

Info0301 (L2 INFO)

Structures de données et programmation C

Chapitre 1 : Algorithmique

1. « Algorithmique » ?
 - langage de description
 - algorithme paramétré
2. Structures de contrôle
3. Tableaux : rappels / point de vue *algo*



Ch Jaillet + L. Hollard

- URCA > UFR Sciences > Département Info
- christophe.jaillet@univ-reims.fr
lilian.hollard@univ-reims.fr

1

1. « Algorithmique » ? a. Concepts

Ch Jaillet – URCA
Info0301 : Struct D. ; Prog C
Ch. 1 : Algorithmique

[Unisciel] Algorithmique – Définitions et concepts

<https://ressources.unisciel.fr/algoprogram/s00a00root/aa00module1/co/aa00module1.html>

□ Citations :

- *L'algorithmique c'est le permis de conduire de l'informaticien.*
- *L'algorithmique c'est le couteau bien aiguisé du boucher.*

□ Algorithme :

▪ Le problème :

- De quoi dispose-t-on ?
- Quelles sont les hypothèses ?
- Quelle est la situation de départ ?

⇒ Etant donné [...] on demande [...]
 les données l'objectif

▪ Utilité de l'ordinateur :

- Faire pour nous ...
- de façon fiable ...
- rapidement

▪ Algorithme :

- Exprimer ...
- À Xx, personne ou chose limitée,
- Comment aboutir au résultat

⇒ Nécessite de décrire les étapes nécessaires

⇒ Définition [Enc. Universalis] : *schéma de calcul sous forme d'une suite (finie) d'opérations élémentaires obéissant à un enchaînement déterminé pour parvenir à un résultat.*

NB : *calcul* au sens large (action) ; ...

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

2

2

1. « Algorithme » ?

b. Définitions

□ Algorithme :

- *processus (de calcul) non ambigu, déterministe, fini, exprimé en instructions élémentaires exécutables, et si possible efficace, càd qui atteint l'objectif pour lequel il a été conçu dans un temps optimal, quelles que soient les valeurs des données.*

▪ Propriétés :

- non-ambigüité
- déterminisme
- les étapes élémentaires étant explicites et précises
- finitude

□ Algorithmique :

- *étude formelle des algorithmes*

▪ En particulier

- preuve d'un algorithme (? réalise l'objectif, en un nb fini d'étapes)
- complexité des algorithmes : *une mesure théorique de leurs performances indépendamment d'un environnement matériel et logiciel particulier*

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

3

3

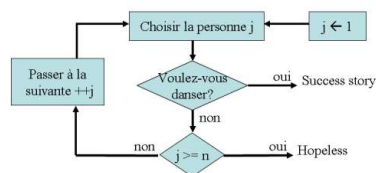
1. « Algorithme » ?

c. Langage de description

□ Langage de description

- *a priori* le langage naturel
- *organigramme de programmation* = *algorithme* = *logigramme*
- *arbre programmatique* (interdépendance de tâches)
- *pseudo-code* = langage algorithmique

□ Exemple : Let's dance !



```
Algorithme LetsDance
Variable j, n : Entier
Variable succes : Booléen
Variable reponse : Chaîne
Début
| Saisir ( n )
| j < 1
| succes <- Faux
| TantQue ( j <= n Et Non succes ) Faire
| | choisir la personne j // ...
| | Afficher ( "Voulez-vous danser ?" )
| | Saisir ( reponse )
| | succes <- ( reponse = "oui" )
| FinTantQue
| Si ( succes ) Alors
| | Afficher ( "Cool !" )
| Sinon
| | Afficher ( "Sans espoir !" )
| FinSi
Fin
```

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

4

4

1. « Algorithme » ?

d. Structure

- La structure d'un algorithme

```
Algorithme Xxx_Generique
Début
| initialisations
| traitement
| conclusions
Fin
```
- Les structures fondamentales :
 - séquence
 - structures conditionnelles : ...
 - structures répétitives : ...
- Thèse de Church-Turing
 - toute fonction calculable est calculable par une machine de Turing
 - tout algorithme peut être décrit à l'aide des trois structures :
 - La séquence
 - La sélective *Si*
 - La répétitive *TantQue*

Ch Jaillet - URCA Info0301 – Ch1 : Algorithmique 5

5

1. « Algorithme » ?

d. Structure

- Analyse descendante (*top-down-design*)
 - ajuster la finesse de description des blocs imbriqués... pour la rendre intelligible à notre interlocuteur
 - *Tout en bas* ? Les instructions élémentaires
 - Lectures, écritures ; affectations
- Matériel nécessaire
 - Variable : identifiant ; type ; valeur
 - déclaration = création
 - ≠ initialisation = associer une valeur
 - Types intégrés = primitifs = fondamentaux = de base
 - Entier, Réel, Booléen ; Caractère, Chaîne
 - Expressions élémentaires
 - Expressions arithmétiques / de comparaison / logiques
 - Conversions implicites (promotion) ; conversion explicite (= transtypage : *casting*)
 - Appels de fonctions mathématiques [pré-existantes]
 - valeur absolue ; partie entière ; arrondi
 - exponentielle, logarithme, racine carrée, ...
 - + appel de primitives [en programmation] :
 - Exemple : nombres aléatoires
 - Fonction d'une bibliothèque externe
 - constante : littéral + identifiant

```
Algorithme Xxx_Generique
| déclaration_des_variables_et_constantes
Début
| saisie_des_données
| instructions_utilisant_les_données_lues
| communication_des_résultats
Fin
```

Ch Jaillet - URCA Info0301 – Ch1 : Algorithmique 6

6

1. « Algorithmique » ?
e. Algorithmme paramétré

❑ Fonctions et procédures

▪ autonomes et spécialisées

▪ motivation : réutilisation de blocs d'instructions

⇒ Programmation modulaire

• diviser un algorithme en sections (*modules* = sous-programmes) ; leur associer un nom ; le module est activé par l'appel de cet identifiant (= *évalué* = *appelé* = *invoqué*)

• **module**

- *prototype* = *signature* = *profil* = *en-tête*
- *arité* (nb de paramètres) : fixée

▪ Procédure : module paramétré assimilable à une instruction
=> *macro-instruction* = *macro-traitement*

▪ Fonction : module paramétré qui renvoie oblig. une valeur au module appelant
=> *macro-expression*

⇒ Appel : *paramètres effectifs*, *paramètres formels* (=muets), *association* ;
variables locales ; *portée et visibilité des éléments internes* ; ...

▪ Schémas

```
Procédure nomProc( param1 : Type1, ..., paramN : TypeN)
Variables ...
Début
| ...
Fin
```

```
Fonction nomFct( param1 : Type1, ..., paramN : TypeN) : TypeRes
Variables res : TypeRes
Début ...
| ...
| res <- expression
| retourner res
Fin
```

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

7

7

1. « Algorithmique » ?
e. Algorithmme paramétré (2)

❑ Paramètres formels des procédures : ~~un parti pris~~ **notre parti pris**

▪ Profil : `Procédure nomProc(paramètres)`

▪ Pour chaque paramètre : son nom, son type, sa **CARACTÉRISTIQUE**

	VALEUR AVANT	PENDANT	APRÈS
• Entrant (= Donnée)	valeur connue	valeur consultée	valeur d'origine
• Sortant (= Résultat)	indéterminée	renseignée pendant	valeur utilisable après
• Mixte (= Modifié)	valeur connue	modifiée pendant	nouvelle valeur utilisable

▪ Schéma :

```
Procédure nomProc(
  D d1 : D1 ; D d2 : D2 ; ... ; // les données
  R r1 : R1 ; ... ; // les résultats
  DR m1 : M1 ; ... ) // les modifiés
Variables ... // variables locales
Début
| ... // donnée D : si modifiée, modification locale
Fin
```

```
Procédure echanger2( DR a : Réel ; DR b : Réel )
Variable tmp : Réel
Début
| tmp <- a ; a <- b ; b <- tmp
Fin
```

```
Algorithme TestEch2
Variable x , y , z : Réel
Début
| Saisir( x , y , z )
| echanger2( x , y )
| Afficher( x , " ", y , " ", z )
| ...
Fin
```

• **NB** : **D** ou ↓ ; **R** ou ↑ ; **DR** ou ⇕

▪ Passage de paramètre

• **D** : par *valeur* : une valeur ou expression
PF = variable locale : modif. interne inopérante à l'extérieur

• **R / DR** : par *variable* (= par *référence*) : une variable, du type exact prévu
pas de variable créée pour le PF : c'est LA MÊME variable (indirection sur une variable EXISTANTE)

Ch Jaillet – URCA
Info0301 : Struct D. ; Prog C
Ch. 1 : Algorithmique

8

8

Chap. 1 : Algorithmique

4

Info0301 (L2 INFO)

Structures de données et programmation C

Chapitre 1 : Algorithmique

1. « Algorithmique » ?
2. Structures de contrôle : synthèse rapide
 - a. Conditionnelles
 - b. Boucles
3. Tableaux : rappels / point de vue *algo*

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

9

9

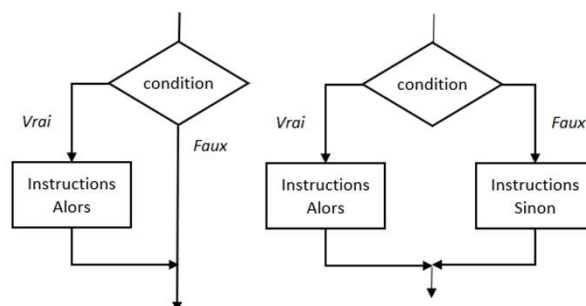
3. Structures de contrôle a. Conditionnelles

Ch Jaillet – URCA
Info0301 : Struct D. ; Prog C
Ch. 1 : Algorithmique

Juste une synthèse

- ▣ Si Alors
- ▣ Si Alors Sinon
- ▣ Si Alors Sinon Si

NB : saut en avant



10

10

3. Structures de contrôle

b. Boucles

4 types de boucles : **TantQue**, **Répéter** ; **Pour** et **Itérer**

choix :

```

graph TD
    A{Nb d'itérations connu ?} -- OUI --> B[Pour]
    A -- NON --> C{Au moins une itération ?}
    C -- OUI --> D[Répéter = Faire - TantQue]
    C -- NON --> E[TantQue]
        
```

■ **Répéter** [Jusqu'à] = **Faire** [TQ]

```

graph TD
    Start((...)) --> I[instructions]
    I --> C{condition}
    C -- faux --> I
    C -- vrai --> End((...))
        
```

Répéter
| instructions
Jusqu'à condition

■ équivalent

```

condition <- faux
TantQue Non condition Faire
| instructions
FinTantQue
        
```

■ **TantQue** [Faire]

```

TantQue condition Faire
| instructions
FinTantQue
        
```

```

graph TD
    Start((...)) --> C{condition}
    C -- vrai --> I[instructions]
    I --> C
    C -- faux --> End((...))
        
```

11

11

3. Structures de contrôle

b. Boucles

■ **Pour** [Faire]

```

Pour nomVar <- valDeb à valFin Faire
| instructions
FinPour
        
```

version généralisée (pas ≠ 0)

```

Pour nomVar <- valDeb à valFin Pas valPas Faire
| instructions
FinPour
        
```

■ **Itérer** (cas particulier)

```

Itérer n Fois
| instructions
FinItérer
        
```

■ NB : les instructions sont indépendantes du n° de l'itération

```

nomVar <- valDeb
TantQue nomVar ≤ valFin Faire
| instructions
| nomVar <- nomVar + 1
FinTantQue
        
```

```

nomVar <- valDeb
Si valPas > 0 Alors
| TantQue nomVar ≤ valFin Faire
| | instructions
| | nomVar <- nomVar + valPas
| FinTantQue
Sinon Si valPas < 0 Alors
| TantQue nomVar ≥ valFin Faire
| | instructions
| | nomVar <- nomVar + valPas
| FinTantQue
FinSi
        
```

■ **Boucles imbriquées** (ici **Pour**)

```

Pour i <- ideb à ifin Pas ipas Faire
| Pour j <- jdeb à jfin Pas jpas Faire
| | instructions // dépend de i et j (?)
| FinPour // jdeb et jfin en fonction de i ?
FinPour
        
```

12

12

Info0301 (L2 INFO)

Structures de données et programmation C

Chapitre 1 : Algorithmique

1. « Algorithmique » ?
2. Structures de contrôle
3. Tableaux : rappels / point de vue *algo*
 - a. Généralités
 - b. Passage de paramètre
 - c. Parcours : *complet* / *partiel*
 - d. Tableaux multi-dimensionnels

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

13

13


4. Tableaux (1D) : rappels / point de vue *algo* a. Généralités

- variable regroupant sous le même nom plusieurs valeurs de même type, accessibles par leur position
- [par défaut] *mono-dimensionnel* = *linéaire*
 - *taille* = *capacité* (> 0)
 - taille statique vs variable ?
 - ici statique (ne peut pas être modifiée)
 - taille logique vs taille physique (nb max de cases)
 - taille logique = *effective* : nb de cases utilisées à un moment donné
- Déclaration

```
Variable tab : Entier[8]
Constante MAX = 80
Variable message : Caractère[MAX] // taille : littéral ou expression constante
Variable valeurs : Entier[-29..30] // indices de -29 à +30 (par défaut à partir de 1)
```

- DES tableaux d'un type donné ?

```
Typedef TypeElement = Réel // par exemple
Typedef TypeTab = TypeElement[100] // pas des nouveaux types
```

- accès aux *cases* = *cellules*
 - pas de trou : *indices* successifs (= *index* / *rang* / *position*)
 - direct *via* l'indice : `tab[k]` \Rightarrow en temps constant quel que soit *k*
 \Rightarrow Attention : ne pas dépasser ! 

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

14

14

4. Tableaux (1D) : rappels / point de vue *algo*

b. Passage de paramètre

- Traitement d'un tableau = parcours + traitement des cases
 - ⇒ fonctions / procédures pour les différents traitements
 - remplissage, affichage ; calcul de la somme / moyenne / ... ; du max / indice du max ; ...
 - modifications diverses ; recherche(s) / tri(s) ; ...
- Attention : une fonction ne peut pas fournir un résultat qui soit un tableau !
 - une procédure ne peut pas fixer la taille d'un tableau
- un tableau ne connaît pas sa taille logique ⇒ la passer en paramètre
 - ... ni sa taille physique
- Passage par valeur ?
 - occasionne une recopie de la donnée
 - or un tableau contient toutes ses valeurs
 - ⇒ caractéristique DR (pas D !)
 - ⇒ **progⁿ : faire au mieux selon le langage**

```

Typedef T = ...           // par exemple Réel
Constante MAX = ...
Typedef Tableau = T[MAX]

Fonction indDuMax(DR t : Tableau ; D n : Entier)
...
    
```

```

Algorithme LetsGo
Variable tab : Tableau
Variable nb : Entier
Début
| Répéter
| | Saisie(nb)
| Jusqu'à nb > 0 Et nb <= MAX
| remplir(tab, nb) ; afficher(tab, nb)
| Afficher("max en pos. ", indDuMax(tab, nb) )
Fin
    
```

Ch Jaillet - URCA

Info0301 – Ch1 : Algorithmique

15

15

4. Tableaux (1D) : rappels / vue *algo*

c. Parcours : complet / partiel

Ch Jaillet – URCA
Info0301 : Struct D. ; Prog C
Ch. 1 : Algorithmique

- **COMPLET** : tout jusqu'à la taille effective ⇒ **Pour**
- **PARTIEL** : arrêt possible avant la fin ⇒ **TantQue**
 - v1 : sans booléen ... mais avec condition de continuation ou de sortie

```

Pour i <- 1 à n Faire
| traiter t[i]
FinPour
    
```

```

i <- 1
TantQue i <= n Et continuer(t[i]) Faire
| traiter t[i]
| i <- i + 1
FinTantQue
Si i > n Alors // ne pas tester t[i] !!
| ... // sortie au bout
Sinon
| ... // sortie en cours de route
FinSi
    
```

- v2 : avec booléen

```

i <- 1 ; stop <- faux
TantQue i <= n Et Non stop Faire
| Si fini(t[i]) Alors
| | stop <- vrai
| Sinon
| | traiter t[i]
| | i <- i + 1
| FinSi
FinTantQue
Si Non stop Alors // ne pas tester t[i] !!
| ... // sortie au bout
Sinon ... // sortie en cours de route
FinSi
    
```

- v2bis : condition inversée

```

... ; encore <- vrai
TantQue i <= n Et encore Faire
| Si fini(t[i]) Alors
| | encore <- faux
| Sinon ... // idem
| FinSi
FinTantQue
...
    
```

16

16

4. Tableaux (1D) : rappels / point de vue *algo*

b. Tableaux 2D / multi-dimensionnels

- Définition : *Tableau dont le type de base est un tableau* => « tableau de tableaux »
- dimension : Nombre d'indices utilisé pour faire référence à un des éléments
 - ≠ taille (nb de cases)
- vocabulaire mathématique :
 - algo : 1D, 2D, 3D ou +
 - Maths : *vecteur, matrice, tenseur*
- Utilisation d'un k-tableau :
 - déclaration
 - Variable `tab2` : Caractère[8, 16]
 - Variable `semaine` : Chaîne[1..7, 0..4*24-1]
 - accès indicial
 - `Tab2[5, 3]` // forme officielle
 - `Semaine[2][0]` // accepté aussi
- Parcours (tableau 2D) :
 - A. *d'une partie*
 - d'une ligne donnée
 - d'une colonne donnée
 - en diagonale
 - l'autre diagonale ?
 - B. Parcours complet
 - ligne par ligne : **1**
 - avec une seule boucle : **2a, 2b**
 - « serpent » / « spirale »
 - C. Parcours partiel
 - deux boucles / une boucle
 - avec booléen / sans
 - => 4 possibilités
 - Exemples :
 - 2 boucles, avec booléen : **3**
 - 1 boucle, sans booléen : **4**

Ch Jaillet - URCA Info0301 – Ch1 : Algorithmique 17

17

Info0301 (L2 INFO)



Structures de données et programmation C

Chapitre 1 : Algorithmique => *bilan*

1. « Algorithmie » ?
2. Structures de sélection
3. Boucles et itérations
4. Tableaux : rappels / point de vue *algo*

=> *à suivre* :

Chapitre 2 : Récursivité



Ch Jaillet + L Hollard

- URCA > UFR Sciences > Département Info
- christophe.jaillet@univ-reims.fr
- lilian.hollard@univ-reims.fr

18