

TP : Chaine de caractères

Charles Yaiche avec l'aide de Abdul Farouk

charles.yaiche.lf@gmail.com

Architecture de rendu



Les fonctions seront écrites dans dans le fichier main.c en dessous de la fonction main.

Bibliothèque autorisé :

stdio.h, stdlib.h

Structure du devoir

```
#include <stdio.h>

//Prototype des fonctions
size_t my_strlen(char *s);
void my_strupcase(char *s);

int main() {
    //TODO test des fonctions
    return 0;
}

//Exercice 1
size_t my_strlen(char *s){
    //TODO
}

//Exercice 2
void my_strupcase(char *s){
    //TODO
}
```



Exercice 1(my_strlen)

Description	
Ecrire une fonction qui détermine la taille d'une chaîne de caractère	
Prototype abstrait	Prototype de fonction
my_strlen : Tableau[Caractère] -> Entier	size_t my_strlen(char *s);
Exemples	Résultat
<pre>int a = my_strlen("salut"); printf("%d \n", a);</pre>	<pre>//affiche : 5</pre>
Conseils	
<ul style="list-style-type: none">- une chaîne de caractère est un tableau de caractère finissant par le caractère '\0'- size_t est un type Entier positif ou nul (ne peut pas être négatif) souvent utilisé pour les données gérées en mémoire	



Exercice 2 (my_strupcase)

Description	
la fonction change toutes les lettres minuscule en majuscule d'une string (chaine de caractère)	
Prototype abstrait	Prototype de fonction
my_strupcase : Tableau[Caractères] -> Vide	void my_strupcase(char str[]);
Exemples	valeur
<pre>char *a = "tototo"; my_strupcase(a); printf("%s", a);</pre>	<pre>//affiche TOTOTO</pre>
Conseils	
regarder du coté des conversion de la table ascii (google table ascii)	



Exercice 3 (my_strlowcase)

Description	
la fonction change toutes les lettres minuscule en majuscule d'une string	
Prototype abstrait	Prototype de fonction
my_strlowcase : Tableau[Caractères] -> Vide	void my_strlowcase(char *str);
Exemples	valeur
<pre>char *a = "ToTOTO"; my_strlowcase(a); printf("%s", a);</pre>	<pre>//affiche tototo</pre>
Conseils	



Exercice 4 (my_strcmp)

Description	
La fonction strcmp() compare les deux chaînes s1 et s2. Elle renvoie un entier négatif, nul, ou positif, si s1 est respectivement inférieure, égale ou supérieure à s2.	
Prototype abstrait	Prototype de fonction
my_strcmp : Tableau[Caractère] x Tableau[Caractère] -> Entier	int my_strcmp(char s1[], char s2[]);
Exemples	valeur
my_strcmp("a", "a"); // returns 0 as ASCII value of "a" and "a" are same i.e 97 my_strcmp("a", "b"); // returns -1 as ASCII value of "a" (97) is less than "b" (98) my_strcmp("a", "c"); // returns -1 as ASCII value of "a" (97) is less than "c" (99) my_strcmp("z", "d"); // returns 1 as ASCII value of "z" (122) is greater than "d" (100) my_strcmp("abc", "abe"); // returns -1 as ASCII value of "c" (99) is less than "e" (101) my_strcmp("apples", "apple"); // returns 1 as ASCII value of "s" (115) is greater than "0" (101)	
Conseils	



Exercice 5 (my_strncmp)

Description

La fonction `strncmp()` est identique sauf qu'elle ne compare que les n (au plus) premiers caractères de $s1$ et $s2$.

Prototype abstrait

`my_strncmp` : Tableau[Caractère] x
Tableau[Caractère] x Entier -> Entier

Prototype de fonction

`int my_strncmp(char s1[], char s2[],
size_t n);`

Exemples

```
char *a = "Tot";  
char *b = "Tota";  
int c = my_strncmp(a, b, 2);  
printf("%d", c);
```

valeur

```
//affiche  
0
```

Conseils



Exercice 6 (my_strcasecmp)

Description

La fonction `strcasecmp()` compare les deux chaînes *s1* et *s2*, en ignorant les différences entre majuscules et minuscules. Elle renvoie un entier négatif, nul, ou positif, si la chaîne *s1* est respectivement inférieure, égale ou supérieure à *s2*.

Prototype abstrait

`my_strcasecmp : Tableau[Caractère] x
Tableau[Caractère] -> Entier`

Prototype de fonction

`int my_strcasecmp(char s1[], char
s2[]);`

Exemples

```
char *a = "To";  
char *b = "tOt";  
int c = my_strcasecmp(a, b);  
printf("%d", c);
```

valeur

```
//affiche  
0
```

Conseils



Exercice 7 (my_strncasecmp)

Description	
La fonction <code>strncasecmp()</code> est similaire, à la différence qu'elle ne prend en compte que les n premiers caractères de $s1$.	
Prototype abstrait	Prototype de fonction
<code>my_strncasecmp : Tableau[Caractère] x Tableau[Caractere] x Entier -> Entier</code>	<code>int my_strncasecmp (char s1[], char s2[], size_t n);</code>
Exemples	valeur
<code>char *a = "To"; char *b = "tata"; int c = my_strncmp(a, b, 2); printf("%d", c);</code>	<code>//affiche 0</code>
Conseils	



Exercice 8 (my_strspn)

Description

Renvoie la longueur de la plus grande sous-chaîne (en partant du début de la chaîne initiale) ne contenant que des caractères spécifiés dans la liste des caractères acceptés.

Prototype abstrait

my_strspn : Tableau[Caractère] x
Tableau[Caractère] -> Entier

Prototype de fonction

```
size_t my_strspn ( char s[], char  
accept []);
```

Exemples

```
char *a = "to1234";  
char *b = 'abcdefghijklmnopqrstuvwxyz';  
int c = my_strspn(a, b);  
printf("%d", c);
```

valeur

```
//affiche  
2
```

Conseils

chercher des exemples sur internet de la fonction strspn



Exercice 9 (my_strcspn)

Description	
Renvoie la longueur de la plus grande sous-chaine (en partant du début de la chaîne initiale) ne contenant aucun des caractères spécifiés dans la liste des caractères en rejet.	
Prototype abstrait	Prototype de fonction
my_strcspn : Tableau[Caractère] x Tableau[Caractere] -> Entier	size_t my_strcspn (char s[], char reject[]);
Exemples	valeur
char *a = "ta123ta"; char *b = 'ta'; int c = my_strncmp(a, b); printf("%d", c);	//affiche 3
Conseils	
cherchez des exemples sur internet de la fonction strcspn	



Exercice 10 (my_strstr)

Description	
<p>la fonction <code>my_strstr</code> recherche la première occurrence d'une sous-chaîne (paramètre <code>substring</code>) dans la chaîne de caractères principale (paramètre <code>fullString</code>).</p> <p>Si la sous-chaîne est trouvée dans la chaîne principale, la fonction renvoi l'index de sa première occurrence. Dans le cas contraire, -1 vous sera renvoyé.</p>	
Prototype abstrait	Prototype de fonction
<code>my_strstr : Tableau[Caractère] x Tableau[Caractère] -> Entier</code>	<code>int my_strstr(char haystack[], char needle[]);</code>
Exemples	valeur
<pre>char *b = "aaatoeauie"; char *a = "to"; int c = my_strstr(a, b); printf("%d", c);</pre>	<pre>//affiche 3</pre>
Conseils	



Exercice 11 (my_atoi)

Description	
La fonction <code>atoi()</code> convertit le début de la chaîne pointée par <i>nptr</i> en entier de type <i>int</i> . Le résultat est identique à un appel	
Prototype abstrait	Prototype de fonction
<code>my_atoi : Tableau[Caractère] -> Entier</code>	<code>int my_atoi(char str[]);</code>
Exemples	valeur
<pre>char *a = "1234"; int c = atoi(a); printf("%d", c);</pre>	<pre>//affiche 1234</pre>
Conseils	
Regarder dans la table ascii pour les conversion	



Exercice 12 (my_atoi_base)

Description	
my_atoi_base à le même comportement que my_atoi, à la différence que my_atoi_base spécifie la base utilisé pour la conversion	
Prototype abstrait	Prototype de fonction
my_atoi_base : Tableau[Caractère] x Tableau[Caractère] -> Entier	int my_atoi_base(char str[], char base[]);
Exemples	valeur
my_atoi_base("ff", "0123456789 abcdef"); my_atoi_base("-ff", "0123456789 abcdef"); my_atoi_base("77", "01234567"); my_atoi_base("WQWW", "QW");	// doit retourner 255 // doit retourner -255 // doit retourner 63 // doit retourner 11
Conseils	
la table ascii est ta meilleure amie.	



Exercice 13 (sieve of Eratosthenes)

Description	
Le sieve of Eratosthenes est un procédé qui permet de trouver tous les nombres premiers inférieurs à un certain entier naturel donné N.	
Prototype abstrait	Prototype de fonction
sieve : Entier -> Vide	void sieve(int n);
Exemples	valeur
sieve(11); sieve(0); sieve(42);	//affiche 2 3 5 7 11 //affiche rien affiche 2 3 5 7 11 13 17 19 23 29 31 37 41
Conseils	



Exercice 14 (flyTo)

Description

l'objectif est d'écrire une procédure qui tri un emplois du temps de vol, chaque temps est associé avec une destination, par exemple "23h42 Tokyo"

Prototype abstrait

fly : Tableau[caractère] -> vide

Prototype de fonction

void fly(char *s);

Exemples

valeur

fly("8h00 Berlin; 9h0 Tokyo; 10h Paris");

//affiche
8h00 Berlin
9h0 Tokyo
10h Paris

fly("23h Berlin; 12h Tokyo; 11h Paris");

//affiche
11h Paris
12h Tokyo
23h Berlin

fly("8h10 Berlin; 8h12 Tokyo; 8h11 Paris");

//affiche
8h10 Berlin
8h11 Paris
8h12 Tokyo

fly("8h42 Berlin; 8h42 Paris; 8h42 Tokyo")

//affiche
8h42 Berlin
8h42 Paris
8h42 Tokyo

Conseils»



Exercice 14 (freq_analysis)

Description

freq_analysis est l'étude de la répartition des lettres dans un texte. Elle facilite le déchiffrement de messages chiffrés par substitution en se basant sur le fait que certaines lettres ou combinaisons de lettres n'apparaissent pas aussi souvent que d'autres dans les langages : en français, E est la lettre la plus utilisée, alors que W l'est beaucoup moins.

Prototype abstrait

freq_analysis: Tableau[Caractère] x
Tableau[Caractere] -> Vide

Prototype de fonction

void freq_analysis(char text[], char
table[]);

Exemples

valeur

freq_analysis("AAB", "XY");

//affiche
A X
B Y

freq_analysis("ABBCCDDDD", "ABCD");

//affiche
A D
B C
C B
D A

freq_analysis("FXOWFFOWOFF", "ABCD");

//affiche
F A
O B
W C
X D

Conseils

BONUS (+1 point) :

<pre>main() { int A = 1; int B = 2; int C = 3; int *P1, *P2; P1=&A; P2=&C; *P1=(*P2)++; P1=P2; P2=&B; *P1-=*P2; ++*P2; *P1*=*P2; A=++*P2**P1; P1=&A; *P2=*P1/=*P2; return 0; }</pre>	complétez-le pour chaque instruction du programme ci-contre.					
		<u>A</u>	<u>B</u>	<u>C</u>	<u>P1</u>	<u>P2</u>
	Init.	1	2	3	/	/
	P1=&A	1	2	3	&A	/
	P2=&C					
	*P1=(*P2)++					
	P1=P2					
	P2=&B					
	*P1-=*P2					
	++*P2					
	P1=*P2					
	A=++*P2**P1					
	P1=&A					
	*P2=*P1/=*P2					