

Chapitre 4

Entrées / Sorties

Plan

- 1. Syntaxe de `printf()`**
2. Spécifications de format
3. Séquences d'échappement `'\'`
4. Syntaxe de `scanf()`
5. Précautions avec `scanf()`
6. Autres fonctions d'entrée/sortie

4.1 Affichage & saisie

Affichage d'informations à l'écran

`printf()`

Texte, contenu de variables

Saisie de valeurs à partir du clavier

`scanf()`

Les valeurs saisies sont toujours stockées dans des variables.

Ces fonctions nécessitent d'inclure le fichier

```
#include <stdio.h>
```

4.1 printf()

```
printf("<Chaîne de commandes>", <Liste d'expression>);
```

<Chaîne de commandes>

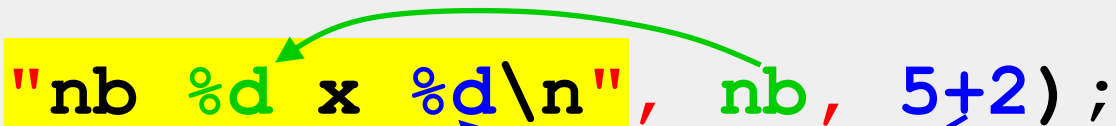
1. Un **texte** ("Hello World!")
2. La **spécification du format d'affichage** des variables → `'%d'`
3. Les **caractères d'échappement** pour la mise en page → `'\n'`

<Liste d'expression>

Liste des variables ou d'expressions à afficher

Exemple

```
nb=20;  
printf("nb %d x %d\n", nb, 5+2);
```



→

```
nb 20 x 7
```

Plan

1. Syntaxe de `printf()`
- 2. Spécifications de format**
3. Séquences d'échappement `'\'`
4. Syntaxe de `scanf()`
5. Précautions avec `scanf()`
6. Autres fonctions d'entrée/sortie

4.2 printf() – formats d’affichage

Caractères

| Format | Type | Représentation |
|-----------|--------|---|
| %c | char | caractère simple |
| %s | char * | chaîne de caractères (via un pointeur sur char) |

4.2 printf() – formats d’affichage

Entiers

| Format | Type | Représentation |
|--------------|---------------|---|
| %d | int | entier décimal signé |
| %i | int | entier signé (selon le préfixe) |
| %u | unsigned int | entier décimal non-signé |
| %x %X | unsigned int | entier hexadécimal non-signé [a...f] ou [A...F] |
| %o | unsigned int | entier octal non-signé |
| %p | void * | pointeur pur |

4.2 printf() – formats d’affichage

Nombre à virgule flottante

| Format | Type | Représentation |
|-------------------------------------|------------------------|--|
| <code>%f</code> <code>%lf</code> | float double | valeur signée <code>[-]zzz.fff</code> |
| <code>%e</code> <code>%E</code> | float double | valeur signée <code>[-]z.fffffffeE±nn</code> <code>[-]z.fffffffEe±nn</code> |
| <code>%g</code> <code>%G</code> | float double | Utilise automatiquement le format <code>%e</code> ou <code>%f</code> suivant la valeur et la précision données. |

4.2 `printf()` – formats d’affichage

D'autres spécifications de formats peuvent être utilisées par l'instruction `printf()`

| Type | Format |
|--------|------------------|
| short | <code>%hd</code> |
| long | <code>%ld</code> |
| float | <code>%e</code> |
| double | <code>%e</code> |

| Type | Format |
|----------------|------------------|
| unsigned char | <code>%u</code> |
| unsigned short | <code>%hu</code> |
| unsigned int | <code>%u</code> |
| unsigned long | <code>%lu</code> |

4.2 Affichage multiple

Un seul **printf()** peut afficher simultanément plusieurs variables et constantes.

Exemple

```
printf("Annee %d, Mois %d, Jour %d\n", 1991, 5, 10);
```



→ `Annee 1991, Mois 5, Jour 10`

La *liste d'expressions* doit **posséder le même nombre de valeurs** qu'il y a de spécifications de format.

4.2 Affichage multiple – Erreurs possibles

- (a) valeurs supplémentaires → ignorées ☐
- (b) valeurs manquantes → affichage aléatoire ☐
- (c) type des expressions ne correspondant pas au format d'affichage →
décodage de façon incorrecte ☐

```
/* What should have been implemented: */  
printf("Year %d, month %d, day %d\n", 1991, 5, 10);  
  
/* 3 types of implementation issues: */  
printf("(a) Year %d, month %d\n", 1991, 5, 10);  
printf("(b) Year %d, month %d, day %d\n", 1991, 5, 10);  
printf("(c) Year %d, month %d, day %f\n", 1991, 5, 10);
```



```
Year 1991, month 5, day 10  
(a) Year 1991, month 5  
(b) Year 1991, month 5, day -2120229984  
(c) Year 1991, month 5, day 0.000000
```

4.2 printf() – le type char

Avec le **type char**

%c affiche le caractère **Z**

%d affiche son code ASCII **90**

```
printf("%c \n", 'Z');  
printf("%d \n", 'Z');  
printf("%c \n", "Z");  
printf("%s \n", "Zoo");
```

→

```
Z  
90  
  
Zoo
```

4.2 printf() – types réelles

Avec les **types réels**, le format peut contenir des indications pour la **largeur** et la **précision**.

```
% [L] [.P] f  
% [L] [.P] lf
```

La spécification de **largeur L** fixe la largeur minimale de la valeur à afficher.

La spécification de **précision P** commence toujours par un point et indique que **P** décimales sont affichées.

4.2 printf() – formats d’affichage

```
printf("%f", 157.89260032);
```

1 5 7 . 8 9 2 6 0 0

```
printf("%6.1f", 157.89260032);
```

| | | | | | |
|--|---|---|---|---|---|
| | 1 | 5 | 7 | . | 9 |
|--|---|---|---|---|---|

→ **décalage ET arrondi**

```
printf("%+09.2f", 157.89260032);
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| + | 0 | 0 | 1 | 5 | 7 | . | 8 | 9 |
|---|---|---|---|---|---|---|---|---|

```
printf("%7.5f", 157.89260032);
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 7 | . | 8 | 9 | 2 | 6 | 0 |
|---|---|---|---|---|---|---|---|---|

→ **L (7) est un minimum**

```
printf("%.3f", 157.89260032);
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 5 | 7 | . | 8 | 9 | 3 |
|---|---|---|---|---|---|---|

```
printf("%f \n", 157.89260032);
printf("%6.1f \n", 157.89260032);
printf("%+09.2f \n", 157.89260032);
printf("%7.5f \n", 157.89260032);
printf("%.3f \n", 157.89260032);
```

→

| |
|------------|
| 157.892600 |
| 157.9 |
| +00157.89 |
| 157.89260 |
| 157.893 |

Plan

1. Syntaxe de `printf()`
2. Spécifications de format
- 3. Séquences d'échappement `'\'`**
4. Syntaxe de `scanf()`
5. Précautions avec `scanf()`
6. Autres fonctions d'entrée/sortie

4.3 `printf()` – caractères de contrôle

Les caractères de contrôle commence par un ***backslash*** '`\`'

Appelé également **caractère d'échappement**
→ ***escape character***

Il existe plusieurs caractères de contrôles.

En particulier '`\n`' correspond à un «retour à la ligne».

4.3 printf() – caractères de contrôle

| Escape sequence ↕ | Hex value in ASCII ↕ | Character represented ↕ |
|------------------------------|----------------------|---|
| \a | 07 | Alert (Beep, Bell) (added in C89) ^[1] |
| \b | 08 | Backspace |
| \e ^{note 1} | 1B | Escape character |
| \f | 0C | Formfeed Page Break |
| \n | 0A | Newline (Line Feed); see notes below |
| \r | 0D | Carriage Return |
| \t | 09 | Horizontal Tab |
| \v | 0B | Vertical Tab |
| \\ | 5C | Backslash |
| \' | 27 | Apostrophe or single quotation mark |
| \" | 22 | Double quotation mark |
| \? | 3F | Question mark (used to avoid trigraphs) |
| \nnn ^{note 2} | any | The byte whose numerical value is given by <i>nnn</i> interpreted as an octal number |
| \xhh... | any | The byte whose numerical value is given by <i>hh...</i> interpreted as a hexadecimal number |
| \uhhhh ^{note 3} | none | Unicode code point below 10000 hexadecimal (added in C99) ^{[1]:26} |
| \Uhhhhhhhh ^{note 4} | none | Unicode code point where <i>h</i> is a hexadecimal digit |

Source : *Escape sequences in C*,
https://en.wikipedia.org/wiki/Escape_sequences_in_C

4.3 printf() – caractères de contrôle

Quelques exemples

```
/* Bip sonore sur le haut-parleur du PC */  
printf("\\a");
```

```
/* Afficher : C:\\EXEMPLES\\TOTO.TXT à l'écran. */  
printf("C:\\\\EXEMPLES\\\\TOTO.TXT");
```

```
/* Afficher: "Pardon ?" */  
printf("\\\"Pardon ?\\\"");
```

Plan

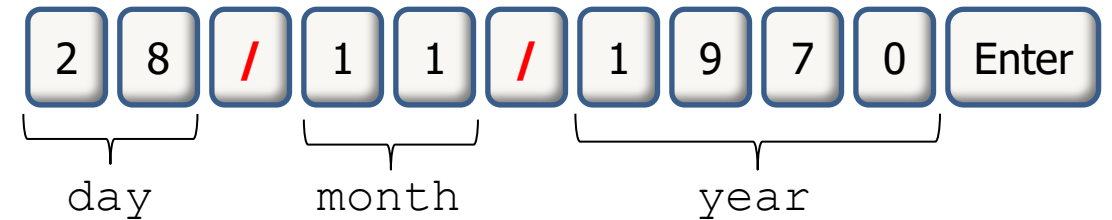
1. Syntaxe de `printf()`
2. Spécifications de format
3. Séquences d'échappement `'\'`
- 4. Syntaxe de `scanf()`**
5. Précautions avec `scanf()`
6. Autres fonctions d'entrée/sortie

4.4 scanf () – syntaxe

```
scanf("<Chaîne de commande>", &<Var1>, &<Var2>, ..., &<VarN>);
```

Par exemple

```
int day, month, year;  
scanf("%d/%d/%d", &day, &month, &year);
```



<Chaîne de commandes>

1. la **spécification du format de lecture** des variables → `'%d'`
2. des **caractères alphanumériques**

<Liste de variables>

Liste des adresses des variables à saisir

Les variables sont autorisées à figurer comme paramètres

Les constantes sont interdites

4.4 scanf () – commentaire

Chaque nom de variable doit être précédé du symbole '&'

Ainsi '&day' signifie **adresse de** la variable 'day'

Lors de la saisie, il faut entrer tous les caractères alphanumériques de la chaîne de commande

Exemple

```
int day, month, year;  
printf("\n\nFill as specified: dd/mm/yyyy ");  
scanf("%d/%d/%d", &day, &month, &year);  
printf("\nSorted info: %d / %d / %d", day, month, year);
```



```
Fill as specified: dd/mm/yyyy 28/11/1970  
Sorted info: 28 / 11 / 1970
```

4.4 scanf () – format d'affichage

La chaîne de contrôle de `scanf ()` a **les mêmes spécifications de format** que `printf ()` **sauf pour les types réels**

| Format | type | Représentation |
|------------------|--------|--------------------------|
| <code>%f</code> | float | valeur signée [-]zzz.fff |
| <code>%lf</code> | double | valeur signée [-]zzz.fff |

En résumé, avec `printf` :

`%f` et `%lf` pour les variables de type `float` et `double`

et avec `scanf` :

`%f` pour les variables de type `float`

`%lf` pour les variables de type `double`

4.4 scanf () – format d'affichage

Certains caractères jouent un rôle particulier avec `scanf`. On les appelle **séparateurs** ou "**caractères blancs**". Les plus courants sont   

Si l'un de ces caractères fait partie de la chaîne de commande, on peut saisir à l'endroit où il se trouve autant de caractères blancs que l'on désire (ou aucun).

Exemple

```
scanf(" %d %d", &jour, &mois);
```

les deux nombres peuvent être séparés par des

espaces

tabulations

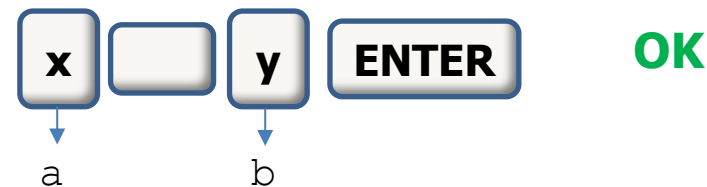
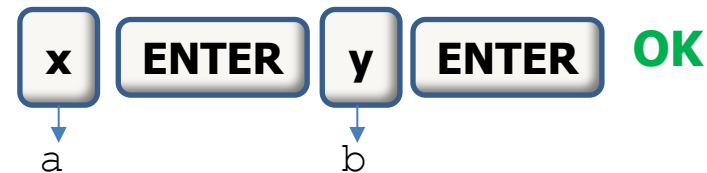
retours à la ligne



4.4 scanf () – précautions (1)

Comment lire deux caractères ?

Méthode 1 `char a, b;
scanf ("%c %c", &a, &b) ;`



4.4 scanf () – précautions (2)

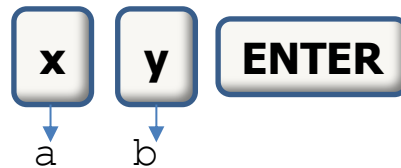
Comment lire deux caractères ?

Méthode 2:

```
char a, b;  
scanf ("%c", &a);  
scanf ("%c", &b);
```

Pas d'espace !

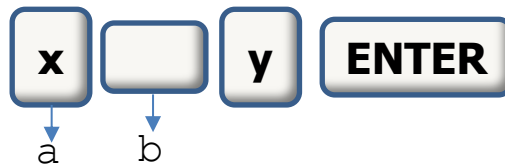
`scanf ("%c%c", &a, &b);`



OK



PAS OK



PAS OK

Précaution :

En cas de plusieurs scanf, mettre un espace devant le premier caractère de formatage

`scanf (" %c", &a);`

4.4 Exemple (avec des char)

```
#include <stdio.h>

int main(void)
{
    int year, month, day;
    char name1, name2;
    int status = 0;

    printf("First letter in your first name: ");
    status = scanf(" %c", &name1);
    printf("First letter in your last name: ");
    status = scanf(" %c", &name2);

    printf("Your birthdate (jj/mm/aa): ");
    status = scanf(" %d/%d/%d", &day, &month, &year);

    printf("\nYour initials are %c.%c.\n", name1, name2);
    printf("You are born on %d/%d/%d\n", day, month, year);

    printf("\nASCII CODE: %c=%d, %c=%d\n", name1, name1, name2, name2);

    return 0;
}
```

4.4 Exemple (avec des strings)

```
#include <stdio.h>

int main(void)
{
    // Messages are "const char *" because they are not supposed to be modified
    const char *enter_your_first_name_msg = "Enter your first name: ";
    const char *enter_your_last_name_msg = "Enter your last name: ";
    // first_name and last_name are arrays of characters because they are supposed to be modified
    char first_name[80];
    char last_name[80];

    int status = 0;

    // WARNING: this will not work if the name has a space in it
    printf("%s", enter_your_first_name_msg);
    status = scanf(" %s", first_name);
    printf("%s", enter_your_last_name_msg);
    status = scanf(" %s", last_name);

    printf("Your name is %s %s\n", first_name, last_name);

    return 0;
}
```

Plan

1. Syntaxe de `printf()`
2. Spécifications de format
3. Séquences d'échappement `'\'`
4. Syntaxe de `scanf()`
- 5. Précautions avec `scanf()`**
6. Autres fonctions d'entrée/sortie

4.5 Précautions avec scanf ()

```
char ligne[80];
```

```
scanf ("%[ ABCDEF]s", ligne);
```

N'accepte que les caractères ABCDEF, ainsi que l'espace ici, dans la variable *ligne*

Comme vu précédemment, il n'y a pas de signe &

```
scanf ("%[^ABCDEF]s", ligne);
```

Accepte tous les caractères sauf ABCDEF dans la variable ligne

Dès qu'un de ces caractères est trouvé, la saisie s'arrête

```
status = scanf ("%d:%d", &h, &m);
```

Lit les variables *h* et *m* au clavier, séparées du signe :

La variable *status* (de type *int*) le nombre de paramètres saisis sans erreur

Plan

1. Syntaxe de `printf()`
2. Spécifications de format
3. Séquences d'échappement `'\'`
4. Syntaxe de `scanf()`
5. Précautions avec `scanf()`
- 6. Autres fonctions d'entrée/sortie**

4.6 Autres possibilités d'entrée / sortie

```
char ch, line[80];           // 'ch': caractère, 'line': chaîne de caractères
int Nmax=256;

puts(line);                  // écrit Le texte sur l'écran avec un <CR>*
fputs(line, stdout);         // Idem sans <CR>*

gets(line);                 // lit la variable ligne au clavier* (déprécié)
gets_s(line, Nmax);           // remplacement avec C11*
fgets(line, Nmax, stdin);     // ou lecture d'un flux

putchar(ch);                  // affiche le caractère 'ch' sur l'écran*

ch = getchar();               // Lit le caractère 'ch' au clavier (tampon)*
ch = getch();                 // Lit le caractère 'ch' du clavier (direct)**
```

*<stdio.h>

**<conio.h>

4.6 Exemple – utilisation du tampon

```
char premier, lettre, mot[32], phrase[128];  
  
premier = getch();  
  
scanf("%c",&lettre);  
  
scanf("%s",mot);  
  
fgets(phrase,128,stdin);  
  
printf("premier : %c\n",premier);  
printf("lettre : %c\n",lettre);  
printf("mot : %s\n",mot);  
printf("phrase : %s\n",phrase);  
return 0;
```



alut comment vas-tu?

a l u t c o m m e n t v a s - t u ?

Tampon

premier : s
lettre : a
mot : lut
phrase : comment vas-tu?

Exercices



Exercices du chapitre 04