



Chaînes de caractères

Chaînes de caractères

"Chaîne de caractères"

C	h	a	î	n	e		d	e		c	a	r	a	c	t	è	r	e	s	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

char ch[21] = "Chaîne de caractères";

↓
caractère nul
= 0

ch[0] : 'C'; ch[12] : 'r'; ch[20] : '\0'

"A" ≠ 'A'

↳

A	\0
0	1

ASCII TABLE

Decimal	Hex	Char
0	0	[NULL]
1	1	[START OF HEADING]
2	2	[START OF TEXT]
3	3	[END OF TEXT]
4	4	[END OF TRANSMISSION]
5	5	[ENQUIRY]
6	6	[ACKNOWLEDGE]
7	7	[BELL]

I/O : printf() / scanf()

format pour les chaînes de caractères : %s

```
char prenom[20];
```



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

```
printf("Bonjour %s !", prenom);
```



I/O : puts()

écrit une chaîne de caractères suivie d'un saut de ligne.

```
puts("bonjour");
```

est équivalent à :

```
printf("bonjour\n");
```



Pas d'insertion de variables



I/O : gets()

La fonction gets (abréviation de get string) permet de récupérer une chaîne de caractères saisie.

```
char prenom[20];  
gets(prenom);
```

est équivalent à :

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



Manipulation des chaînes de caractères

Ces fonctions sont disponibles dans la bibliothèque string.h

```
#include <string.h>
```

- strlen
- strcat
- strncat
- strcmp
- strncmp
- strcpy
- strncpy
- strchr
- strrchr
- strstr
- sprintf
- sscanf

Récupérer la longueur d'une chaîne de caractères

On peut compter les caractères jusqu'au marqueur de fin de chaîne :

```
char ch1[50] = "toto";  
int cpt = 0;  
  
for (int i = 0; ch1[i] != '\0'; i++) {  
    cpt++;  
}  
  
printf("%d", cpt);
```

Mais il existe une fonction pour ça...



strlen()

```
strlen(chaine);
```

Renvoie la longueur d'une chaîne de caractères (sans le \0)

```
// Exemple
```

```
char chaine[] = "salut";
```

```
int n = strlen("bonjour"); // n : 7
```

```
int m = strlen(chaine);    // m : 5
```


Concaténer deux chaînes de caractères

On peut recopier chaque élément de la seconde chaîne à la fin de la première, en prévoyant bien d'avoir assez de place et en rajoutant le '\0' :

```
char ch1[13] = "Hello,",  
      ch2[7]  = " world";  
  
for (int i = 6, j = 0; i < 12; i ++, j++) {  
    ch1[i] = ch2[j];  
}  
  
ch1[12] = '\0';  
  
puts(ch1); // Hello, world
```



strcat()

```
strcat(destination, source);
```

Permet de concaténer deux chaînes de caractères (redimensionne destination si nécessaire)

```
// Exemple
char ch1[50] = "bonjour";
char ch2[50] = " monsieur";
strcat(ch1, ch2); // ch1 : "bonjour monsieur"
                  // ch2 : " monsieur"
```



strncat()

```
strncat(destination, source, n);
```

Permet de concaténer deux chaînes de caractères en limitant à **n** le nombre de caractère de source.

```
// Exemple
```

```
char ch1[50] = "bonjour";
```

```
char ch2[50] = " monsieur";
```

```
strncat(ch1, ch2, 5); // ch1 : "bonjour mons"
```

```
                // ch2 : " monsieur"
```

Comparer 2 chaînes de caractères

On serait tenter de faire :

```
char ch1[13] = "Hello", ch2[50];
```

```
gets(ch2);
```



```
if (ch1 == ch2) {  
    printf("How are you ?");  
}  
else {  
    printf("and politeness?");  
}
```



ch1 et ch2 sont les adresses des tableaux, on ne compare donc pas les valeurs qui y sont stockées...

Le résultat sera donc toujours faux !

ON NE FAIT PAS ÇA !

Mais il existe une fonction pour ça...

strcmp()

```
strcmp(chaine1, chaine2);
```

Compare deux chaînes de caractères et renvoie :

- 0 si elle sont identiques,
- 1 si chaîne1 est placée après chaîne2 dans le dictionnaire
- -1 si chaîne1 est placée avant chaîne2 dans le dictionnaire

// Exemple

```
int a;
```

```
a = strcmp("bonjour", "bonjour"); // a : 0
```

```
a = strcmp("bonjour", "banane"); // a : 1
```

```
a = strcmp("bonjour", "bonobo"); // a : -1
```

strncmp()

```
strncmp(chaine1, chaine2, n);
```

Renvoie le même résultat que `strcmp` mais en ne comparant que les `n` premiers caractères des chaînes soumises.

// Exemple

```
int a;
```

```
a = strncmp("bonjour", "bon", 3);      // a : 0
```

```
a = strncmp("bonjour", "bonbon", 3);   // a : 0
```

Affecter une valeur à une chaîne de caractères

Ceci est une initialisation, c'est un autre mécanisme et c'est autorisé !

On serait tenter de faire :

```
char ch1[13] = "Hello", ch2[50];
```

```
ch2 = " world";  
// ou  
ch1 = ch2;
```



ch1 et ch2 sont les adresses des tableaux, on ne peut pas affecter des adresses en mémoire comme cela.

ON NE FAIT PAS ÇA !

Mais il existe une fonction pour ça...

strcpy()

```
strcpy(destination, source);
```

Copie une chaîne dans une autre.



La chaîne destination doit avoir une longueur suffisante pour accueillir la chaîne source.

```
// Exemple :
```

```
char ch1[] = "bonjour";
```

```
strcpy(ch1, "spirou");    // ch1 : "spirou"
```

```
strcpy(ch1, "schtroumpf"); // Erreur de compilation !
```


strncpy()

```
strncpy(destination, source, n);
```

Copie les **n** premiers caractères d'une chaîne dans une autre.



La chaîne `destination` doit au moins avoir une longueur de **n**.

Les caractères non remplacés persistent (dépendant du compilateur).

Le `\0` peut être écrasé.

// Exemple :

```
char ch1[] = "bonjour";
```

```
strncpy(ch1, "schtroumpf", strlen(ch1)); // ch1 : "schtrou"
```

```
strncpy(ch1, "spip", 3); // ch1 : "spitrou"
```





strchr()

```
strchr(chaine, caractere);
```

Retourne la position de la première occurrence d'un caractère dans une chaîne de caractères.

Renvoie NULL si le caractère est absent de la chaîne, sinon l'adresse du caractère dans la chaîne.

```
// Exemple :
```

```
char ch1[] = "bonjour";
```

```
if (strchr(ch1, 'j') != NULL) {
```

```
    printf ("le caractere 'j' est dans la chaine");
```

```
}
```



strrchr()

```
strchr(chaine, caractère);
```

Retourne la position de la première occurrence d'un caractère dans une chaîne de caractères **en partant de la droite**.

Renvoie NULL si le caractère est absent de la chaîne, sinon l'adresse du caractère dans la chaîne.

```
// Exemple :  
char ch1[] = "bonjour";  
int a = strchr(ch1, 'o'); // a : 61fe19  
a = strrchr(ch1, 'o');    // a : 61fe1c
```



strstr()

```
strstr(chaine, sous_chaine);
```

Retourne la position de la première occurrence d'une sous_chaine de caractères dans une chaîne de caractères.

Renvoie NULL si la sous_chaine est absente de la chaîne, sinon l'adresse de la sous_chaine dans la chaîne.

```
// Exemple :  
char ch1[] = "bonjour";  
int a = strstr(ch1, "jour"); // a : 61fe1b  
if (a != NULL) {  
    printf("la sous-chaîne a été trouvée !");  
}
```




sprintf()

```
sprintf(chaine, format, variable 1,..., variable n);
```

permet de stocker le résultat d'un printf dans une chaîne de caractères.

```
// Exemple :
```

```
int n = 15,
```

```
float p = 73.35;
```

```
char tab[100];
```

```
sprintf(tab, "%d articles coutent %.2f euros", n, p);
```

```
printf("%s",tab);
```

```
// 15 articles coutent 73.35 euros
```

sscanf()

```
sscanf(chaine, format, adresse 1,..., adresse n);
```

permet d'exécuter un « scanf » sur une chaîne de caractères.

```
// Exemple :  
char input[] = "25/12/1984";  
int d, m, y;  
sscanf(input, "%d/%d/%d", &d, &m, &y);  
printf("jour : %d, mois : %d, année : %d\n", d, m, y);  
// jour : 25, mois :12, année : 1984
```

I/O : scanf

Arrêt prématuré et tampon : scanf utilise un tampon pour stocker les saisies utilisateur : tant qu'il reste dedans des valeurs éligibles, il utilise le tampon plutôt que de redemander une saisie.

```
int a;  
char b;  
scanf("%d", &a);           // 31287 + entrée pour valider → tampon : 

|   |   |   |   |   |    |
|---|---|---|---|---|----|
| 3 | 1 | 2 | 8 | 7 | \n |
|---|---|---|---|---|----|

  
  
scanf("%c", &b);           // tampon contient 

|    |
|----|
| \n |
|----|

 = char = %c           %d = 1 entier  
                           // ↳ scanf ne redemande pas de saisie !           à stocker dans a  
  
printf("%d\n", a);         // 31287  
printf("%c", b);           // \n
```

NE PAS utiliser `fflush(stdin);`



Solution avec gets/sscanf !

```
int a;  
char b;  
char input[20];  
  
gets(input);  
sscanf(input, "%d", &a);  
gets(input);  
sscanf(input, "%c", &b);  
  
printf("%d\n", a);  
printf("%c", b);
```





Manipulation des chaînes de caractères

Ces fonctions sont disponibles dans la bibliothèque stdlib.h

```
#include <stdlib.h>
```

- atoi
- atol
- atof

atoi() / atol() / atof()

`atoi` convertit une chaîne de caractères en un int si possible sinon 0.

```
char chaine[] = "123";  
int i = atoi(chaine);           // i = 123
```

`atol` convertit une chaîne de caractères en un long si possible sinon 0.

```
char chaine[] = "1234";  
long l = atol(chaine);          // l = 1234
```

`atof` convertit une chaîne de caractères en un float si possible sinon 0.

```
char chaine[] = "123.4"  
float f = atof(chaine);         // f = 123.400000
```