

Programmation en C

Devoir de Groupe : Chaînes de caractère - Pointeur

Pour mener à bien cet atelier, chaque étudiant devra créer un projet par exercice en suivant la notation : *Atl04_ExoX_Groupe_Y*.

Le *X* doit être remplacé le numéro correspondant de l'exercice. *Y* doit être remplacé par votre numéro de groupe.

Exercice 1

Ecrire un programme C qui lit une chaîne de caractères et vérifie si elle est palindrome ou non.

On rappelle qu'une chaîne de caractères est dite palindrome, si elle se lit de la même manière dans les deux sens. Exemple: *non*, *touot* et *1234321* sont toutes des chaînes de caractères palindromes.

Exercice 2

Ecrire un programme C qui utilise la notion de pointeur pour lire deux entiers et calculer leur somme.

Exercice 3

Ecrire un programme C qui lit deux chaînes de caractères et les affiche dans l'ordre alphabétique, en utilisant les deux méthodes suivantes:

Par exemple, si on donne en entrée les deux chaînes suivantes: *acb* et *abcd*, le programme doit afficher la chaîne *abcd* puis *acb*.

Exercice 4

Ecrire un programme C qui utilise la notion de pointeur pour la permuter le contenu de deux variables de type *char*.

Exercice 5

Ecrire un programme C qui lit deux chaînes de caractères et permute leurs contenus

Exercice 6

Ecrire un programme C qui lit une chaîne de caractères, et transforme chaque caractère majuscule en minuscule et vice versa.

Exercice 7

Ecrire un programme C qui remplit un tableau d'entiers et calcule la somme de ses éléments en utilisant un pointeur pour son parcours.

Exercice 8

Ecrire un programme C qui lit deux chaînes de caractères et vérifie si la deuxième est une sous chaîne de la première ou non.

Exercice 9

Ecrire un programme C qui lit une chaîne de caractères et affiche cette chaîne à partir de la première occurrence d'un caractère entré par l'utilisateur.

Mémo

The C library function **int tolower(int c)** converts a given letter to lowercase.

```
int tolower(int c);
```







- **c** – This is the letter to be converted to lowercase.




This function returns lowercase equivalent to **c**, if such value exists, else **c** remains unchanged. The value is returned as an **int** value that can be implicitly casted to **char**.

The C library function **int toupper(int c)** converts a given letter to uppercase.

Following are the functions defined in the header `string.h` –

Sr.No.	Function & Description
1	void *memchr(const void *str, int c, size_t n) ⓘ Searches for the first occurrence of the character c (an unsigned char) in the first n bytes of the string pointed to, by the argument <i>str</i> .
2	int memcmp(const void *str1, const void *str2, size_t n) ⓘ Compares the first n bytes of <i>str1</i> and <i>str2</i> .
3	void *memcpy(void *dest, const void *src, size_t n) ⓘ Copies n characters from <i>src</i> to <i>dest</i> .
4	void *memmove(void *dest, const void *src, size_t n) ⓘ Another function to copy n characters from <i>str2</i> to <i>str1</i> .
5	void *memset(void *str, int c, size_t n) ⓘ Copies the character c (an unsigned char) to the first n characters of the string pointed to, by the argument <i>str</i> .
6	char *strcat(char *dest, const char *src) ⓘ Appends the string pointed to, by <i>src</i> to the end of the string pointed to by <i>dest</i> .
7	char *strncat(char *dest, const char *src, size_t n) ⓘ Appends the string pointed to, by <i>src</i> to the end of the string pointed to, by <i>dest</i> up to n characters long.
8	char *strchr(const char *str, int c) ⓘ Searches for the first occurrence of the character c (an unsigned char) in the string pointed to, by the argument <i>str</i> .
9	int strcmp(const char *str1, const char *str2) ⓘ Compares the string pointed to, by <i>str1</i> to the string pointed to by <i>str2</i> .

10	int strncmp(const char *str1, const char *str2, size_t n)  Compares at most the first n bytes of <i>str1</i> and <i>str2</i> .
11	int strcoll(const char *str1, const char *str2)  Compares string <i>str1</i> to <i>str2</i> . The result is dependent on the LC_COLLATE setting of the location.
12	char *strcpy(char *dest, const char *src)  Copies the string pointed to, by <i>src</i> to <i>dest</i> .
13	char *strncpy(char *dest, const char *src, size_t n)  Copies up to n characters from the string pointed to, by <i>src</i> to <i>dest</i> .
14	size_t strcspn(const char *str1, const char *str2)  Calculates the length of the initial segment of <i>str1</i> which consists entirely of characters not in <i>str2</i> .
15	char *strerror(int errnum)  Searches an internal array for the error number <i>errnum</i> and returns a pointer to an error message string.
16	size_t strlen(const char *str)  Computes the length of the string <i>str</i> up to but not including the terminating null character.
17	char *strpbrk(const char *str1, const char *str2)  Finds the first character in the string <i>str1</i> that matches any character specified in <i>str2</i> .
18	char *strrchr(const char *str, int c)  Searches for the last occurrence of the character <i>c</i> (an unsigned char) in the string pointed to by the argument <i>str</i> .
19	size_t strspn(const char *str1, const char *str2)  Calculates the length of the initial segment of <i>str1</i> which consists entirely of characters in <i>str2</i> .

20	char *strstr(const char *haystack, const char *needle)  Finds the first occurrence of the entire string <i>needle</i> (not including the terminating null character) which appears in the string <i>haystack</i> .
21	char *strtok(char *str, const char *delim)  Breaks string <i>str</i> into a series of tokens separated by <i>delim</i> .
22	size_t strxfrm(char *dest, const char *src, size_t n)  Transforms the first <i>n</i> characters of the string <i>src</i> into current locale and places them in the string <i>dest</i> .