombre de dimension, ni le type des données. Un tableau à 2 dimensions de 20 fois 20 string pourra être redimensionné en tableau de 30 fois 30 String, mais pas en tableau d'entiers ou à 3 dimensions.

Dim T(20,20) As String

...

Redim Preserve T(30,30)

Il est possible d'écrire Dim T(,) As String

Dim T(,) As String 'Sans donner les dimensions du tableau :

Il est déclaré mais n'existe pas car T(1,1)= "toto" déclenche une erreur.

Il faut avant de l'utiliser écrire Redim T(30,30), (sans remettre As String).

Certaines instructions comme Split redimensionnent elles-mêmes le tableau au nombre d'éléments nécessaires.

```
Dim Nom() as String
```

```
Nom=S.Split(Separateur)
```

5.5. Autres méthodes

Erase efface le tableau et récupère l'espace.

Erase Tableau (équivalent à tableau= Nothing)

Clear réinitialise le tableau. (remise à 0 d'un tableau de numérique par exemple)

Array.Clear(t, 2, 3) réinitialisation tableau t à partir de l'élément 1 et pour 3 éléments.

6. Programmation modulaire

6.1. Syntaxe

```
Sub <Nom procédure>(<Déclaration paramètres>)
    ...
End Sub
```

Dans la partie déclaration des paramètres, si on veut expliciter le mécanisme de passage, il faut déclarer les paramètres avec :

- ByRef (Par adresse) ;
- ByVal (Par valeur).

6.2. Passage par adresse

```
Imports System.Console
Module Module1
    Sub permute(ByRef a As Integer, ByRef b As Integer)
        Dim permut As Integer
        permut = a
        a = b
        b = permut
    End Sub
    Sub Main()
        Dim v1 As Integer
        Dim v2 As Integer
        Dim v3 As Integer
        Write("Donnez un nombre: ")
        v1 = ReadLine()
        Write("Donnez un nombre: ")
        v2 = ReadLine()
        Write("Donnez un nombre: ")
        v3 = ReadLine()
        If v1 > v2 Then permute(v1, v2)
        If v3 < v2 Then
            If v3 < v1 Then
                permute(v1, v3)
                permute(v3, v2)
            End If
        End If
        WriteLine("Voici les trois nombres dans l'ordre croissant : " & v1 & " " & v2 & "
" & v3)
        ReadKey()
    End Sub
End Module
```

Donnez un nombre : 3
Donnez un nombre : 2
Donnez un nombre : 6
Voici les trois nombres dans l'ordre croissant : 2 3 6

6.3. Passage par valeur

```
Imports System.Console
Module Module1
    Sub permute(ByVal a As Integer, ByVal b As Integer)
        Dim permut As Integer
        permut = a
        a = b
        b = permut
    End Sub
    Sub Main()
        Dim v1 As Integer
        Dim v2 As Integer
        Dim v3 As Integer
        Write("Donnez un nombre: ")
        v1 = ReadLine()
        Write("Donnez un nombre: ")
        v2 = ReadLine()
        Write("Donnez un nombre: ")
        v3 = ReadLine()
        If v1 > v2 Then permute(v1, v2)
        If v3 < v2 Then
            If v3 < v1 Then
                permute(v1, v3)
                permute(v3, v2)
            End If
        End If
        WriteLine("Voici les trois nombres dans l'ordre croissant : " & v1 & " " & v2 & " "
        & v3)
        ReadKey()
    End Sub
End Module
```

Donnez un nombre : 3

Donnez un nombre : 2

Donnez un nombre : 6

Voici les trois nombres dans l'ordre croissant : 3 2 6

6.4. Visibilité d'un identificateur

6.5. Les fonctions définies par l'utilisateur

```
Function <Nom fonction>(<Déclaration paramètres>) As <Type valeur
retournée>
    <Nom fonction>=<Valeur rentrée>
End Function
```

Ecrivons le programme permettant de calculer le nombre de combinaisons de P éléments pris parmi N éléments sachant que ce nombre est déterminé par la formule suivante:

$N! / (P! * (N-P)!)$ où ! signifie factorielle de

La nature même de l'exercice nous incite à utiliser une fonction. La fonction Factorielle n'étant pas standard, nous devons la décrire.

```
Imports System.Console
Module Module1
    Function factorielle(ByVal n As Integer) As Integer
        Dim fact As Integer
        Dim compteur As Integer
        fact = 1
        For compteur = 2 To n
            fact = fact * compteur
        Next
        factorielle = fact
    End Function
    Sub Main()
        Dim P As Integer
        Dim N As Integer
        Dim Combinaison As Double
        Write("Veuillez indiquer la valeur de P:")
        P = ReadLine()
        Write("Veuillez indiquer la valeur de N:")
        N = ReadLine()
        Combinaison = factorielle(N) / (factorielle(P) * factorielle(N - P))
        WriteLine("Le nombre de combinaison de " & P & " parmi " & N & " est égal à
" & Combinaison)
        ReadKey()
    End Sub
End Module
```

7. Série d'exercices n°1

7.1. Enoncés

7.1.1. Instructions d'affectation et d'entrée-sortie

1. Ecrire l'algorithme permettant de calculer la circonference et la surface d'un cercle.
2. Ecrire l'algorithme permettant à partir d'un montant hors taxes saisi d'afficher le montant T.T.C.
3. Ecrire l'algorithme permettant de calculer la sommes des N premiers nombres entiers positifs à partir de la formule mathématique suivante $S = (N \cdot (N+1)) / 2$.
4. Ecrire l'algorithme permettant de calculer le quotient et le reste de la division de 2 entiers positifs

7.1.2. Instructions conditionnelles

5. Ecrire avec une structure alternative l'algorithme permettant de résoudre une équation du second degré. (On utilisera l'opérateur rac 2 qui donne la racine carrée d'un nombre).
6. Ecrire avec une structure de choix l'algorithme permettant de simuler une calculatrice.

On saisit :

- Deux nombres
- Puis un opérateur parmi l'ensemble (+,-,* , /, %, mod.div, ^)

Le calcul à effectuer est fonction de l'opérateur saisi, le programme boucle sur la saisie d'un nouvel opérateur ou jusqu'à ce que l'utilisateur tape le mot 'stop'.

7.1.3. Instructions répétitives

7. calculez la somme des N premiers nombres entiers pairs.
8. Saisir n nombres réels, en calculant la somme et la moyenne. Le nombre de nombres saisis est indiqué par l'utilisateur.
9. Afficher la table de multiplication (jusqu'à 0) d'un nombre n
10. Calculer la factorielle d'un nombre entier.

On rappelle que la factorielle de 5 est égale à $5 \times 4 \times 3 \times 2 \times 1 = 120$.

7.1.4. Exercice récapitulatif avec structures alternatives et répétitives

Ecrire l'algorithme permettant de rechercher tous les nombres parfaits inférieurs à 10000.

Un nombre parfait est un nombre entier positif présentant la particularité d'être égal à la somme de tous ses diviseurs, excepté lui-même.

Exemple : $6 = 1 + 2 + 3$

Les nombres parfaits inférieurs à 10000 sont 6, 28, 49 et 8128.

7.2. Corrigés

7.2.1. Programme permettant de calculer la circonférence et la surface d'un cercle

```

Imports System.Console

Module Module1

    Sub Main()
        Dim rayon As Double
        Dim circonference As Double
        Dim surface As Double
        Const pi = 3.14

        Write("Entrez le rayon: ")
        rayon = ReadLine()
        circonference = 2 * pi * rayon
        surface = pi * rayon ^ 2
        WriteLine("La circonference est de " & circonference & " et la surface est de " &
surface)
        ReadKey()
    End Sub

End Module

```

7.2.2. Programme permettant de calculer le montant TTC

```
Imports System.Console  
Module Module1  
  
Sub Main()  
    Dim MontantHT As Integer  
    Dim MontantTTC As Double  
    Const TauxTVA = 20  
    Write("Saisir le montant hors taxe:")  
    MontantHT = ReadLine()  
    MontantTTC = MontantHT * (1 + TauxTVA / 100)  
    WriteLine("Le montant TTC est de " & MontantTTC)  
    ReadKey()  
End Sub  
  
End Module
```

7.2.3. Programme permettant de calculer la somme des n premiers nombres entiers

```
Imports System.Console  
Module Module1  
  
Sub Main()  
    Dim somme As Double, n As Double  
    Write("Veuillez indiquer un nombre entier: ")  
    n = ReadLine()  
    somme = (n * (n + 1)) / 2  
    WriteLine("La somme est de ")  
End Sub  
  
End Module
```

7.2.4. Programme permettant de calculer le quotient et le reste de la division de 2 entiers

```
Imports System.Console
Module Module1

Sub Main()
    Dim a As Integer, b As Integer, r As Integer, q As Integer
    Write("Entrez un nombre entier: ")
    a = ReadLine()
    Write("Entrez un autre nombre entier: ")
    b = ReadLine()
    q = a \ b
    r = a Mod b
    WriteLine("Le quotient est " & q & " et le reste est " & r)
    ReadKey()
End Sub

End Module
```

7.2.5. Programme permettant de calculer la somme des n premiers nombres entiers pairs

```
Imports System.Console
Module Module1

Sub Main()
    Dim n As Integer, compteur As Integer, somme As Integer, pair As Integer
    Write("Entrer un nombre: ")
    n = ReadLine()
    somme = 0
    compteur = 0
    pair = 0
    While compteur <> n
        pair = pair + 2
        somme = somme + pair
        compteur = compteur + 1
    End While
    WriteLine("La somme des " & n & " premiers nombres pairs est " & somme)
    ReadKey()
End Sub

End Module
```

7.2.6. Programme permettant de calculer la somme et la moyenne de n nombres

```

Imports System.Console
Module Module1

Sub Main()
    Dim somme As Double, moyenne As Double, nombre As Double
    Dim n As Integer, compteur As Integer
    Write("Combien de chiffres voulez vous saisir ? : ")
    n = ReadLine()
    somme = 0
    compteur = 0
    While compteur <> n
        Write("Saisir un chiffre: ")
        nombre = ReadLine()
        compteur = compteur + 1
        somme = somme + nombre
        Clear() 'Efface l'écran
    End While
    moyenne = somme / n
    WriteLine("La moyenne est de " & moyenne & " et la somme est de " & somme)
    ReadKey()
End Sub

```

7.2.7. Programme permettant d'afficher la table de multiplication d'un nombre

```

Imports System.Console
Module Module1

Sub Main()
    Dim nombre As Integer, k As Integer
    Write("De quel nombre voulez-vous la table de multiplication ? : ")
    nombre = ReadLine()
    For k = 1 To 10
        WriteLine(nombre & " * " & k & " = " & (nombre * k))
    Next
    ReadKey()
End Sub

End Module

```

7.2.8. Programme de calcul de la factorielle de n

```

Imports System.Console
Module Module1

Sub Main()
    Dim n As Integer, compteur As Integer, facto As Integer
    Write("Entrez un nombre: ")
    n = ReadLine()
    facto = 1
    For compteur = 2 To n
        facto = facto * compteur
    Next
    WriteLine("La factorielle de " & n & " est de " & facto)
    ReadKey()
End Sub

End Module

```

7.2.9. Programme permettant d'identifier les nombres parfaits

```

Imports System.Console
Module Module1

Sub Main()
    Dim nombre As Integer, diviseur As Integer, somme As Integer
    WriteLine("Liste des nombres parfaits inférieurs à 10000: ")
    For nombre = 1 To 10000
        somme = 0
        For diviseur = 1 To nombre - 1
            If nombre Mod diviseur = 0 Then somme = somme + diviseur
        Next
        If somme = nombre Then WriteLine(nombre)
    Next
    ReadKey()
End Sub

End Module

```

8. Série d'exercices n°2

8.1. Recherche d'un élément dans un tableau

8.1.1. Recherche à l'aide de la position relative de l'élément

```
Imports System.Console  
Module Module1  
  
Sub Main()  
    Dim tnote(100) As Double  
    Dim NombreNote As Integer  
    Dim compteur As Integer  
    Dim position As Integer  
  
    Write("Entrez le nombre de notes à saisir: ")  
    NombreNote = ReadLine()  
    For compteur = 0 To NombreNote - 1  
        Write("Entrez une note: ")  
        tnote(compteur) = ReadLine()  
        Clear()  
    Next  
    Write("Donnez la position de l'élément recherché: ")  
    position = ReadLine()  
    WriteLine("La note est " & tnote(position))  
    ReadKey()  
End Sub  
  
End Module
```

8.1.2. Recherche à l'aide de la valeur de l'élément

```
Imports System.Console  
Module Module1  
  
Sub Main()  
    Dim tnotes(100) As Double  
    Dim NombreNotes As Integer  
    Dim compteur As Integer  
    Dim ValeurRecherche As Double  
    Dim trouve As Boolean  
  
    Write("Entrez le nombre de notes à saisir: ")  
    NombreNotes = ReadLine()  
    For compteur = 0 To NombreNotes - 1  
        Write("Entrez une note: ")  
        tnotes(compteur) = ReadLine()  
        Clear()  
    Next  
    Write("Donnez la valeur de l'élément à rechercher: ")  
    ValeurRecherche = ReadLine()  
    compteur = 1  
    trouve = False  
    While (compteur <= NombreNotes) And (Not trouve)  
        If tnotes(compteur) = ValeurRecherche Then  
            WriteLine("La note est à la position " & compteur)  
            trouve = True  
        End If  
        compteur = compteur + 1  
    End While  
    If Not trouve Then WriteLine("La note n'est pas dans le tableau.")  
    ReadKey()  
End Sub  
  
End Module
```

8.1.3. Recherche de l'élément le plus grand

Méthode : L'idée est d'initialiser le maximum avec la valeur du premier élément du tableau. Ensuite, comparer successivement le maximum avec chaque élément. Si la valeur de l'élément est plus grande que celle du maximum, on affecte cette valeur à maximum.

```

Imports System.Console
Module Module1

Sub Main()
    Dim tnotes(100) As Double
    Dim NombreNotes As Integer
    Dim compteur As Integer
    Dim Maximum As Double
    Dim position As Integer

    Write("Entrez le nombre de notes à saisir: ")
    NombreNotes = ReadLine()
    For compteur = 0 To NombreNotes - 1
        Write("Entrez une note: ")
        tnotes(compteur) = ReadLine()
        Clear()
    Next
    For compteur = 0 To tnotes.Count - 1
        If Maximum < tnotes(compteur) Then
            Maximum = tnotes(compteur)
            position = compteur
        End If
    Next
    WriteLine("Le maximum est " & Maximum & " et il se trouve à la " & (position + 1)
    & "° position")
    ReadKey()
End Sub

End Module

```

9. Série d'exercices n° 3

9.1. Exercices sur les tableaux à une dimension

9.1.1. Exercice 1

Ecrire le programme permettant :

- De saisir 2 tableaux de nombres réels de même taille chacun ;
- De saisir un opérateur (*, 1, +, -) ;
- D'afficher le résultat éléments 1^e tableau **opérateur** éléments 2^e tableau ;
- De boucler sur la saisie d'un autre opérateur jusqu'à ce que le caractère 's' (comme stop) soit saisi.

```
Imports System.Console
Module Module1

    Sub Main()
        Dim tab1(100) As Double
        Dim tab2(100) As Double
        Dim k As Integer, nombre As Integer
        Dim resultat As Double
        Dim operateur As Char

        Write("Entrez le nombre d'éléments: ")
        nombre = ReadLine()
        WriteLine("Le premier tableau:")
        For k = 0 To nombre - 1
            Write("Entrer le réel de rang " & k & ":")
            tab1(k) = ReadLine()
        Next
        WriteLine("Le deuxième tableau:")
        For k = 0 To nombre - 1
            Write("Entrer le réel de rang " & k & ":")
            tab2(k) = ReadLine()
        Next
        Write("Entrez l'opérateur choisi (*,+,-) ou s : ")
        operateur = ReadLine()
        While operateur <> "s" And operateur <> "S"
            resultat = 0
            For k = 0 To nombre - 1
                Select Case operateur
                    Case "+" : resultat = tab1(k) + tab2(k)
                    Case "-" : resultat = tab1(k) - tab2(k)
                    Case "*" : resultat = tab1(k) * tab2(k)
                    Case "/" : resultat = tab1(k) / tab2(k)
                    Case Else : WriteLine("Opérateur non supporté !")
                End Select
            Next
        End While
    End Sub
End Module
```

37

```

End Select
WriteLine("Le résultat de " & tab1(k) & operateur & tab2(j) & "=" & resultat)
Next
Write("Entrez l'opérateur choisi (*,+,-) ou s : ")
operateur = ReadLine()
End While
WriteLine("Au revoir !")

ReadKey()
End Sub

End Module

```

9.1.2. Exercice 2

Ecrire le programme permettant de trier un tableau de nombres.

On utilise une méthode de tri par sélection. On cherche la position du plus grand élément parmi les n éléments du tableau. On permute cet élément avec le dernier élément du tableau. On recommence de la même façon avec les n-1 premiers éléments du tableau, puis les n-2 éléments et ainsi de suite.

Exemple : soit à trier le tableau suivant :

11	2	5	8
----	---	---	---

Recherche du maximum parmi les quatre éléments du tableau et permutation avec le dernier élément :

8	2	5	11
---	---	---	----

Recherche du maximum parmi les trois premiers éléments du tableau et permutation avec le dernier élément :

5	2	8	11
---	---	---	----

Recherche du maximum parmi les deux premiers éléments du tableau et permutation avec le dernier élément :

2	5	8	11
---	---	---	----

Le tableau est trié !

```

Imports System.Console
Module Module1

Sub Main()
    Dim t(100) As Integer
    Dim permut, MaxPos, Nb, k, j As Integer
    Do
        Write("Entrer le nombre d""éléments du tableau (<=100): ")
        Nb = ReadLine()
    Loop Until Nb <= 100
    For k = 1 To Nb
        Write("Entrer le nombre à l""indice n°", k, ": ")
        t(k) = ReadLine()
    Next
    For k = Nb To 1 Step -1
        MaxPos = k
        For j = k To 1 Step -1
            If t(j) > t(MaxPos) Then
                MaxPos = j
                permut = t(k)
                t(k) = t(MaxPos)
                t(MaxPos) = permut
            End If
        Next
    Next
    WriteLine("Après triage:")
    For Each i As Object In t
        If Not i.Equals(0) Then
            WriteLine(i)
        End If
    Next
    ReadKey()
End Sub

End Module

```

9.1.3. Exercice 3

Ecrire le programme permettant de trier un tableau de nombres :

La méthode utilisée cette fois est dite méthode du tri "bulle". On parcourt le tableau en comparant chaque élément avec son suivant et en permutant les éléments lorsque $T[k] > T[k+1]$. On refait des balayages tant qu'il y a eu des échanges (détectés par une variable de type booléen).

Ainsi les éléments les plus grands "montent" peu à peu vers la fin du tableau, comme des bornes.

Exemple :

Soit à trier le tableau suivant:

11	0	5	3
----	---	---	---

1^e passage : rangement par comparaisons successives de la plus grande valeur à droite du tableau.

11	0	5	3
0	11	5	3
0	5	11	3
0	5	3	11

La bulle 11 est rangée.

2^e passage (puisque il y avait une permutation)

0	3	5	11
---	---	---	----

La bulle 5 est rangée.

3^e passage (puisque il y avait une permutation)

0	3	5	11
---	---	---	----

Il n'y avait plus ce permutation. Donc, le programme s'arrête car le tableau est trié !

```

Imports System.Console
Module Module1

Sub Main()
    Dim t(100) As Integer
    Dim permut, MaxPos, Nb, k, j As Integer
    Dim permutation As Boolean
    Do
        Write("Entrer le nmbre d""éléments du tableau (<=100): ")
        Nb = ReadLine()
    Loop Until Nb <= 100
    For k = 0 To Nb - 1
        Write("Entrer le nombre à l""indice n"", k, ": ")
        t(k) = ReadLine()
    Next
    Do
        permutation = False
        For k = 0 To Nb - 2
            If t(k) > t(k + 1) Then
                permut = t(k)
                t(k) = t(k + 1)
                t(k + 1) = permut
                permutation = True
            End If
        Next
        Loop Until Not permutation

        WriteLine("Après triage:")
        For Each i As Object In t
            If Not i.Equals(0) Then
                WriteLine(i)
            End If
        Next
        ReadKey()
    End Sub

End Module

```

9.2. Exercices sur les tableaux à deux dimensions

Ecrire le programme permettant la saisie et l'addition de deux matrices de nombre entier 10 x 10.

```

Imports System.Console

Module Module1

    Sub Main()
        Dim t1(10, 10), t2(10, 10), t3(10, 10) As Integer
        Dim nbl, nbc, i, c As Integer

        Write("Nombre de lignes (<=10): ")
        nbl = ReadLine()
        Write("Nombre de colonnes (<=10): ")
        nbc = ReadLine()
        Clear()
        WriteLine("Tableau n°1")
        For i = 0 To nbl - 1
            For c = 0 To nbc - 1
                Write("t1(" & i & "," & c & ")=")
                t1(i, c) = ReadLine()
            Next
        Next
        WriteLine("Tableau n°2")
        For i = 0 To nbl - 1
            For c = 0 To nbc - 1
                Write("t2(" & i & "," & c & ")=")
                t2(i, c) = ReadLine()
            Next
        Next

        For i = 0 To nbl - 1
            For c = 0 To nbc
                t3(i, c) = t1(i, c) + t2(i, c)
            Next
        Next

        WriteLine("Tableau résultat")
        For i = 0 To nbl - 1
            For c = 0 To nbc - 1
                WriteLine("t3(" & i & "," & c & ")=" & t3(i, c))
            Next
        Next
        ReadKey()
    End Sub

End Module

```

9.3. Exercices sur les procédures et fonctions

Ecrire l'algorithme permettant la saisie de notes dans un tableau, leur affichage, l'affichage de leur somme, de leur moyenne, et des notes minimales et maximales. Pour cela vous utiliserez les procédures et fonctions suivantes :

- La procédure **Saisie Dimensions** permet la saisie du nombre d'éléments du tableau ;
- La procédure **SaisieTableau** permet la saisie dans un tableau à une dimension d'un ensemble de notes ;
- La procédure **AfficheTableau** permet l'affichage des notes du tableau ;
- La fonction **Somme** renvoie la somme des notes ;
- La fonction **Moyenne** renvoie la moyenne des notes ;
- La fonction **Maximum** renvoie la note maximale ;
- La fonction **Minimum** renvoie la note minimale.

```
Imports System.Console
Module Module1
    Sub SaisieDimension(ByRef n As Integer)
        Do
            Write("Nombre d'éléments (entre 1 et 10): ")
            n = ReadLine()
        Loop Until n >= 1 And n <= 10
    End Sub
    Sub SaisieTableau(Val n As Integer, ByRef t() As Integer)
        Dim k As Integer

        For k = 0 To n - 1
            Write("t(" & k & ")=")
            t(k) = ReadLine()
        Next
    End Sub
    Sub AfficheTableau(Val n As Integer, Val t() As Integer)
        Dim k As Integer
        Clear()
        For k = 0 To n - 1
            Write("t(" & k & ")=" & t(k) & " ")
        Next
        WriteLine("")
    End Sub
End Module
```

```
Function SommeTableau(ByVal n As Integer, ByVal t() As Integer) As Integer
    Dim somme As Integer = 0
    For k As Integer = 0 To n - 1
        somme = somme + t(k)
    Next
    SommeTableau = somme
End Function

Function MoyenneTableau(ByVal n As Integer, ByVal t() As Integer) As Double
    Dim moyenne As Double
    moyenne = SommeTableau(n, t) / n
    MoyenneTableau = moyenne
End Function

Function Minimum(ByVal n As Integer, ByVal t() As Integer) As Integer
    Dim min As Integer = t(0)
    For k As Integer = 1 To n - 1
        If t(k) < min Then min = t(k)
    Next
    Minimum = min
End Function

Function Maximum(ByVal n As Integer, ByVal t() As Integer) As Integer
    Dim max As Integer = t(0)
    For k As Integer = 1 To n - 1
        If t(k) > max Then max = t(k)
    Next
    Maximum = max
End Function

Sub Main()
    Dim t(10) As Integer
    Dim nbElements As Integer

    SaisieDimension(nbElements)
    SaisieTableau(nbElements, t)
    AfficheTableau(nbElements, t)
    WriteLine("Somme=" & SommeTableau(nbElements, t))
    WriteLine("Moyenne=" & MoyenneTableau(nbElements, t))
    WriteLine("Minimum=" & Minimum(nbElements, t))
    WriteLine("Maximum=" & Maximum(nbElements, t))

    ReadKey()
End Sub

End Module
```