Département IMA / 3A (S5/S6)
Laure Gonnord et Ingrid Violet

Programmation Structurée et Analyse numérique

Syntaxe Algorithmique, C, Matlab

Syntaxe de base

Nom	Syntaxe Algo	Syntaxe C	Syntaxe MATLAB
Affectation exemple	$\leftarrow$ x$\leftarrow$ 42	`=` `x=42`	`=` `x=42`
Constante numérique	Constante N:10	`#define N 10`	`N=10` Pas une véritable constante (on peut changer la valeur)
Type entier déclaration d'entier	Entier x:Entier	`int` `int x`	Pas de déclaration de variable : inférence automatique de type
Type réel déclaration de réel	Réel r:Réel	`float` `float r`	
Type charactère déclaration de réel	Charactère c:Caractère	`char` `char c`	
Type booléen déclaration de réel	Booléen b:Booléens	`bool` (inclure `stdbool`) `bool b`	
Caractères Booléens constants	'a', 'b' Vrai, Faux	'a','b' true,false	'a', 'b' ?? true,false
expressions booléennes expressions numériques fonctions maths de base	a et b,a ou b, non(a), $x \geq 2$ 2x+21 cos,sin,exp	`a || b`, `a && b`, `!a` ,`x>=2` `2*x+21` idem (inclure `stdmath.h` et lier avec `-lm`)	`a && b`, `~a` ,`x>=2` `2*x+21` fonctions maths " avancées" déjà écrites (dérivée, équas diff, ...)
Tableau Exemple	Vecteur[taille] de type toto:Vecteur[10] d'entiers	`type t[taille]` `int toto[10]`	Tout est matrice. Pas besoin de spécifier la taille
Matrice Exemple	Matrice[n][m] de type titi:Matrice[10][12] d'entiers	`type t[n][m]` `int titi[10][2]`	
Initialisation tableau à 0 Initialisation matrice à 0	(boucle pour) (boucle pour)	`int t[4]={0}` (uniquement à la déclaration); `int t[4][8]={{0}}` (idem);	`zeros(4)` si matrice carrée
Type chaîne exemple Chaînes constantes	Chaîne s : chaîne "toto"	`char[N]` ou `char*` `char s[10]` ou `char* s` `"toto"`	Pas de déclaration idem 'toto'

Structures de contrôle Dans tous les tests si dessous, les *conditions* sont des expressions booléennes (qui s'évaluent donc en vrai/true ou faux/false).

Nom	Syntaxe Algo	Syntaxe C	Syntaxe MATLAB
Conditionnelle I	**Si** *condition* **alors** | instructions si vrai **Fsi**	`if (condition) {` `instructionssivrai` `}`	`if condition` `instructionssivrai` `end`
Conditionnelle II	**Si** *condition* **alors** | instructions si vrai **Sinon** | instructions si faux **Fsi**	`if (condition) {` `instructionssivrai` `} else {` `instructionssifaux` `}` Les accolades ne sont pas nécessaires en cas d'instruction unique.	`if condition` `instructionssivrai` `else` `instructionssifaux` `end`
Conditionnelle III	**Si** *cond1* **alors** | instructions si | cond1 vraie **Sinon** | **Si** *cond2* **alors** | | instructions si | | cond1 fausse | | et cond2 vraie | **Sinon** | | instructions si | | cond1 et | | cond2 fausses | **Fsi** **Fsi**	`if (condition) {` `instructionssivrai` `} else {` `if (cond2) {` `instructionssifaux` `}` `}`	`if condition` `inst1` `elseif cond2` `inst2` `else` `inst3` `end` inst1 faites si cond1 vraie, inst2 si cond1 fausse et cond2 vraie, inst3 sinon.
Boucle Pour	**Pour** *i* **de** *inf* **à** *sup* **Faire** | instructions **Fpour** (incrément de 1 implicite)	`int i;` `for (i=inf; i<=sup; i++){` `instructions` `}` (i++ raccourci pour i=i+1) (on peut aussi décrémenter)	`for i=inf:sup` `instructions` `end` Par défault, l'incrément est 1. Pour spécifier un autre incrément k on écrit : for i=inf:k:sup k peut être positif ou négatif.
Boucle Tant Que	**Tq** *condition* **faire** | instructions **Ftq**	`while (condition){` `instructions` `}`	`while condition` `instructions` `end`

Fonctions, programmes

Nom	Syntaxe Algo	Syntaxe C	Syntaxe MATLAB
Programme principal	**Programme** *Main* .... **Retourner** *0* **FProgramme**	`int main(){` `...` `...` `return 0;` `}`	`...` `...` `...` Pas de fonction "d'entrée" pour un **script** Matlab. Le fichier est exécuté directement, du début à la fin.
Fonction	**Fonction** *fonct(c) : Entier* **D**: c:Charactère **L**: s:Entier .... **Retourner** *s* **FFonction**	`int fonct(char c){` `int s;` `...` `...` `...` `return s;` `}` `return` obligatoire.	`function[s]=fonct(c)` `instruction(s);` `s=instruction;` `end` Remarques : 1. Les points virgule (optionnels) empêchent l'affichage de toutes les instructions dans la fenêtre de commande. 2. Il est important que le nom du fichier contenant la fonction soit **exactement** le même que celui de la fonction. 3. Le `return` est facultatif, mais utile si on veut interrompre le flot d'exécution.
Action	**Action** *monact(a,b,c)* **D**: a : Entier **D/R**: b : Entier **R**: c : Charactère ... ... **FAction** (les modifications faites à b et c sont enregistrées)	`void monact(int a,` `int* pb, int* pc){` `...` `...` `...` `}` (pointeurs : passage par adresse)	Pas de passage par adresse, mais on peut retourner plusieurs résultats : `fonction[b,c]=` `monact(a,b,c)` `...` `...` `...` `end`

Caractéristiques comparées

Caractéristique	C	MATLAB
But des progs	généralistes	calcul scientifique
Langage	compilé	interprété (script)
Passage de paramètres standard	valeur (copie)	valeur
Accès aux adresses	oui (&x)	seulement pour les fonctions (`@nomfonction`)