

# TRADUIRE SON ALGORITHME EN C#

1

**Vous trouvez ci-dessous des aides pour traduire votre pseudo code en code C#. Vous êtes libre d'enrichir votre document au fur et à mesure de vos découvertes.**

# DU PSEUDO CODE À C#

Pseudo code	C#
<u>entier</u> nb1,nb2 <u>chaine</u> ch1 <u>réel</u> r <u>double</u> d <u>booleen</u> b	int nb1,nb2; string ch1; float r; double d; bool b;
<u>chaine</u> a <u>lire</u> a	string a; a = Console.ReadLine();
<u>entier</u> b; <u>lire</u> b	int b; b = int.Parse(Console.ReadLine());
<u>ecrire</u> c	Console.WriteLine( c );
//commentaire	//commentaire

# DU PSEUDO CODE À C#: INSTRUCTIONS CONDITIONNELLES

Pseudo code	C#
<u>si</u> condition <u>alors</u> action(s) vraie(s) <u>sinon</u> action(s) fausse(s) <u>finsi</u>	if (condition) { action(s) vraie(s); } else { action(s) fausse(s); }
Ex : <u>Si</u> nbre1 supérieur ou égal nbre2 <u>Alors</u> ok <-vrai <u>Sinon</u> ok<-faux <u>Fin si</u>	<pre>if (nbre1 &gt;= nbre2) {     ok = true; } else {     ok = false;. }</pre>

# DU PSEUDO CODE À C#: INSTRUCTIONS CONDITIONNELLES

Pseudo code	C#
<p>Ex : <u>si</u> A inférieur à 8     <u>alors</u>         ok &lt;- faux     <u>sinon</u>         <u>si</u> A inférieur à 10             <u>alors</u>                 ok &lt;- faux             <u>sinon</u>                 <u>si</u> A supérieur à 12                     <u>alors</u>                         ok &lt;- faux                     <u>sinon</u>                         ok &lt;- true                         <u>ecrire</u> ok                     <u>finsi</u>                 <u>finsi</u>     <u>finsi</u></p>	<pre>if (A &lt; 8) {     ok = false; } else {     if (A &lt; 10)     {         ok = false;     }     else     {         if (A &gt; 12)         {             ok = false;         }         else         {             ok = true;             Console.WriteLine(ok.ToString());         }     } }</pre>

# DU PSEUDO CODE À C#: INSTRUCTIONS CONDITIONNELLES

Pseudo code	C#
<p><u>selon</u> variable</p> <p>    <u>choix</u> valeur1 : //action1(s)</p> <p>    <u>choix</u> valeur2 : //action2(s)</p> <p>    ....</p> <p>    <u>choix</u> valeurn : //actionn(s)</p> <p>    <u>défaut</u> : //actionDéfauts(s)</p> <p><u>finsel</u></p>	<pre>switch (variable) {     case valeur1:     {         //action1(s);     }     break;     case valeur2:     {         //action2(s);     }     break;     default:     {         // actionDefault(s);     } }</pre>

# DU PSEUDO CODE À C#: INSTRUCTIONS CONDITIONNELLES

Pseudo code	C#
<p>Ex :</p> <p><u>selon</u> jour_de_semaine</p> <p>    <u>choix</u> 1 : <u>écrire</u> « lundi »</p> <p>    <u>choix</u> 2 : <u>écrire</u> « mardi »</p> <p>    .....</p> <p>    <u>défaut</u> : <u>écrire</u> « Erreur ! »</p> <p><u>fin</u><u>selon</u></p>	<pre>switch (jour_de_semaine) {     case 1: Console.WriteLine("Lundi");             break;     case 2: Console.WriteLine("Mardi");             break;     default: Console.WriteLine("Erreur");             break; }</pre>

# DU PSEUDO CODE À C#: BOUCLES CONDITIONNELLES

Pseudo code	C#
<u>Tant que</u> condition <u>Faire</u> Action(s) <u>Fin tant que</u>	while (condition) { Action(s) }
Ex : <u>Entier</u> A <-1 <u>Tant que</u> A inférieure à 10 <u>Faire</u> <u>Ecrire</u> A A<-A+1 <u>Fin tant que</u>	<pre>int A = -1; while (A &lt; 10) {     Console.WriteLine (A);     A = A + 1; }</pre>

# DU PSEUDO CODE À C#: BOUCLES CONDITIONNELLES

Pseudo code	C#
<p><b><u>pour</u></b> <i>valeur</i>. <b><u>de</u></b> <i>valeurInitial</i> <b><u>à</u></b> <i>valeurFinal</i> <b><u>par</u></b> <i>pas</i> <b><u>faire</u></b> groupe d'opérations <b><u>finpour</u></b></p>	<pre>for (instructions_départ ; condition :instructions_fin_boucle) {     groupe d'opérations ; }</pre>
<p><b><u>pour</u></b> <i>i</i> <b><u>de</u></b> 0 <b><u>à</u></b> 10 <b><u>pas</u></b> 1 <b><u>faire</u></b>     <b><u>écrire</u></b> <i>i</i> <b><u>finpour</u></b></p>	<pre>for(int i=0;i&lt;10;i++) {     Console.Write(i); }</pre>



# DU PSEUDO CODE À C#: BOUCLES CONDITIONNELLES

Pseudo code	C#
<b><u>répéter</u></b> Action(s) <b><u>jusqu'à</u></b> ( <i>condition</i> )	do { Action(s) }while (condition)
Ex : <u>entier</u> A<-1 <u>faire</u> <u>écrire</u> A A<-A+1 <u>tantque</u> A inférieure à 10	<pre>int A = -1; do {     Console.WriteLine(A);     A = A + 1; } while (A &lt; 10);</pre>

# DU PSEUDO CODE À C#: INSTRUCTION D'AFFECTATION ET OPÉRATEURS LOGIQUES

Pseudo code	C#
A<-valeur;	A = valeur;
A<-5	A = 5;
<u>bool</u> A,B	bool A, B;
A <u>ou</u> B	A    B;
A <u>et</u> B	A && B;
A <u>et non</u> B	A && !B;
<u>si</u> A ou b sont vraies	
<u>alors</u>	if (A    B)
<u>finsi</u>	{
	}

# DU PSEUDO CODE À C#: OPÉRATEURS BINAIRES

Pseudo code	C#
<u>entier</u> A,B A <u>ou binaire</u> B A <u>et binaire</u> B	<pre>int A, B;  A   B; A &amp; B;</pre>
<u>entier</u> a,b,c a<-2 b<-10 C<-a ou binaire b	<pre>int a, b, c; a = 2; b = 10; c = a   b; //le resultat est c=10</pre>
+, -, diviser, multiplier, modulo	+, -, /, *, %

# DU PSEUDO CODE À C#: OPÉRATEUR TERNAIRE

Pseudo code	C#
<u>Si</u> condition <u>Alors</u> action(s) vraie(s) <u>Sinon</u> action(s) fausse(s) <u>Fin si</u>	(condition) ? { action(s) vraie(s); } : { action(s) fausse(s); }
Ex : <u>Si</u> nbre1 <u>supérieur ou égal</u> nbre2 <u>Alors</u> valeur <-15 <u>Sinon</u> valeur <-16 <u>Fin si</u>	<pre>valeur = (nbre1 &gt;= nbre2) ? true : false;</pre>
+, -, diviser, multiplier, modulo	+, -, /, *, %

## DU PSEUDO CODE A C#. OPERATION DE CASTING ET PIEGES!

Pseudo code	C#
<u>reel</u> r =(reel)a	float c = (float)d;
Attention : La division d'un entier par un entier est un entier en C#	Ex : int a = 3, b = 5; float c; c = ((float)a / b);

# DU PSEUDO CODE À C#: DÉCLARATION DE TABLEAUX STATIQUES À 1 DIMENSION

Pseudo code	C#
<u>tableau</u> nom_tableau(taille) <u>de</u> type	type[] nom_tableau=new type[taille] ;
Ex : <u>tableau</u> nbStagF(5) d' <u>entier</u> //instanciation nbStagF(0)<-0 nbStagF(1)<-0 nbStagF(2)<-0 nbStagF(3)<-0 nbStagF(4)<-0	<pre>//déclaration et instanciation //1ere possibilite int[] nbStagF = new int[5];  nbStagF[0] = 0; nbStagF[1] = 0; nbStagF[2] = 0; nbStagF[3] = 0; nbStagF[4] = 0;  //déclaration et instanciation //2ere possibilite (CONTRACTE) int[] nbStagF = new int[] { 0,0,0,0,0};</pre>
<u>entier</u> B = nbStagF(4)	int B = nbStagF[4];
<u>entier</u> lg =   nbStagF	int lg = nbStagF.Length;

# DU PSEUDO CODE A C#. DECLARATION DE TABLEAUX STATIQUES À 2 DIMENSIONS

Pseudo code	C#
<u>tableau</u> nom_tableau(n,m) <u>de</u> type	Type[,] Tableau=new Type[n,m];
Ex : <u>tableau</u> tab2Dim(5,2) d' <u>entier</u> //instanciation tab2Dim (0,0)<-0 tab2Dim (1,0)<-0 tab2Dim (2,0)<-0 tab2Dim (3,0)<-0 tab2Dim (4,0)<-0 tab2Dim (0,0)<-0 tab2Dim (1,0)<-0 tab2Dim (2,0)<-0 tab2Dim (3,0)<-0 tab2Dim (4,0)<-0	<pre>//déclaration et instanciation //1ere possibilite int[,] tab2Dim = new int[5,2];  tab2Dim[0,0] = 0; tab2Dim[1,0] = 1; tab2Dim[2,0] = 2; tab2Dim[3,0] = 3; tab2Dim[4,0] = 4; tab2Dim[0,1] = 5; tab2Dim[1,1] = 6; tab2Dim[2,1] = 7; tab2Dim[3,1] = 8; tab2Dim[4,1] = 9;  //déclaration et instanciation //2ere possibilite (CONTRACTE) int[,] <u>tab2Dim</u> = new int[,] { {0,5}, {1,6}, {2,7}, {3,8}, {4,9}};</pre>
<u>entier</u> caseNum1 <- tab2Dim (0,0)	int caseNum1 = tab2Dim[0,0];
<u>entier</u> nbLigne =   tab2Dim   0 <u>entier</u> nbColonne =   tab2Dim   1	int nbLigne = tab2Dim.GetLength(0); int nbColonne = tab2Dim.GetLength(1);

# DU PSEUDO CODE À C#: DÉCLARATIONS DE PROCÉDURE : (UNE PROCÉDURE NE RETOURNE RIEN)

## Pseudo code

**PROCEDURE** *nom\_procedure* (**VAL** *type nom\_arg1*, ..., **VAR** *type nom\_argn*,...)

Déclarations des variables locales

Actions

**FIN PROCEDURE**

## C#

```
void nom_procedure( [type] arg1, ref [type] arg1,..)
{
    //Déclarations des variables locales
    //Actions
}
```

Ex:

```
void Modifier(int nb1, ref int nb2)
{
    nb1 = 15;
    nb2 = 20;
}
```

**//VAL par valeur**

**Le passage par valeur ne modifie pas la valeur de la variable passé en paramètre**

**//VAR par référence**

**Le passage par référence modifie la valeur de la variable passé en paramètre**



# DU PSEUDO CODE À C#: DÉCLARATIONS DE PROCÉDURE : (UNE PROCÉDURE NE RETOURNE RIEN)

## Pseudo code

**PROCEDURE** *nom\_procedure* (**VAL** *type nom\_arg1*, ..., **VAR** *type nom\_argn*,...)

Déclarations des variables locales

Actions

**FIN PROCEDURE**

## C#

```
void Nom_procedure( [type] arg1, ref [type] arg1,..)
```

```
{
```

```
    //Déclarations des variables locales
```

```
    //Actions
```

```
}
```

Ex:

```
void Modifier(int nb1, ref int nb2)
```

```
{
```

```
    nb1 = 15;
```

```
    nb2 = 20;
```

```
}
```

**//VAL par valeur**

**Le passage par valeur ne modifie pas la valeur de la variable passé en paramètre**

**//VAR par référence**

**Le passage par référence modifie la valeur de la variable passé en paramètre**

# DU PSEUDO CODE À C#: DÉCLARATION D'UNE FONCTION: (UNE FONCTION RETOURNE TOUJOURS UNE ET UNE SEULE VALEUR)

## Pseudo code

```
type FONCTION nom_fonction (VAL type nom_arg1, ..., VAR type nom_argn,...)  
//Déclarations des variables locales  
//Actions  
RETOURNE valeur  
FIN FONCTION
```

## C#

```
[type_retour] Nom_fonction ([type] arg1, ref [type] arg2 ...)  
{  
    [type_retour] variableARetourner ;  
    //Déclarations des variables locales  
    //Actions  
    return variableARetourner;  
}
```

**//VAL par valeur**

Le passage par valeur ne modifie pas la valeur de la variable passé en paramètre

**//VAR par référence**

Le passage par référence modifie la valeur de la variable passé en paramètre

# DU PSEUDO CODE À C#: DÉCLARATION D'UNE FONCTION: (UNE FONCTION RETOURNE TOUJOURS UNE ET UNE SEULE VALEUR)

## Exemple: Pseudo code

```
Réel Fonction Somme ( VAL réel nb1, VAR réel nb2)  
    réel somme  
    somme<-nb1+nb2  
    retourne somme  
fin fonction
```

## Exemple: C#

```
float Somme(float nb1,ref float nb2)  
{  
    float somme;  
    somme = nb1 + nb2;  
    return somme; //valeur de retour  
}
```

# DU PSEUDO CODE À C#: FUNCTION

Pseudo code	C#
a <b><u>mod</u></b> b	a%b
<b><u>chaine</u></b> _prénom   <b>prénom</b>	string prénom prénom.Length
<b>prénom</b> <sub>2&lt;-4</sub>	prénom.Substring(2,3)
<b>x<sup>y</sup></b>	Math.Pow(x,y)
'tutu'	@"tutu"

# A VOUS DE CONTINUER!

Pseudo code	C#
...	...
s	