Traduction du langage algorithmique vers C

Les éléments des langages sont écrits en **gras**

Les parties en *italique* doivent être remplacés par des éléments des langages

Langage algorithmique	Langage C
Structure d'un programme	
déclaration des types et des fonctions *déclaration des variables/constantes* **début** *instructions* *II commentaire* *autres_instructions* *I* autre commentaire *I* **fin**	*déclaration des types et des fonctions* **int main(void)** **{** *déclaration des variables/constantes* *instructions* **//** *commentaire* *autres_instructions* **/*** *autre commentaire* ***/** **}**
Types de base	
entier	**int**
réel	**double**
booléen	ajouter au début : **typedef int** bool; **#define** FALSE 0 **#define** TRUE 1 bool
caractère	**char**
chaine	ajouter au début : **typedef char** string[1024]; string
Opérateurs	
affectation : ← arithmétiques : **+ - * / div mod** logiques : **non et ou** comparaison de nombres : **= ≠ < > ≤ ≥** conversion de type : **(entier)** (2*3.5)	affectation : **=** arithmétiques : **+ - * / / %** logiques : **! && ||** comparaison de nombres : **== != < > <= >=** conversion de type : **(int)**(2*3.5)
Déclaration des variables/constantes	
variables : *type* identifiant (ex : **entier** x) *constantes : type* identifiant ← *valeur*	*type* identifiant (ex : **int** x) ***const*** *type* identifiant = *valeur*
Entrées-Sorties	
lire a;	ajouter au début : **#include <stdio.h>** si a est un entier/booléen : **scanf("%d", &**a**);** si a est un réel : **scanf("%lf",&**a**);** si a est une chaine : **scanf("%s",** a**);**
écrire a; **écrire** "la valeur de l'entier est ", a ;	ajouter au début : **#include <stdio.h>** si a est un entier/booléen : **printf("%d",** a**);** si a est un réel : **printf("%lf",** a**);** si a est une chaine : **printf("%s",** a**);** **printf(**"La valeur de l'entier est **%d",** a**);**

Opérations sur les chaines	
chaine c1, c2, c3 ; entier n ;	`string c1, c2, c3 ;` `int n ;`
c1 ← c2 ; c1 **=** c2 c1 ← c2 + c3 ;	ajouter au début : `#include <string.h>` `strcpy(c1, c2);` `strcmp(c1,c2) == 0` `sprintf(c1,"%s%s", c2, c3);`
c1 ← entierEnChaine(n) ;	`sprintf(c1,"%d", n);`
n ← chaineEnEntier(c1) ;	ajouter au début : `#include <stdlib.h>` `n = atoi(c1);`
Conditionnelles	
si (*condition***) alors** *instructions_1* **sinon** *instructions_2* **finsi** NB : le bloc sinon est optionnel	`if (condition)` `{` `  instructions_1` `}` `else` `{` `  instructions_2` `}`
Boucle pour	
pour (i **allant de** 1 **à** 10 **pas** 1**) faire** *instructions* **finpour**	`for (i=1; i<=10; i=i+1)` `{` `  instructions` `}`
pour (i allant de 10 **à** 1 **pas** -1**) faire** *instructions* **finpour**	`for (i=10; i>=1; i=i-1)` `{` `  instructions` `}`
Boucle tant que	
tantque (*condition***) faire** *instructions* **fintantque**	`while (condition)` `{` `  instructions` `}`
Fonctions	
fonction avec retour *type* nom **(***paramètres***)** *déclaration des variables* **début** *instructions* **retourne** *valeur* ; **fin**	`type nom(paramètres)` `{` `  déclaration des variables` `  instructions` `  return valeur ;` `}`
fonction sans retour nom **(***paramètres***)** *déclaration des variables* **début** *instructions* **fin**	`void nom(paramètres)` `{` `  déclaration des variables` `  instructions` `}`

<table>
<tr><th colspan="2">Fonctions prédéfinies</th></tr>
<tr><td>nombreAléatoire()</td><td>ajouter au début :
#include <stdlib.h>
#include <time.h>
puis au début de la fonction <code>int main()</code> :
srand(time(NULL));
rand();</td></tr>
<tr><td>racineCarrée(x)</td><td>ajouter au début :
#include <math.h>
sqrt(x)
puis compiler en ajoutant
-lm</td></tr>
<tr><th colspan="2">Tableaux</th></tr>
<tr><th colspan="2">Déclaration d’un tableau et accès à une case</th></tr>
<tr><td>type Tableau nomTab [longueur] ;
nomTab[i]</td><td>type nomTab[longueur] ;
nomTab[i]</td></tr>
<tr><td>type Tableau nomTab [nbLignes][nbColonnes] ;
nomTab[i][j]</td><td>type nomTab[nbLignes][nbColonnes];
nomTab[i][j]</td></tr>
<tr><th colspan="2">Tableau et fonction</th></tr>
<tr><td>Exemple (tableau à une dimension) :
fonction sans retour nomFct (type Tableau nomTab,
entier longueur)</td><td>void nomFct(int longueur,
type nomTab[longueur])</td></tr>
<tr><td>Exemple (tableau à deux dimensions) :
fonction sans retour nomFct (type Tableau nomTab,
entier nbLignes,
entier nbColonnes)</td><td>void nomFct(int nbLignes,
int nbColonnes,
type nomTab[nbLignes][nbColonnes])</td></tr>
</table>