REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de M'sila
Faculté des Mathématiques et de l'Informatique
Département d'informatique

جامعة المسيلة
كلية الرياضيات والإعلام الآلي
قسم الإعلام الآلي

Chapitre 2:
L'algorithme séquentiel simple

Algorithmique et structure de données

Présenté par: Dr. Benazi Makhlouf
Année universitaire: 2022/2023

Contenu du chapitre 02:

1. Introduction
2. Notion de langage
   - Langage
   - Les commentaires
   - Mots réservés
   - L'identificateur
3. La structure d'un algorithme
4. Les Données:
   - Les valeurs
   - Les types de données
   - Les constantes
   - Les variables
5. Instructions
   - Opérations de base
   - Expression
   - Instruction et Bloc d'instructions
   - Instruction d'affectations
   - Instruction d'entrée: lire()
   - Instruction de sortie: écrire()
6. Un algorithme simple
7. Algorigramme
8. Traduction en langage C
   - Le préprocesseur
   - Les types
   - Déclaration des variables
   - Déclaration des constantes
   - L'affectation
   - Entrée: printf
   - sortie: scanf
   - Structure d'un programme en C

1. Introduction

Lorsque l'ordinateur a été inventé, il fonctionnait à l'aide des tubes à vides, et sa programmation se faisait en entrant des zéros (0) et des uns (1).

Avec la création du transistor et des circuits intégrés, sa taille est réduite et ses capacités ont augmenté. Des langages de programmation de haut niveau ont été créés pour rendre la tache de programmation plus facile.

Il existe trois grandes façons de représenter un algorithme :

1. Pseudocode: Décrire l'algorithme dans des langages proches aux langages naturels
2. Organigramme: représenter l'algorithme a l'aide des flèches et des formes géométriques
3. langages de programmation: écrire directement dans un langage de programmation.

2. Notion de langage

Langage

C'est la façon dont un ordinateur comprend les commandes humaines. Il se compose de caractères, de vocabulaire, de règles grammaticales et de significations.

langage de programmation:

C'est un langage intermédiaire entre le langage humain et le langage machine. Et qui nous fournit un cadre pour développer des programmes qu'un ordinateur peut exécuter.

Il permet de décrire les structures de données, et les opérations qui seront réalisées,

les types de langages de programmation

Interprété: Si le programme est lu et exécuté instruction par instruction par un interpréteur. Tels que: Matlab et les langages du Web.

compilé: si le programme est entièrement traduit en langage machine par un Compilateur avant son exécution. Tels que: C

Environnement de développement intégré (IDE)

un ensemble de programmes qui fournissent tous les outils nécessaires au processus de développement des applications c'est-à-dire concevoir, développer, tester, déboguer et déployer des applications.

- Éditeur de code:
- Gestionnaire de projet:
- Débogueur:
- Raccourcis pour compiler et exécuter le programme

exemple: Dev-C++, Embarcadero, visual studio...

Les commentaires

Ce sont des textes qui sont ignorés lors de la compilation et qui ne font pas partie du programme.

Il existe deux types:

- Les commentaires d'un seul ligne: il commence par // (deux slashs) et se termine par la fin de la ligne.
- Les commentaires de plusieurs lignes: il commence par /* (slash étoile) et se termine par */ (étoile slash).

exemple:

```
// Commentaire d'une seule ligne
/* Commentaire
multiligne*/
```

les mots réservés

Ce sont des mots qui ont une signification préexistante pour le langage de programmation et qui ne peuvent pas être utilisés pour identifier de nouveaux éléments.

exemple: dans l'algorithmique "algorithme", "début" , "fin", "et", "si"

dans C: "if" et "while".

L’identificateur

C’est le nom que le programmeur donne à tout élément de l’algorithme qu’il veut créer. Tels que: nom de l’algorithme, variable, type, constante, fonction …

Règles pour les identificateurs:

1. Se compose de A-Z , a-z, 0-9 ou _
   - Doit être un seul mot (pas d’espace)
   - Ne contient pas des caractères spéciaux ;:!/.?§…
2. Ne commence pas par un chiffre
3. Ne doit pas être un mot réservé.
4. Doit être unique, pas possible de définir plus d'un élément avec le même nom.

N.B. Il est recommandé d'utiliser des noms significatifs

Exemples

identificateurs valides: x, pi, Mat_info, estVide ,n5, _debut, _0a

identificateurs non valides: α, ƨ, π, élève, 3a, Mat info, debut (algo), if (C)

3. La structure d'un algorithme

Un algorithme est constitué de données et d'instructions, il est très similaire à une recette de cuisine. Il se constitué d' en-tête, de déclarations et d' instructions, et il prend la forme suivante:

```
Algorithme nom
  Déclaration
Début
  instructions
Fin
```

- **en-tête**: composé du mot Algorithme, suivi du nom qui explique le problème à résoudre. Il doit s'agir d'un identifiant valide
- **Déclarations**: des places sont réservées en mémoire pour des données (constantes et variables)
- **Instructions**: un ensemble de commandes qui seront exécutées pour résoudre le problème Il commence par **Début** et se termine par **Fin**.

Les Données:

- Les valeurs
- Les constantes
- Les variables
- Les types de données

Les valeurs

- Des nombres comme: 2, -7, 3.12, 6.2e-7
- Des caractères toujours entre guillemets simples comme: ‘k’, ‘1’, ‘?’, ‘ ‘, ‘\n’ , ‘خ’
- Des chaînes de caractères toujours entre guillemets doubles comme: "azety", "السلام عليكم", ""
- vrai, faux

Types de donnés

Type: est le domaine auquel appartiennent les données. tel que les nombres, textes, images, audio ou vidéo.

Le type détermine comment les bits sont traduits et la taille à réserver en mémoire, c'est-à-dire le nombre de bits, et les opérations autorisées.

Il existe 5 types de base dans l'algorithme:

1. Entier, Integer:-5, 0, 1, 13
2. Réel, Real: -7, 0, 1. , 3.14, 2.7e03
3. Booléen, Boolean ne contient que 2 valeurs: vrai ou faux.
4. Caractère, Character: Il comprend toutes les icônes sur le clavier: chiffres, lettres dans toutes les langues et symboles imprimés (visuels) et non imprimés. : 'a', 'M', '1', '+', ',', 'س'.
5. Chaines de caractères, String: Un ensemble de caractères, de longueur 0 ou plus, toujours entre guillemets doubles: "informatique", "Bonne chance\n", "1", "3.14".

constantes

Une valeur (chiffres ou symboles) qui a un nom et ne peut pas être modifiée pendant l'exécution du programme.

Déclaration:

```
Const Identificateur=valeur
```

Const ou **Constante**: Ce sont deux mots réservés.

identificateur: est le nom qu'on donne a la constante.

exemple:

```
Const
    Pi =3.1415926
    DEP  = "قسم الاعلام الالي"
```

Avantages des constantes

- Code rétracté: remplacer une grande phrase par un petit mot. P au lieu de 3.1415926.
- Évitez les erreurs: un nom significatif. Par exemple: PI au lieu de 3,1415926.
- Simplifiez la maintenance: la valeur ne soit modifiée qu'à un seul endroit.

Les variables

Emplacement dans la mémoire utilisé pour stocker des données. Il a un nom, un type et une valeur

- **Nom**: identifiant utilisé par le programmeur pour manipuler la variable. **Ex**: poids.
- **Type**: **Ex**: entier, réel, caractère, chaine
- **Valeur**: c'est la chose qui change pendant l'exécution **Ex**: 5,'v', "azrty"

Déclaration des variables:

```
Var Identificateur: Type
```

- **Var** ou **Variable**: Ce sont deux mots réservés qui permettent la déclaration de variables.
- Identificateur: Le nom qu'on donne à la variable.
- Type: Le type de la variable.

Exemples:

```
Var
    age: entier
    x, y, z: réel
```

5. Instructions

- Opérations de base
- Expression
- Instruction et Bloc d'instructions
- Instruction d’affectations
- Instruction d’entrée: lire()
- Instruction de sortie: écrire()

Opérations de base

Opérations arithmétiques

Op	C	observation	exemple
- +	- +	signe	+3 -7 -a
- +	- +	Les deux opérandes entiers, Le résultat est un entier. l'un est réel, résultat est un nombre réel.	5+3.0
*	*	Pour le produit, comme l'addition et la soustraction.	5*3 entier
/	/	pour la division réelle. Le résultat est toujours un nombre réel	5/3 ou 5.0/3 en C
mod	%	Pour calculer le reste de la division. Les deux opérandes sont entiers Le résultat est toujours un entier.	5 mod 3 ou 5%3 en C
div	/	Pour calculer le quotient (division entière). Comme le reste	5 div 3 ou 5/3 en C
^		Pour calculer la puissance, en C, on utilise la fonction pow(), le résultat est un nombre réel	5^2 ou pow(5,2) en C
√		Pour calculer la racine, en C, on utilise la fonction sqrt(), le résultat est un nombre réel	√5 ou sqrt(5) en C

Opérations de base

Opérateurs relationnels

Opération	C	observation
>, >=, <, <=		les même en C
=	==	En C « == » deux fois = est lu égal, tandis que « = » est utilisé pour l'affectation et lu reçoit.
≠	!=	lu défirent de

Observation:

Le résultat de la comparaison est toujours de type logique booléen, c'est-à-dire vrai ou faux

Opérations de base

Opérateurs booléens

Opération		C	observation
non	négation	!	vrai si l'opérande est faux. et faux si vrai.
et		&&	vrai si les deux opérandes sont vrais, sinon faux
ou		||	fauxsi les deux opérandes sont faux, sinon vrai
xou	ou exclusif		vrai si l'un est vrai et l'autre est faux , sinon faux
=	équivalence	==	vrai si sont égaux.

Opérations de base

Opérateurs de chaîne

+ est utilisé pour concaténer deux chaines de caractère.

Exemple: "bonne" + " " + "Chance" donne "bonne Chance«

la priorité

priorité	l'opération
0	()
1	+ et - de signe, non.
2	* / mod div
3	- +
4	>, >=, <, <=
5	≠ , =
6	et
7	ou

si la priorité est égale, la priorité est pour l'opération de gauche

Expression

C'est une structure de valeurs et d'identifiants(), liée à l'aide des opérations, et lors de son évaluation, nous obtenons une seule valeur.

Exemple: Supposons a=2, b=3 et ok=vrai

expression	résultat	expression	résultat	expression	résultat
5	5	a+3	5	ok	vrai
a	2	"In"+"fo"	Info	a*(b-7)>8 et ok	faux

Instruction

C’est une phrase ou une étape de la solution

Bloc d'instructions

- un ensemble d’instructions qui commence par le mot Début et se termine par Fin, ou commence par un mot réservé qui définit le début tel que si et se termine par le mot fin + le mot de départ tel que finSi.
- En C, il commence par { et se termine par }.

exemple:

algorithme	C
Début Instruction 1 … Instruction n **Fin**	**{** Instruction 1 ; … Instruction n ; **}**

Affectation

C’est le processus qui nous permet de stocker une valeur dans une variable.

Syntaxe:

variable←expression

←lu reçoit, et en anglais gets. La flèche pointe toujours vers la variable.

- la valeur de l'expression et la variable doivent être du même type, ou de types compatibles.
- Avant qu'une variable puisse être utilisée, elle doit être déclarée et initialisée.
- Pour obtenir la valeur d’une variable ou constante, il suffit d'écrire son nom.

Exemple

a←5	a reçoit 5
b←a*2	b reçoit 10
a←0	a reçoit 0
b←b-1	b reçoit 9
c←'b'	c reçoit la lettre b
d←b>a	d reçoit vrai
s←"name"	s reçoit le mot nom

a	b
5	?
5	10
0	10
0	9

Instructions d'Entrées/Sorties

Affin d’interagir avec l’utilisateur le développeur dispose deux instructions:

- lire()
- écrire()

Entrée: Lire()

Permet de récupérer une valeur saisie par l'utilisateur, via le clavier, et de l'affecter à la variable entre parenthèses. Il est toujours utilisé pour saisir des données.

Syntaxe:

```
Lire(variable)
```

lire() ne peut être utilisé qu'avec des variables.
Lorsque l'instruction lire() est rencontrée, l'exécution s'interrompt jusqu'à ce que l'utilisateur entre les données. Le processus de saisie se termine en appuyant sur la touche Entrée.
Plusieurs variables peuvent être saisies à la fois, séparées par une virgule « , ».

Exemple

```
Lire(nom)
Lire(a, b)
```

Remarques

- toujours avant l'instruction Lire(), vient l'instruction Ecrire(), afin d'expliquer à l'utilisateur ce qui lui est demandé d'entrer.

Sortie: Ecrire()

Elle affiche sur l'écran tout ce que nous mettons à l'intérieur de ses parenthèses. Elle est toujours utilisée pour imprimer les résultats.

Syntaxe:

```
Ecrire(expression)
```

ou

```
Ecrire("message")
```

- expression, calculée pour obtenir une seule valeur, qui sera affichée à l'écran.
- "message": est tout texte à afficher tel quel à l'écran. Il n'est pas calculé. C'est dans n'importe quelle langue. Il doit être entre guillemets doubles, qui ne sont pas affichés
- Plusieurs valeurs et textes peuvent être affichés à la fois, séparés par « , ».

Exemple:

```
a←5
Ecrire(a+3)
Ecrire("a+3")
Ecrire("carré de ", a, " est ", a*a)
Ecrire("b=",a)
```

6. Construction d'un algorithme simple

- L’en-tête: Algorithme + nom
- La déclaration: les variables et les constantes
- Les instructions: La partie instruction, généralement, comprend trois étapes de base:
  - La première étape, « Inputs »: les données nécessaires à la mise en œuvre sont saisies. en utilisant l’instruction <Lire()>.
  - La deuxième étape, « le traitement »: Elle contient un ensemble d'instructions nécessaires pour résoudre le problème. à l'aide de l'instruction d'affectation.
  - La troisième étape, "Outputs": les résultats sont présentés. à l'aide de l'instruction <Ecrire()>.

Exemple 1:

Écrivez un algorithme qui calcule l'aire d'un cercle.

```
Algorithme surf_cercle
Const
     P=3.14
Var
   r, s:entier //r est le rayon et s est la surface
Début
   Ecrire("Entrer le rayon")
   Lire(r)
   s←p*r*r
   Ecrire("L'aire du cercle est:" , s)
Fin
```

Exemple 2:

Écrivez un algorithme qui calcule la moyenne pour ASD1.

```
Algorithme moy_ASD1
Var
   cont, td, tp, moy:réel
Début
   Ecrire("Saisir la note d'examen, de travaux dirigés
       et de travaux pratiques")
   Lire(cont, td, tp)
   moy←(cont*3+td+tp)/5
   Ecrire("La moyenne est:" , moy)
Fin
```

Exécution d'un algorithme

Elle vise à connaître la valeur de chaque variable après chaque instruction.

après chaque affectation ou lecture, la valeur de la variable change.

Exemple

Algorithme miroir	Exécution		
Var			
a, b, c:entier	a	b	c
Début			
a←357	357	?	?
c←0	357	?	0
b←a mod 10	357	7	0
c←c*10+b	357	7	7
a←a div 10	35	7	7
b←a mod 10	35	5	7
c←c*10+b	35	5	75
a←a div 10	3	5	75
b←a mod 10	3	3	75
c←c*10+b	3	3	753
Fin			

7. L’algorigramme

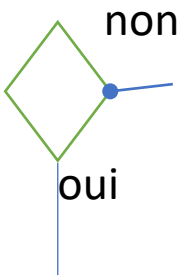
L’algorigramme est une représentation visuelle d'un algorithme . Il nous montre la séquence des opérations,

Un certain nombre de formes sont utilisées dans l'algorigramme

Début, fin, interruption

Entrée – Sortie

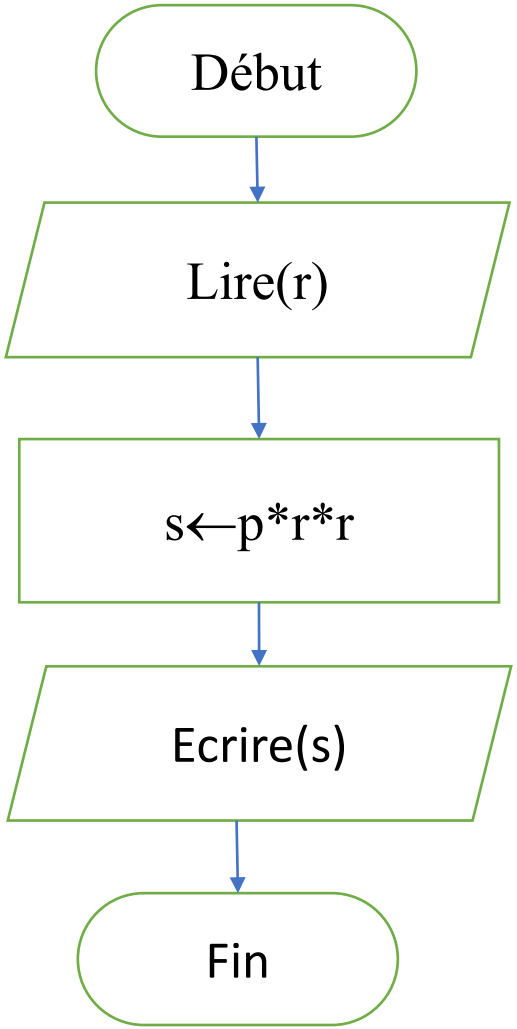
Traitement comme l'affectation



Choix avec condition

Exemple:

l'algorithme qui calcule l'aire d'un cercle



8. Traduction en langage C

"**C**" est un langage impératif de haut niveau entièrement compilé. C'est l'un des langages de programmation les plus utilisés au monde. Il est considéré comme la langue mère de nombreux langages de programmation.

"C" est sensible à la casse, car il fait la distinction entre MAJUSCULES et minuscules . <main> n'est pas la même chose que <Main>, <MAIN> ou <mAin>. Par conséquent, nous vous recommandons de toujours écrire en minuscules.

Toutes les instructions simples (déclaration, affectation, E/S, retour) doivent se terminer par un point-virgule ";".

Le préprocesseur

Avant la compilation du code source le fichier est analysé par un logiciel spécial connu par le préprocesseur afin d' inclure d'autres fichiers (bibliothèques), remplacer des mots par d'autres phrases (macro).

Les directives de préprocesseur commencent toujours par #.

#include: pour inclure le contenu d'un fichier dans le code du programme en cours.

```
#define nom_macro le_texte_de_remplacement
```

Exemple:

- Pour utiliser les fonctions I/O (scanf et printf), nous utilisons la librairie stdio.h.
- Pour utiliser les fonctions mathématiques (sin, cos, exp, pow, sqrt, ...), on utilise la bibliothèque math.h.

```
#include <stdio.h>
#define N 10
```

Le préprocesseur remplace toutes les occurrences du mot N par 10.

Les types

- Entiers dans l'algorithme de -∞ à +∞

Types	Taille en octets	Taille en bits	intervalle
char	1	8	-2^7 ,2^7-1
short	2	16	-2^{15} ,$2^{15}-1$
long	4	32	-2^{31} ,$2^{31}-1$
int	4	32	-2^{31} ,$2^{31}-1$

Les nombres naturels dans les algorithmes de 0 à +∞.
nous utilisons généralement des entiers pour les exprimer.
On peut ajouter <unsigned> devant un type en C pour exprimer les nombres naturels.

Types	Taille en octets	Taille en bits	intervalle
unsigned char	1	8	0 ,2^8-1
unsigned short	2	16	0 ,$2^{16}-1$
unsigned long	4	32	0 ,$2^{32}-1$
unsigned int	4	32	0 ,$2^{32}-1$

Les types

Nombres réels

Types	Taille en octets	la précision
float	4	6
double	8	8
long double	10	8

- Type booléen: il n'y a pas en C, mais int est utilisé à la place. où vrai est le nombre 1 et faux est 0. Tout nombre autre que 0 est traduit par vrai.
- Le type de caractères est char.
- Type chaîne de caractères: Pour exprimer la chaîne de caractères en C, on utilise des tables (Chapitre 5) char[] ou des pointeurs (second semestre) char*.
- Le type char est utilisé pour les entiers et les caractères. Où chaque caractère est associé à un nombre.
- Dans ce cours, nous utilisons **char** pour les caractères, **int** pour les entiers et booléen, et **float** pour les nombres réels.

Déclaration des variables et des constantes

Variables

Syntaxe:

```
Type variable;
```

exemple:

```
int age;
char sexe;
float x, y, z;
Banane b;
```

Constantes

Syntaxe:

const Type Identificateur=valeur;

ou

const Type Identificateur=valeur;

exemple:

```
int const N = 10;
const float Pl = 3.1415926;
```

Remarque: Des macros peuvent être utilisées pour déclarer des constantes.

#define DEP "Département informatique"

L'affectation

Nous utilisons = à la place de ← et lisons reçoit et non pas égale.

```
variable = expression;
```

exemple:

```
a=5;

b=a*2;

a=0;

d=b>a;
```

Déclaration avec initialisation

```
float x, y=3, z;//y est initialisé avec 3
```

Dans le cas où l'affectation apparaît plusieurs fois, la priorité est à droite. Tel que:

```
b=3 ;
a=b=5+3 ;
```

Raccourcis d'affectation en C.

<table>
<tr><th>expression</th><th>observation</th><th>exemple</th></tr>
<tr><td>v+=exp ; ⇔ v=v+(exp) ;</td><td rowspan="5">Les parenthèses sont importantes</td><td rowspan="5">x=2 ;
x*=5+3 ; ⇔ x=x*(5+3) ;
≠ x=x*5+3 ;
x devient 16</td></tr>
<tr><td>v-=exp ; ⇔ v=v-(exp) ;</td></tr>
<tr><td>v*=exp ; ⇔ v=v*(exp) ;</td></tr>
<tr><td>v/=exp ; ⇔ v=v/(exp) ;</td></tr>
<tr><td>v%=exp ; ⇔ v=v%(exp) ;</td></tr>
<tr><td>v++ ; ⇒ v=v+1 ;</td><td rowspan="2">Dans le cas où il est contenu dans une autre instruction, le calcul est effectué avec la valeur actuelle de la variable, et après achèvement, 1 est ajouté ou soustrait à la variable selon l'opération.</td><td rowspan="2">x=2 ;
y=3+x++ ;⇔
y=3+x ;x=x+1 ;
Autrement dit, y devient 5 et x 3</td></tr>
<tr><td>v-- ; ⇒ v=v-1 ;</td></tr>
<tr><td>++v; ⇒ v=v+1 ;</td><td rowspan="2">S'il est inclus dans une autre instruction, 1 est ajouté ou soustrait selon l'opération avant le calcul de l'expression, qui se fait avec la nouvelle valeur de la variable.</td><td rowspan="2">x=2 ;
y=3+ ++x ;⇔ x=x+1 ;
y=3+x ;
Autrement dit, y devient 6 et x 3</td></tr>
<tr><td>--v ; ⇒ v=v-1 ;</td></tr>
</table>

printf (print formatted)

La fonction printf, définie dans la bibliothèque stdio.h, est utilisée pour écrire des données formatées sur l'écran

Syntaxe:

```
printf (format, expression_1,… , expression_n);
```

format: est un texte ou une chaîne de caractères, qui s'affiche tel quel à l'écran. Sauf pour le symbole "%" pour exprimer les formats des expressions et le caractère "\" pour le symbole d'échappement.

expression: calculée pour être afficher dans le format <format>.

Le format prend la forme suivante:

`%type_char`

type_char

%d	décimal (10)	%o	octal (8)	%x	hexadécimal (16)
%u	naturel **u**nsigned	%i	int	%f	float
%c	char	%s	string	%e	format scientifique

technique d'échappement

Certains caractères sont spéciaux, nous devons donc utiliser une technique d'échappement pour les utiliser. En C, le caractère d'échappement est le (\), barre oblique inverse (antislash), et nous l'utilisons pour ajouter une nouvelle ligne '\n' ou une tabulation (un grand espace) '\t'. et pour imprimer " nous utilisons \", pour imprimer \ nous utilisons \\ et pour imprimer % nous utilisons %%.

exemple:

```
printf("Bonjour");
```

affiche Bonjour

```
int a=13;
printf("a=(%d)10\ta=(%o)8\ta=(%x)16\n",a ,a ,a);
```

a=(13)10 a=(15)8 a=(D)16

```
a=66;
printf("a=%i\ta=%f\ta=%c\ta=%c\n",a ,a ,a ,a+32);
```

a=66 a=0.000000 a=B a=b

Parce que 66 n'est pas forcément 66 en float

Et 66 si nous le voyons comme un caractère, il représente la lettre B tandis que 66 + 32 = 98 représente le codage de la lettre b en minuscules.

```
float pi=3.1415926;
printf("%f\t%.4f\t%06.2f" ,pi ,pi ,pi);
```

3.141593 3.1416 003.14

scanf (scan formatted)

La fonction scanf, définie dans la bibliothèque stdio.h, est utilisée pour lire les données formatées sur le clavier

Syntaxe:

```
scanf(format, &variable_1, … , &variable_n);
```

format: Chaîne de caractères représentant le format de lecture.

variable: Représente le nom de la variable. Il est précédé par <&>, sauf s'il s'agit d'une variable de type chaîne (de type pointeur).

Le **format** prend la forme suivante:

%type_char (mêmes symboles dans printf)

Exemple:

```
scanf("%s", nom);

scanf("%d%f",&a, &b);
```

problème de scanf avec les caractères

Lors de la lecture d'un caractère à l'aide de scanf("%c"...), l'utilisateur saisit le premier caractère, puis appuie sur la touche Entrée, qui, à son tour produit le caractère '\n', qui n'est pas supprimé de la mémoire, et lorsque le programme rencontre l'instruction scanf("%c", ...) pour la deuxième fois, il n'attend pas ce que l'utilisateur entrera, mais assigne plutôt le caractère \n à la deuxième variable.

Pour éviter ce problème, dans le second scanf nous utilisons un espace ' ' après % comme ceci: scanf("% c"...). Ou nous utilisons la fonction **getch**

Exemple:

```
scanf("%c", &c1) ;
scanf("% c", &c2) ;
```

problème de scanf avec les chaînes.

Le problème apparaît lorsque nous essayons d'insérer une chaîne contenant des espaces dans une variable. par exemple:

```
scanf("%s%s", v1,v2) ;
```

Elle met le premier mot dans v1 et le deuxième dans v2

```
scanf("%s", v1) ;
```

Lorsque nous entrons les mots math info et que nous appuyons sur Entrée, le programme attribue le premier mot à v1 et le deuxième mot sera perdu.

Pour éviter ce problème, nous utilisons la fonction gets définie dans la bibliothèque string.h.

```
#include <sting.h>

gets(v1);
```

Structure d'un programme en C

```
1.  #include <stdio.h>
2.    Déclaration des constantes, des types et des variables
3.  int main()
4.  {
5.    Déclaration des constantes et des variables
6.    Les instructions
7.    return 0;
8.  }
```

Un exemple montrant le processus de traduction d'un algorithme en C

algorithme	C
Algorithme surf_cercle	
Const P=3.14	Const float P=3.14;
Var r, s:entier	int r, s;
//r le rayon et s surface	//r le rayon et s surface
Début	int main() {
Ecrire("Entrer le rayon")	printf("Entrez le rayon\n");
Lire(r)	scanf("%d", &r);
s←p*r*r	s=p*r*r;
Ecrire("L'aire du cercle est:" , s)	printf("L'aire du cercle est: %d" , s);
Fin	}

exemple

Écrivez un programme qui calcule la moyenne pour ASD1.

```
#include <stdio.h>
int main() {
   float cont, td, tp, moy ;
   printf("Entrez la note d'examen\n") ;
   scanf("%f", &cont) ;
   printf("Entrez la note de TD\n") ;
   scanf("%f", &td) ;
   printf("Entrez la note de TP\n") ;
   scanf("%f", &tp) ;
   moy = (cont * 3 + td + tp) / 5 ;
   printf("La moyenne est %.2f" , moy) ;
   return 0;
}
```

Fin Chapitre 02