COURS

ALGORITHMIQUE & PROGRAMMATION C

CHAPITRE

LECTURE ET ECRITURE EN LANGAGE C

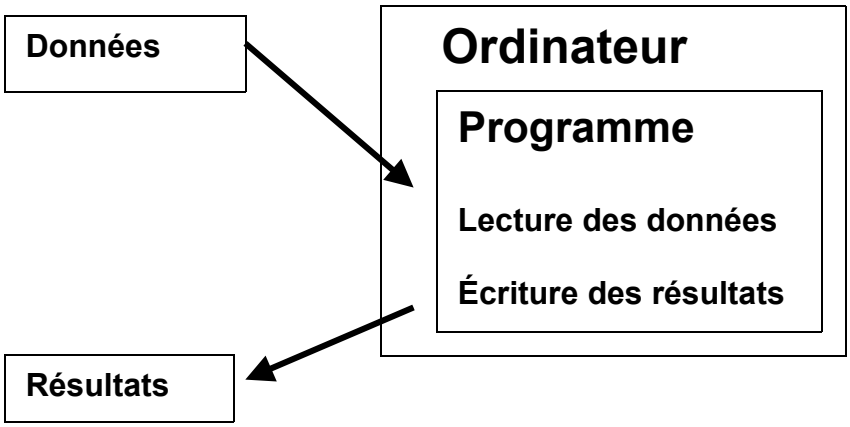
1. INTRODUCTION

L'instruction de **lecture** permet de fournir des informations (**données**) à notre programme par l'intermédiaire d'un périphérique (clavier).

L'instruction d'**écriture** permet à un programme de communiquer des informations (**résultats**) par l'intermédiaire d'un périphérique (écran).

La bibliothèque standard <**stdio**> contient un ensemble de fonctions permettant la communication de l'ordinateur avec le monde extérieur.

Les principales fonctions de lecture et d'écriture en langage C sont **scanf** et **printf**. Ces deux fonctions font parties de la bibliothèque standard <**stdio**>.



2. ECRITURE DES DONNÉES

2.1. La fonction *printf*

La fonction, formatée d'écriture de données, **printf** permet de transférer du texte, des valeurs de variables ou des résultats d'expressions vers l'écran. Elle exige l'utilisation de formats de sortie.

2.2. Syntaxe

printf ("<format> ", <expr1>, <expr2>, ...);

- ✓ <format>: texte, séquence d'échappement, spécificateur de format
- ✓ Autant de spécificateurs de formats que d'expressions
- ✓ Spécificateur de format avec : **%caractère_du_ type** (%d, %f, …)

2.3. Spécificateurs de format

Voici quelques Spécificateurs de format:

%d : entier
%c : caractère
%f : rationnel en notation décimale
%s : chaîne de caractère
…

Exemples:

- ✓ printf("Bonjour\n");
- ✓ x=100 ; y=x ;
  printf("La valeur de y est %d\n", y);
  printf("La somme = %d\n", x+y);
- ✓ moyenne=12.3333 ;
  printf("La moyenne est %.2f\n", moyenne);
- ✓ c='A' ;
  printf("Le caractère %c a pour valeur %d", c,c);
  va afficher sur l'écran:
  Le caractère A a le code 65 !
  La valeur de **c** est donc affichée sous deux formats différents.

3. LECTURE DES DONNÉES

3.1. La fonction *scanf*

La fonction, formatée de lecture de données, **scanf** permet de lire à partir du clavier des données.
Les variables à saisir sont formatées, le nom de la variable est précédé du symbole **&** désignant l'adresse de la variable. La saisie s'arrête avec "RETURN" (c'est à dire taper entré), les éléments saisis s'affichent à l'écran.

3.2. Syntaxe

scanf ("<format> ", <Adrv1>, <Adrv2>, …);

- ✓ <format>: format de lecture des données
- ✓ Autant de format que de données à lire
- ✓ <Adrv>: &NomVariable

- La chaîne de format détermine comment les données reçues doivent être interprétées.
- Les données reçues correctement sont mémorisées successivement aux ***adresses*** indiquées par **<AdrV1>,…** .

– ***L'adresse d'une variable*** est indiquée par le nom de la variable précédé du signe **&**.

Exemples:

- ✓ char alpha;
  int i ;
  float r;
  scanf("%c",&alpha); /* saisie d'un caractère */
  scanf("%d",&i); /* saisie d'un entier en décimal */
  scanf("%x",&i); /* saisie d'un entier en hexadécimal*/
  scanf("%f",&r); /* saisie d'un réel */

- ✓ int jour, mois, annee;
  scanf("%d %d %d", &jour, &mois, &annee);
  /*lit trois entiers relatifs, séparés par des espaces, tabulations ou interlignes. Les valeurs sont attribuées respectivement aux trois variables **JOUR**, **MOIS** et **ANNEE**.*/

REMARQUE :

- ✓ Si l'utilisateur ne respecte pas le format indiqué dans **scanf**, la saisie est ignorée. Aucune erreur n'est générée.
  Exemple:
  char alpha;
  scanf("%d",&alpha);
  - Si l'utilisateur saisie 97 tout va bien, alpha devient le caractère dont le code ASCII vaut 97.
  - Si l'utilisateur saisie **a**, sa saisie est ignorée.
- ✓ **scanf** retourne comme résultat le nombre de données correctement reçues (type **int**).