

# Chapitre 4:

# Les entrées-sorties

# en C

# La fonction printf

- Pour l’affichage formaté des données, on utilise la fonction `printf`.
- Formaté signifié que nous pouvons contrôler les expressions et les `formats` des données affichées.
- La fonction `printf` admet la syntaxe suivante:

```
printf("formats",arg_1, arg_2,..., arg_n);
```

# La fonction printf

Le format est interprété de la manière suivante:

- i. Les caractères ordinaires sont affichés tels quels.
- ii. Les caractères précédés d'une barre oblique inverse (`\`) permettent de faire de la mise en page (`\n`, `\t`, ...).
- iii. Une séquence commençant par le caractère `%` sera remplacé par la valeur d'un argument. Les caractères qui suivent le `%` précisent le type de l'argument et la manière dont il sera affiché.

## Exemple

## Exemple

```

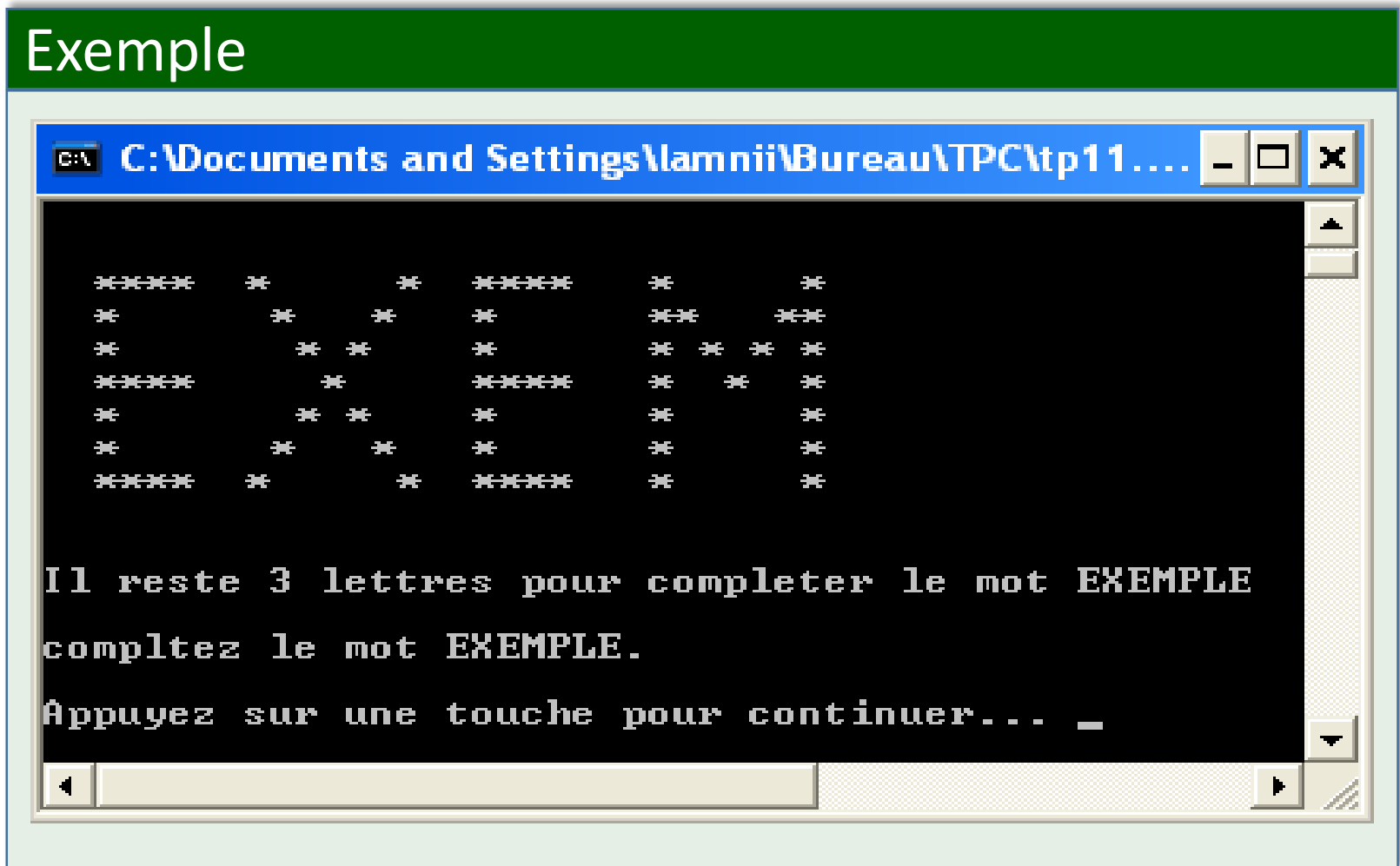
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    int n=3;
    printf("      \n\n\n") ;

    printf("    ****  *      *  ****  *      *  \n") ;
    printf("    *          *  *  *          **      **  \n") ;
    printf("    *          *  *  *          *  *  *  *  \n") ;
    printf("    ****          *      ****  *  *  *  \n") ;
    printf("    *          *  *  *          *          *  \n") ;
    printf("    *          *  *  *          *          *  \n") ;
    printf("    ****  *      *  ****  *      *  \n\n\n") ;
    printf("Il reste %d lettres pour completer le mot EXEMPLE\n\n",n)
    printf("comptez le mot EXEMPLE.\n\n"  ) ;
    system("PAUSE");
    return 0;
}

```

## Exemple

## Exemple



# La fonction printf

- Par défaut, les entiers sont affichés avec le nombre de caractères nécessaires (sans espaces avant ou après). Les flottants sont affichés avec six chiffres après le point (aussi bien pour le code `e` que `f`). Mais on peut introduire entre le `%` et le caractère de spécification de format d'affichage quelques caractères de contrôles d'affichage:
  - ✓ Le caractère `'-'` permet d'afficher l'argument aligné à gauche (par défaut à droite).
  - ✓ Le caractère `'+'` fait qu'un nombre est toujours affiché avec son signe (par défaut seul le moins est affiché).

# La fonction printf

- ✓ Un nombre précisant le **gabarit d'affichage**, c.-à-d. un nombre **minimal** de caractère à utiliser.
- ✓ Le caractère **'.'** suivi d'un nombre permet de spécifier pour les valeurs réels le nombre de chiffres après la virgule.

## Exemple

```
printf ("%10.3f", x) ;
```

x = 1.2345

^^^1.235

x = 1.2345E3

^^1234.500

x = 1.2345E7

12345000.000

# La fonction scanf

- La fonction `scanf` autorise la saisie formatée de données (depuis le clavier) et on la désigne comme `printf`. Sa syntaxe est :

```
scanf("format",adresse_1, adresse_2,..., adresse_n);
```

- La chaîne de format peut contenir ici aussi bien des caractères usuels que des spécifications de format, ces dernières commençant par le caractère `%` comme avec `printf`.



# La fonction scanf

- L'opérateur d'adressage `&` appliqué au nom de la variable informe donc la fonction `scanf` de l'emplacement de stockage de la valeur saisie.
- `scanf` attend donc comme argument non pas le nom (donc la valeur) d'une variable, mais au contraire son adresse.

## Exemple

```
// Lecture d'un entier et d'un réel :  
int n;  
float x;  
scanf("%d%f", &n,&x);
```

# La fonction scanf

- **Notion de tampon et séparateurs**
- ✓ Lorsque scanf attend que vous lui fournissiez des données, l'information frappée au clavier est rangée temporairement dans l'emplacement mémoire nommé «**tampon**». Ce dernier est exploré, caractère par caractère par scanf, au fur et à mesure des besoins.
- ✓ D'autre part, certains caractères dits «**séparateurs**» jouent un rôle particulier dans les données: **\_**, **\n**, **\t**, **\v** et **\f**

# La fonction scanf

- Règles utilisées par scanf
- ✓ Le code de format d'un nombre entraîne d'abord l'avancement éventuel du pointeur jusqu'au premier caractère différent d'un séparateur. Puis scanf prend en compte tous les caractères suivants jusqu'à la rencontre d'un séparateur (en y plaçant le pointeur), du moins lorsque aucun gabarit n'est précisé et qu'aucun caractère invalide n'est présent dans la donnée.
- ✓ Quant au code de format %c, il entraîne la prise en compte du caractère désigné par le pointeur et le pointeur est simplement avancé sur le caractère suivant du tampon.

# La fonction scanf

## Exemple

```
scanf ("%d%d", &n, &p) ;
```

```
12^25@           n = 12  p = 25
^12^^25^^@       n = 12  p = 25
```

```
12@
@
^25@           n = 12  p = 25
```

```
//-----
scanf ("%c%d", &c, &n) ;
```

```
a25@           c = 'a'  n = 25
a^^25@         c = 'a'  n = 25
```

```
//-----
scanf ("%d%c", &n, &c) ;
```

```
12 a@           n = 12  c = ' '
```

# La fonction scanf

## ■ Imposition d'un gabarit maximal

Comme dans les codes de format de printf, on peut, dans un code de format de scanf, préciser un gabarit. Dans ce cas, le traitement d'un code de format s'interrompt soit à la rencontre d'un séparateur, soit lorsque le nombre de caractères indiqués a été atteint (les séparateurs éventuellement sautés auparavant ne sont pas comptabilisés !).

### Exemple

```
scanf ("%3d%3d", &n, &p)
```

```
12^25@
```

```
n = 12  p = 25
```

```
^^^^^12345@
```

```
n=123  p=45
```

```
12@
```

```
25@
```

```
n = 12  p = 25
```

# La fonction scanf

## ■ Rôle d'un espace dans le format

- Un espace entre deux codes de format demande à scanf de faire avancer le pointeur au prochain caractère différent d'un séparateur.
- Notez que c'est déjà ce qui se passe lorsque l'on a affaire à un code de format correspondant à un type numérique.

## Exemple

➤ `scanf("%d^%c",&n,&c);` /\* ^ désigne un espace \*/  
 /\* %d^%c est différent de %d%c \*/

12^a@a                      n = 12 c = 'a'

12^^^a@a                  n = 12 c = 'a'

12@a@a                    n = 12 c = 'a'

# La fonction scanf

## Exercices

- 1- Écrire un programme qui calcul le volume d'un cylindre de rayon  $r$  et de hauteur  $h$  :  $V = \pi * r * r * h$ .
- 2- Écrire un programme qui échange les valeurs de deux entiers.

# La macro putchar

- La macro `putchar` permet d'afficher un seul caractère sur l'écran de l'ordinateur.
- La syntaxe de son utilisation est la suivante :  
`putchar(variable) ;`  
où variable est de type `char`.



# La macro getchar

- La macro `getchar` permet la récupération d'un seul caractère à partir du clavier.
- La syntaxe d'utilisation de `getchar` est la suivante :

`variable=getchar() ;`

- Noter que la variable doit être de type `char` et les parenthèses après `getchar` sont nécessaires.

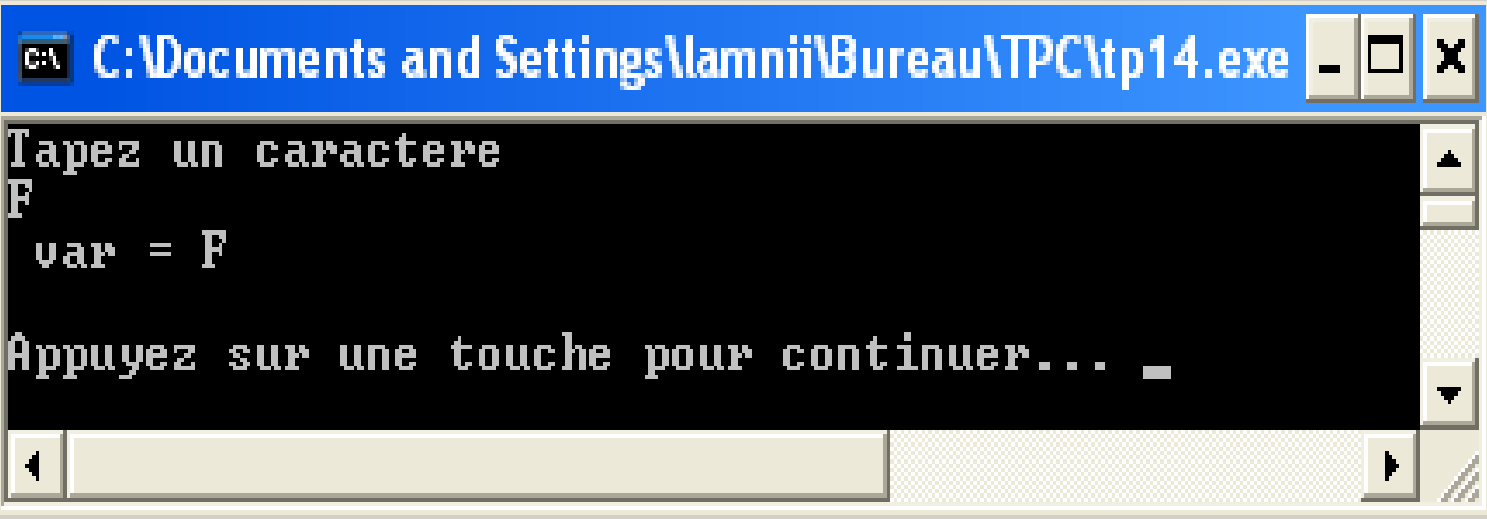
# Exemple

## Exemple

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    char var ;
    printf("Tapez un caractere\n");
    var=getchar() ;
    printf(" var = ") ;
    putchar(var) ;
    printf("\n\n");
    system("PAUSE");
    return 0;
}
```

# Exemple

## Exemple



```
C:\Documents and Settings\lamnii\Bureau\TPC\tp14.exe
Tapez un caractere
F
var = F
Appuyez sur une touche pour continuer... _
```