Le C — Chaînes de caractères

tinyurl.com/2s3xnn8m

Nathanael Demacon

Chaînes de caractères

Vous connaissez et utilisez déjà les chaînes de caractères.

```c
#include <stdio.h>

int main()
{
   printf("Hello, world!\n");

   int age;
   scanf("%d", &age);

   return 0;
}
```

Lien avec les tableaux

Une chaîne de caractère est un tableau de caractères. On peut donc le stocker dans un tableau ou dans un pointeur.

Une chaîne de caractères déclarée directement de cette façon est une chaîne de caractères dite “statique”.

Lien avec les tableaux

Une chaîne de caractère est un tableau de caractères. On peut donc le stocker dans un tableau ou dans un pointeur.

```c
int main()
{
    char str_arr[] = "hello";
    char *str_ptr = "hello";

    return 0;
}
```

Lorsqu'une chaîne de caractères **statique** est mise dans un pointeur, elle est **non modifiable**, contrairement au format tableau au-dessus qui fait une copie de la chaîne.

Lien avec les tableaux

Astuce : pour s'assurer qu'on ne modifie pas la chaîne de caractères en mode pointeur, on peut rajouter la contrainte "const" au type derrière le pointeur. Le compilateur se chargera de nous arrêter.

```c
int main()
{
    const char *str = "hello";
    str[0] = 'H'; // error

    return 0;
}
```

Lien avec les tableaux

On peut donner une taille au tableau de destination. Attention à ce qu'elle ne soit pas plus petite que la taille de la chaîne de caractère.

```c
int main()
{
    char a[6] = "hello";

    return 0;
}
```

Lien avec les tableaux

Il est possible de déclarer une chaîne de caractère avec une liste d'initialisation.

```c
int main()
{
    char a[] = "hello";
    char b[] = {'h', 'e', 'l', 'l', 'o', '\0'};

    return 0;
}
```

Le caractère '\0' signale la fin de la chaîne de caractères. Ce caractère est essentiel au fonctionnement des chaînes de caractères en C.

Manipulations

En C, impossible d'utiliser "==" pour comparer deux chaînes de caractères, on compare les pointeurs. Pour régler le problème, il existe une bibliothèque standard : "string.h".

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

int main()
{
    const char *original_password = "1234";
    char password[10] = {0};

    printf("Enter password: ");
    scanf("%s", password);

    if (strcmp(password, original_password) == 0)
    {
        printf("Access granted\n");
    }

    return 0;
}
```

La fonction strcmp compare deux chaînes de caractères caractère par caractère. Si elles sont égales alors la fonction renvoie 0, sinon elle renvoie la différence entre les deux caractères.

Sortie	Description
0	Les deux chaînes sont égales.
> 0	Le premier caractère non-égale de la première chaîne de caractères est plus grand que celui de la seconde.
< 0	Le premier caractère non-égale de la première chaîne de caractères est plus petit que celui de la seconde.

Manipulations

Pour copier une chaîne de caractères dans une autre, on peut utiliser la fonction “strcpy”.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
   const char *a = "1234";

   char *b = malloc(sizeof(char) * 10);
   strcpy(b, a);

   printf("%s\n", b); // "1234"

   return 0;
}
```

strcpy copie aussi le caractère \0 à la fin de la chaîne source. On a donc pas besoin de l’ajouter manuellement.

Manipulations

La fonction “strncpy” peut être utilisée pour limiter le nombre de caractères copiés.

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

int main()
{
   const char *a = "1234";

   char b[] = "abcdefghij";
   strncpy(b, a, 4);

   printf("%s\n", b); // "1234efghij"

   return 0;
}
```

Attention à rajouter le caractère \0 manuellement si vous utilisez cette méthode dans un string non initialisé à 0.

Manipulations

La fonction “strcat” peut être utilisée pour fusionner deux chaînes de caractères entre elles.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
   const char *a = "1234";
   const char *b = "abcde";

   char *c = malloc(sizeof(char) * (strlen(a) + strlen(b) + 1));
   strcpy(c, a);
   strcat(c, b);

   printf("%s\n", c); // "1234abcde"

   return 0;
}
```

Attention de s'assurer d'avoir suffisamment de mémoire pour stocker la chaîne actuelle + la nouvelle chaîne.

Manipulations

Pour connaître la taille d'une chaîne, on peut utiliser la fonction “strlen”.

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

int main()
{
   const char *a = "1234";

   printf("%lu", strlen(a)); // 4

   return 0;
}
```

Manipulations

Voici un exemple de comment réécrire la fonction “strlen” en C.

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

unsigned long my_strlen(const char *str)
{
    unsigned long length = 0;

    while (*str != '\0')
    {
        length++;
        str++;
    }

    return length;
}

int main()
{
    char str[] = "1234";

    printf("%lu", my_strlen(str)); // 4

    return 0;
}
```

Table ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Table ASCII

Chaque caractère en C est représenté par la table ASCII. Un caractère est donc une valeur entière auquel on peut appliquer des opérations arithmétiques.

```c
#include <stdio.h>

int main()
{
   char c = 'a';

   printf("%c\n", c);     // a
   printf("%c\n", c + 1); // b

   return 0;
}
```