Chapitre

2

Algorithme séquentiel simple

Objectif

L'objectif de ce chapitre est de présenter la structure d'un algorithme séquentiel simple avec ses composants et ses opérations de base : les variables, l'affectation, la lecture et l'écriture. Une traduction en langage C est donnée à la fin de ce chapitre pour permettre à l'étudiant d'aborder le domaine de développement en réalisant son premier programme.

Introduction

Le langage algorithmique est un langage générique permettant de traiter des problèmes par concaténation d'instructions élémentaires. Il est à la base de tous les langages de programmation. Dans ce chapitre nous allons voir comment structurer un algorithme et comment enchainer ses instructions de base pour résoudre un problème donné. Nous focalisons dans ce chapitre sur l'utilisation des variables et l'exploitation des opérations de lecture, d'écriture et d'affectation. Le vocabulaire de base du langage C est introduit également à la fin de ce chapitre pour que les étudiants puissent commencer à apprendre à programmer.

1. Notion de langage et langage algorithmique

Comme tout langage naturel, un algorithme est aussi doté d'un ensemble de règles ayant les caractéristiques suivantes :

- Il doit être fini et doit se terminer après un nombre fini d'opérations.
- Il doit être défini et précis : chaque règle (instruction) doit être définie sans ambiguïté.
- S'il y a des données d'entrée, le domaine d'application doit être précisé (exemple : nombre entier, réel, etc.).
- Il doit posséder au moins un résultat (données de sortie).
- Il doit être effectif : toutes les opérations doivent pouvoir être effectuées exactement et dans un temps fini.

Dans le langage algorithmique, on distingue trois familles de mots [7]:

- Les mots CLES
- Les mots INSTRUCTIONS
- Les mots DELIMITEURS

1.1. Les mots clés

Les mots clés définissent la structure algorithmique utilisée. Exemples de mots clés :

- SI … ALORS… SINON… : définissent un structure ALTERNATIVE

- REPETER… JUSQU'A… : définissent une structure ITERATIVE

1.2. Les mots instructions

Ce sont des verbes d'action qui caractérisent la nature des opérations à effectuer sur une ou plusieurs données. Exemples de mots instructions : LIRE (variable 1)

1.3. Les mots délimiteurs

Les mots délimiteurs fixent :

- Les bornes d'ENTRÉE et de SORTIE de l'algorithme
- Les bornes d'ENTRÉE et de SORTIE des différentes structures utilisées dans l'algorithme si ces bornes ne sont pas définies par la structure elle-même.

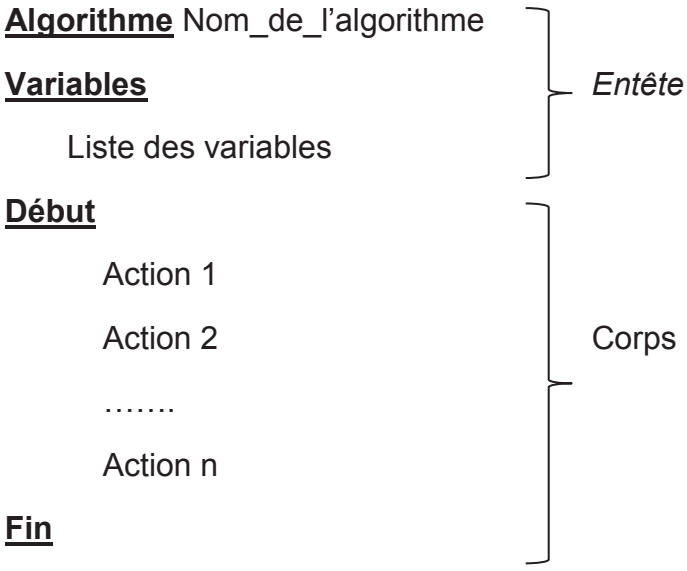
DEBUT et FIN sont les seuls mots délimiteurs et peuvent être suivi éventuellement d'un mot clé.

Exemple : FIN SI

2. Parties d'un algorithme

Un algorithme se compose de deux grandes parties :

- Les informations dont on a besoin au départ (Entête) ;
- La succession d'instructions à appliquer (Corps)



La succession d'instructions n'est pas toujours détaillée explicitement au sein de l'algorithme, mais parfois dans une autre partie, à travers ce que l'on appelle des fonctions ou des procédures. Cela permet de découper l'algorithme en plusieurs sous-algorithmes et de le rendre plus facile à comprendre.

3. Les données : variables et constantes

Dans un programme informatique, on a toujours besoin de stocker provisoirement des valeurs. Il peut s'agir de données stockées sur un support (disque dur, CD, diskette..) ou fournies par l'utilisateur (frappées au clavier). Il peut aussi s'agir de résultats obtenus par le programme, intermédiaires ou définitifs. Ces données peuvent être des nombres, du texte, etc. donc quand on a besoin de ces données au cours d'un programme, on utilise des ***variables*** pour les stocker, ou bien elles restent constantes, dans ce cas on utilise des ***constantes*** pour stocker ces données.

3.1. Les variables

Une variable est un triplet composé [8] :

- D'un type qui caractérise l'ensemble des valeurs que peut prendre la variable :
    - Nombre Entier
    - Nombre flottant (Réel)
    - Booléen (uniquement valeurs Vrai ou Faux)
    - Caractère (alphabétique, numérique)
    - Chaine de caractères (mot ou phrase)

A un type donné, correspond un ensemble d'opérations définies pour ce type.

- D'un nom (a priori toute chaîne alphanumérique),
- D'une valeur.

Dans un programme, les données sont manipulées via des variables

- Une variable est une case mémoire
- Une variable est désignée par un nom (identifiant)
- Une variable a un type de donnée (implicite dans certains langages)
- Une variable contient une valeur du type et cette valeur peut varier

Le cycle de vie d'une variable est caractérisé par :

- Déclaration de la variable (nom et type)
- Affectations de valeurs à la variable
- Suppression de la variable (souvent automatique)

3.1.1. Déclaration d'une variable

La déclaration de variables est très simple, elle se fait de la façon suivante:

variable Nom_de_ma_variable: type
Nom_autre_variable: autre_type

3.1.2. Les identificateurs

On appel identificateurs les noms des variables (et des fonctions). Ils sont composés d'une suite de lettres ***non accentuées*** et de chiffres. Un identificateur doit être significatif pour faciliter la compréhension du programme. Le premier caractère doit être une lettre. Le symbole '_' est aussi considéré comme une lettre. L'ensemble des symboles utilisables est donc: {0,1,2,...,9,A,B,...,Z,_,a,b,...,z}

REM :

- En langage C, Le premier caractère doit être une lettre (ou le symbole '_')
- C distingue les majuscules et les minuscules, ainsi:
- 'Nom_de_variable' est différent de 'nom_de_variable'
- La longueur des identificateurs n'est pas limitée, mais C distingue 'seulement' les 31 premiers caractères.

4. Type de données

Un type caractérise les valeurs que peut prendre une variable. Il définit également les opérations, généralement appelées opérateurs, qui pourront être appliquées sur les données de ce type.

Les types ont deux intérêts principaux :

– Permettre de vérifier automatiquement (par le compilateur) la cohérence de certaines opérations. Par exemple, une valeur définie comme entière ne pourra pas recevoir une valeur chaîne de caractères.

– Connaître la place nécessaire pour stocker la valeur de la variable. Ceci est géré par le compilateur du langage de programmation considéré et est en général transparent pour le programmeur.

En algorithmique les types de données de base sont : l'entier, le réel, le booléen, le caractère et la chaine de caractère. Ci-dessous un petit récapitulatif des types de données :

Type	Utilité	Exemple
Booléen	Représente un état binaire, vrai ou faux	`Vrai, Faux`
Entier	Représente un nombre entier	`215`
Réel	Représente un nombre réel	`163.254`
Caractère	Représente un caractère unique (comme une touche du clavier)	`'c','B', '7', '*'`
Chaîne de caractères	Représente un texte	`"Bonjour, je m'appelle Ali »`

Tableau 2. Types de données fondamentaux

5. Opérations de base

Pour pouvoir comprendre et utiliser correctement les opérations de base en algorithmique, on doit tout d'abord aborder les notions d'opérateur, d'opérande et d'expression.

5.1. Opérateurs

À chaque type est associé un ensemble d'opérations (ou opérateurs). Un opérateur est un symbole d'opération qui permet d'agir sur des variables ou de faire des "calculs".

5.1.1. Opérateurs numériques

Ce sont les quatre opérations arithmétiques tout ce qu'il y a de classique.

+ : addition

- : soustraction

* : multiplication

/ : Division

Avec en plus pour les entiers **div** et **mod**, qui permettent respectivement de calculer une division entière et le reste de cette division. Mentionnons également le ^ qui signifie « puissance ». **45 au carré** s'écrira donc **45 ^ 2**.

Enfin, on a le droit d'utiliser les parenthèses, avec les mêmes règles qu'en mathématiques. La multiplication et la division ont « naturellement » priorité sur l'addition

et la soustraction. Les parenthèses ne sont ainsi utiles que pour modifier cette priorité naturelle.

5.1.2. Opérateurs logiques (ou booléens)

Il s'agit du ET, du OU, du NON et du Ou Exclusif XOR.

5.2. Opérandes

Un opérande est une entité (variable, constante ou expression) utilisée par un opérateur dans une expression.

5.3. Expression

Une expression est une combinaison d'opérateur(s) et d'opérande(s), elle est évaluée durant l'exécution de l'algorithme, et possède une valeur (son interprétation) et un type.

Exemple :

Dans a + b

a est l'opérande gauche

+ est l'opérateur

b est l'opérande droit

a + b est appelé une expression

Si par exemple a vaut 2 et b 3, l'expression a + b vaut 5

Si par exemple a et b sont des entiers, l'expression a + b est un entier

Attention : Pour qu'une expression soit acceptable, il est nécessaire que les types des opérandes d'un opérateur soient compatibles. Par exemple, faire l'addition d'un entier et d'un booléen n'a pas de sens. De même, on ne peut appliquer les opérateurs logiques que sur des expressions booléennes.

6. Instructions de base

Une instruction est une action élémentaire commandant à la machine un calcul, ou une communication avec l'un de ses périphériques d'entrées ou de sorties. Les instructions de base sont : l'affectation et les instructions d'entrée sorties.

6.1. Affectation

C'est l'action de charger une valeur dans une variable. Cette valeur peut elle-même être une variable, le résultat d'une expression arithmétique ou logique ou une constante [9].

Syntaxe

Affectation d'une valeur à une variable : **Nom-variable ← valeur ;**

Affectation d'une variable à une variable : **Nom-variable1 ← Nom-variable2 ;**

Affectation du résultat de calcul à une variable :

Nom-variable ← nom-variable1 opérateur nom-variable2 ;

Ou encore l'affectation du résultat de plusieurs calcul à une variable :

Nom-variable ← nom-variable1 opérateur1 nom-variable2 … opérateur-n nom-variable n ;

Exemple

La résolution d'une équation de second degré se fait par calcul du discriminant delta :

Delta ← b*b – 4*a*c ;

Dans le cas de delta > 0 la machine réalise le calcul des deux solutions X_1 et X_2 en exécutant les instructions :

$X_1 \leftarrow -b + \sqrt{delta} / 2*a$;

$X_2 \leftarrow -b - \sqrt{delta} / 2*a$;

Remarque

6.2. Instructions d'entrée sorties

L'affectation ne vide pas la variable émettrice (à notre droite) de sa valeur. Par contre, le contenu de la variable réceptrice est écrasé.

Au moment de l'exécution de l'algorithme, l'utilisateur et la machine ont besoin d'échanger des données ou des informations, on parle dans ce cas des instructions d'entrées (lecture) et de sorties (écriture). Lors du fonctionnement de l'algorithme et quand on arrive à une instruction de lecture ce dernier s'arrête à cette instruction et ne se poursuit que lorsque l'utilisateur a entré une valeur, et l'algorithme de sa part, affiche à l'utilisateur les résultats de son travail à travers les instructions d'écriture.

6.2.1. La lecture

L'ordre LIRE permet à l'ordinateur d'acquérir des données à partir de l'utilisateur, dans des cases mémoire bien définies (qui sont les variables déclarées).

Rappel

Les variables sont des cases mémoire, supposées contenir un type de données, nommées par le nom de variable.

Syntaxe

LIRE (variable1, [[variable2] …])

Remarque :

1. La saisie se fait uniquement dans des variables. Ce sont les cases (cellules) qui pourront accueillir les données correspondantes.

2. La donnée à introduire doit être de même type que la variable réceptrice.

6.2.2. L'écriture

Cette action permet de communiquer un résultat ou un message sur écran ou sur imprimante pour l'utilisateur.

Syntaxe

ECRIRE (paramètre1 [[,paramètre2]…])

Paramètre = variable | expression | constante

Constante = nombre | message

Exemples :

ECRIRE (" La valeur de 3*2 est égale à ", 3*2)

ECRIRE (" La moyenne est = ", MOY)

7. Construction d'un algorithme simple

Un algorithme se présente en général sous la forme suivante :

Déclaration des variables : On décrit dans le détail les éléments que l'on va utiliser dans l'algorithme (variables, constantes et structures),

Initialisation ou Entrée des données : On récupère les données et/ou on les initialise (par lecture et par affectation),

Traitement des données : On effectue les opérations nécessaires pour répondre au problème posé,

Sortie : On affiche les résultats (par l'écriture)

On rappelle que la structure d'un algorithme subit la forme suivante :

Algorithme Nom_de_l'algorithme
Variables — *Entête*
Liste des variables
Début
Action 1
Action 2
…….
Action n — Corps
Fin

Où la déclaration des variables se fait dans l'entête de l'algorithme et les autres tâches (initialisation, traitement et affichage) se font dans le corps de l'algorithme.

Exemple : calcul de la somme de deux entiers

Algorithme calcul_somme

Variables

A, B, Somme : entier

Début

Lire (A)

Lire (B)

Somme ← A+B

Ecrire (Somme)

Fin

8. Représentation d'un algorithme par un organigramme

Un **organigramme** (parfois appelé **algorigramme**) est une représentation graphique d'un algorithme. En général, on peut représenter un algorithme sous forme structurée ou sous forme graphique. Les opérations dans un organigramme sont représentées par les symboles dont les formes sont normalisées. Ces symboles sont reliés entre eux par des lignes fléchées qui indiquent le chemin. C'est ainsi qu'on a [10]:

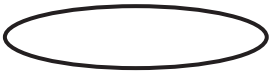
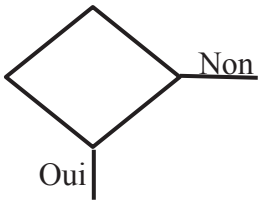
	Début et fin d'un organigramme : on l'utilise pour commencer et pour terminer un algorithme.
	Entrée/ Sortie : on l'utilise pour lire ou écrire des données.
	Traitement : on l'utilise pour le reste des opérations hors la lecture et l'écriture.
	Choix avec condition : on l'utilise pour l'exploitation de conditions variables impliquant le choix d'une voie parmi plusieurs. Symbole couramment utilisé pour représenter une décision ou un aiguillage

Tableau 3. Symboles d'un organigramme

Exemple :

L'algorithme de l'exemple précédent (somme de deux entiers) peut être représenté sous l'organigramme suivant :

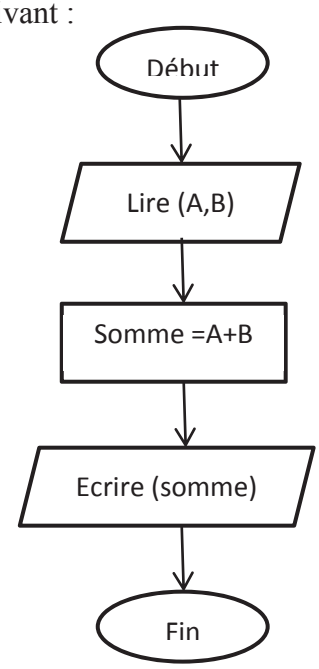


Figure 2. Structure d'un organigramme

9. Traduction en C

C est un langage de programmation impératif généraliste. Inventé au début des années 1970 pour réécrire UNIX, C est devenu un des langages les plus utilisés. De

nombreux langages plus modernes comme C++, C#, Java et PHP ont une syntaxe similaire au C et reprennent en parti sa logique.

9.1. Pourquoi le langage C

Il existe de nombreux langages de programmation de haut niveau comme le C, le Pascal, ou le Basic. Ils sont tous excellents et conviennent pour la plupart des tâches de programmation. Toutefois, les professionnels placent le langage C en tête de liste pour plusieurs raisons [11] :

- Il est souple et puissant. Le langage C est utilisé pour des projets aussi variés que des systèmes d'exploitation, des traitements de textes, des graphiques, des tableurs ou même des compilateurs pour d'autres langages ;
- Disponible pour tous les types de processeurs et de systèmes d'exploitation
- Le langage C contient peu de mots. Une poignée d'expressions appelées mots clés servent de bases pour l'élaboration des fonctions ;
- Il a influencé de nombreux langages plus récents dont C++, Java, C# et PHP ; sa syntaxe en particulier est largement reprise ;
- Le langage C est modulaire. Son code peut (et devrait) être écrit sous forme de sous-programmes appelés fonctions. Ces fonctions peuvent être réutilisées pour d'autres applications ou programmes.

9.2. Structure d'un programme C

Un programme en C se présente de la façon suivante :

[Directives au préprocesseur]
[Déclaration des variables externes]
[Fonctions secondaires]

} ***Partie1 : déclaration***

int main ()
{
Déclaration des variables internes
Instructions
}

} ***Partie 2 : Corps du programme***

La partie ***des déclarations*** comporte la déclaration des fonctions des bibliothèques (bibliothèque standard ou autre) par inclusion de fichiers fournis avec le langage et peut

comprendre des déclarations des variables « globales ». Elle peut contenir aussi des définitions de fonctions qui seront utilisées par l'intermédiaire de la fonction principale **main**.

Les variables internes et les instructions constituent le corps du programme. Elles sont plus ou moins complexes et nombreuses selon les programmes.

Remarque :

Parmi les bibliothèques standards fréquemment utilisées en C nous pouvons citer :

Stdio.h : Il s'agit des fonctions d'une librairie standard utilisée avec les unités classiques d'entrées-sorties, qui sont respectivement le clavier et l'écran. L'appel à la librairie *stdio.h* se fait par la directive au préprocesseur : *# include <stdio.h>*

math.h : Il s'agit un groupe de fonctions de la bibliothèque standard du C qui permet d'utiliser un ensemble de fonctions mathématiques de base. L'appel à la librairie math.h *se fait par la directive au préprocesseur : # include <math.h>*

9.3. Identificateurs

Le rôle d'un identificateur est de donner un nom à une entité du programme. Plus précisément, un identificateur peut désigner :

– un nom de variable ou de fonction,

– un type défini par typedef, struct, union ou enum,

– une étiquette.

En langage C et lors de la déclaration d'un identificateur, nous devons respecter les spécificités suivantes :

- Les caractères acceptés pour d'déclarer un identificateur sont uniquement les chiffres, les lettres non accentuées et le blanc souligné (_).
- Le premier caractère doit être une lettre (ou un blanc souligné symbole '_'). Par exemple, var1, X2 ou _alpha sont des identificateurs valides ; par contre, 8C, *var ne le sont pas ;
- C distingue les majuscules et les minuscules, ainsi: **`'Nom_de_variable'`** est différent de **`'nom_de_variable'`**
- La longueur des identificateurs n'est pas limitée, mais C distingue 'seulement' les 31 premiers caractères.

9.4. Les types prédéfinis

Le C est un langage *typé*. Cela signifie en particulier que toute variable, constante ou Fonction est d'un type précis. Le type d'un objet définit la façon dont il est représenté en mémoire. Le tableau ci-dessous englobe les types les plus utilisés en langage C avec leurs spécificités:

définition	*description*	*domaine min*	*domaine max*	*nombre d'octets*
char	caractère	-128	127	1
short	entier court	-32768	32767	2
int	entier standard	-32768	32767	2
long	entier long	-2147483648	2147483647	4
unsigned char	caractère	0	255	1
unsigned short	entier court	0	65535	2
unsigned int	entier standard	0	65535	2
unsigned long	entier long	0	4294967295	4

Tableau 4. Type entier en langage C

définition	*précision*	*mantisse*	*domaine min*	*domaine max*	*nombre d'octets*
float	simple	6	$3.4 * 10^{-38}$	$3.4 * 10^{38}$	4
double	double	15	$1.7 * 10^{-308}$	$1.7 * 10^{308}$	8
long double	suppl.	19	$3.4 * 10^{-4932}$	$1.1 * 10^{4932}$	10

Tableau 5. Type réel en langage C

Pour déclarer une variable en C, on utilise l'un des spécificateurs de type du tableau précédent avec un identificateur.

Exemple : int Var1 ; (Var1 est une variable de type entier)

float Y ; (Y est une variable de type réel)

char W ; (W est une variable de type caractère)

REM : En C il n'existe pas de type spécial pour variables booléennes. Tous les types de variables numériques peuvent être utilisés pour exprimer des opérations logiques

9.5. Les constantes

La déclaration des constantes en langage C se fait de la manière suivante :

const type nom_constante = valeur ;

Exemple

```
const float Pi=3.14159;
const int NbrCoins=4;
```

9.6. Affectation

En langage C, l'opérateur d'affectation est "=" il signifie "affecter à". Sa Syntaxe est la suivante:

<Destination> = <Source>; Où <Source> et <Destination> doivent être de même nature et où <Source> est une expression et <Destination> une variable ;

Exemple

int x, y ;

x=5 ; //x=5

y=3 ; //y=3

x=y ; //x=3 (valeur de y)

9.7. Lecture

La lecture en langage C s'effectue avec l'utilisation de la fonction ***scanf.*** Il s'agit d'une fonction de la librairie standard ***stdio.h*** utilisée avec les unités classiques d'entrées-sorties, qui sont respectivement le clavier et l'écran.

La fonction scanf permet de saisir des données au clavier et de les stocker aux adresses spécifiées par les arguments de la fonction. La syntaxe de scanf est la suivante :

Scanf ("chaîne de contrôle", argument-1,..., argument-n)

La chaîne de contrôle indique le format dans lequel les données lues sont converties. Le tableau ci-dessous indique les Formats de saisie pour la fonction scanf les plus utilisés [12].

Format	Type d'objet pointé	Représentation de la donnée saisie
%d %hd % ld	int short int long int	décimale signé décimale signé décimale signé
%u %hu % lu	unsigned int unsigned short int unsigned long int	décimale non signé décimale non signé décimale non signé
%o %ho %lo	int short int long int	octale octale octale

%x %hx % lx	int short int long int	hexadécimale hexadécimale hexadécimale
%f % lf % Lf	float double long double long double	flottante virgule fixe flottante virgule fixe flottante virgule fixe
%e %le % Le	float double long double	flottante notation exponentielle flottante notation exponentielle flottante notation exponentielle
%g % lg % Lg	float double long double	flottante virgule fixe ou notation exponentielle flottante virgule fixe ou notation exponentielle flottante virgule fixe ou notation exponentielle
%c	char	caractère
%s	char*	Chaîne de caractères

Tableau 6. Formats de saisie pour la fonction scanf

Les données à entrer au clavier doivent être séparées par des blancs ou des <RETURN> sauf s'il s'agit de caractères. On peut toutefois fixer le nombre de caractères de la donnée à lire. Par exemple **%3s** pour une chaîne de 3 caractères, %10d pour un entier qui s'étend sur 10 chiffres.

Exemple :

```
#include <stdio.h>
main()
{
int i;
printf("entrez un entier sous forme hexadécimale i = ");
scanf("%x", &i);
printf("i = %d\n", i);
}
```

Si on entre au clavier la valeur 1a, le programme affiche i = 26.

9.8. Ecriture

L'écriture en C s'effectue à travers la fonction ***printf***. Cette dernière est une fonction d'impression formatée de la librairie standard ***stdio.h***, ce qui signifie que les données sont converties selon le format particulier choisi. Sa syntaxe est :

printf ("chaîne de contrôle ", expression-1, ..., expression-n);

La *chaîne de contrôle* contient le texte à afficher et les spécifications de format correspondant à chaque expression de la liste. Les spécifications de format sont introduites par le caractère %, suivi d'un caractère désignant le format d'impression (voir tableau 2) [12].

Format	Conversion en	Ecriture
%d	int	décimal signé
%ld	long int	décimale signée
%u	unsigned int	décimale non signée
%lu	unsigned long int	décimale non signée
%o	unsigned int	octale non signée
%lo	unsigned long int	octale non signée
%x	unsigned int	hexadécimale non signée
%lx	unsigned long int	hexadécimale non signée
%f	double	décimale virgule fixe
%lf	long double	décimale virgule fixe
%e	double	décimale notation exponentielle
%le	long double	décimale notation exponentielle
%g	double	décimale, représentation la plus courte parmi %f et %e
%lg	long double	décimale, représentation la plus courte parmi %lf et %le
%c	unsigned char	caractère
%s	char*	chaîne de caractères

Tableau 7. Formats d'impression pour la fonction printf

On peut éventuellement préciser certains paramètres du format d'impression, qui sont spécifies entre le % et le caractère de conversion dans l'ordre suivant :

– largeur minimale du champ d'impression : %10d spécifie qu'au moins 10 caractères seront réservés pour imprimer l'entier. Par défaut, la donnée sera cadrée à droite du champ. Le signe - avant le format signifie que la donnée sera cadrée à gauche du champ (%-10d).

– précision : %.2f signifie qu'un flottant sera imprimé avec 2 chiffres après la virgule. Lorsque la précision n'est pas spécifiée, elle correspond par défaut à 6 chiffres après la virgule. Pour une chaîne de caractères, la précision correspond au nombre de caractères imprimés : %30.4s signifie que l'on réserve un champ de 30 caractères pour imprimer la chaîne mais que seulement les 4 premiers caractères seront imprimés (suivis de 26 blancs).

Exemple

L'exemple ci-dessous calcule la surface et la circonférence d'un cercle. Dans cet exemple on exploite les variables, les constantes, l'affectation, la lecture et l'écriture.

```
#include <stdio.h>
#include <math.h>
main()
{
const float Pi=3.14159;
float R,S,C ;
printf  (" Entrez le rayon du cercle: ");
scanf  ("%f",&R);
S=Pi* (pow (R,2));
C=2*Pi*R;
printf ("La surface de cette cercle = %f \n",S);
printf ("La circonference du cercle = %f\n",C);
}
```

Conclusion

Nous avons présenté à travers ce chapitre les premiers pas pour construire un algorithme séquentiel simple. Par conséquent, les notions de variables, d'affectation, de lecture et d'écriture sont présentées algorithmiquement et techniquement avec le langage de programmation C.