## *Overview of the C-string library*

Include the standard header <cstring> to declare a number of functions that help you manipulate C-strings and other arrays of characters.

**NULL**
#define NULL <either 0, 0L, or (void *)0> [often 0 in C++, always (void *)0 in C]

The macro yields a null pointer constant that is usable as an address constant expression. In C++, the only integer value that may be coerced into a pointer is 0. You cannot coerce a pointer into an integer.

**memchr**
void *memchr(const void *s, int c, size_t n); [not in C++]
const void *memchr(const void *s, int c, size_t n); [C++ only]
void *memchr(void *s, int c, size_t n); [C++ only]

The function searches for the first element of an array of unsigned char, beginning at the address s with size n, that equals (unsigned char)c. If successful, it returns the address of the matching element; otherwise, it returns a null pointer.

**memcmp**
int memcmp(const void *s1, const void *s2, size_t n);

The function compares successive elements from two arrays of unsigned char, beginning at the addresses s1 and s2 (both of size n), until it finds elements that are not equal:

If all elements are equal, the function returns zero.
If the differing element from s1 is greater than the element from s2, the function returns a positive number.
Otherwise, the function returns a negative number.

**memcpy**
void *memcpy(void *s1, const void *s2, size_t n);

The function copies the array of char beginning at the address s2 to the array of char beginning at the address s1 (both of size n). It returns s1. The elements of the arrays can be accessed and stored in any order.

**memmove**
void *memmove(void *s1, const void *s2, size_t n);

The function copies the array of char beginning at s2 to the array of char beginning at s1 (both of size n). It returns s1. If the arrays overlap, the function accesses each of the element values from s2 before it stores a new value in that element, so the copy is not corrupted.

**memset**
void *memset(void *s, int c, size_t n);

The function stores (unsigned char)c in each of the elements of the array of unsigned char beginning at s, with size n. It returns s.

**size_t**
typedef ui-type size_t;

The type is the unsigned integer type ui-type of an object that you declare to store the result of the sizeof operator.

**strcat**
char *strcat(char *s1, const char *s2);

The function copies the string s2, including its terminating null character, to successive elements of the array of char that stores the string s1, beginning with the element that stores the terminating null character of s1. It returns s1.

**strchr**
char *strchr(const char *s, int c); [not in C++]
const char *strchr(const char *s, int c); [C++ only]
char *strchr(char *s, int c); [C++ only]

The function searches for the first element of the string s that equals (char)c. It considers the terminating null character as part of the string. If successful, the function returns the address of the matching element; otherwise, it returns a null pointer.

**strcmp**
int strcmp(const char *s1, const char *s2);

The function compares successive elements from two strings, s1 and s2, until it finds elements that are not equal.

If all elements are equal, the function returns zero.
If the differing element from s1 is greater than the element from s2 (both taken as unsigned char), the function returns a positive number.
Otherwise, the function returns a negative number.

**strcoll**
int strcoll(const char *s1, const char *s2);

The function compares two strings, s1 and s2, using a comparison rule that depends on the current locale. If s1 compares greater than s2 by this rule, the function returns a

positive number. If the two strings compare equal, it returns zero. Otherwise, it returns a negative number.

**strcpy**
char *strcpy(char *s1, const char *s2);

The function copies the string s2, including its terminating null character, to successive elements of the array of char whose first element has the address s1. It returns s1.

**strcspn**
size_t strcspn(const char *s1, const char *s2);

The function searches for the first element s1[i] in the string s1 that equals any one of the elements of the string s2 and returns i. Each terminating null character is considered part of its string.

**strerror**
char *strerror(int errcode);

The function returns a pointer to an internal static-duration object containing the message string corresponding to the error code errcode. The program must not alter any of the values stored in this object. A later call to strerror can alter the value stored in this object.

**strlen**
size_t strlen(const char *s);

The function returns the number of characters in the string s, not including its terminating null character.

**strncat**
char *strncat(char *s1, const char *s2, size_t n);

The function copies the string s2, not including its terminating null character, to successive elements of the array of char that stores the string s1, beginning with the element that stores the terminating null character of s1. The function copies no more than n characters from s2. It then stores a null character, in the next element to be altered in s1, and returns s1.

**strncmp**
int strncmp(const char *s1, const char *s2, size_t n);

The function compares successive elements from two strings, s1 and s2, until it finds elements that are not equal or until it has compared the first n elements of the two strings.

If all elements are equal, the function returns zero.

If the differing element from s1 is greater than the element from s2 (both taken as unsigned char), the function returns a positive number.
Otherwise, it returns a negative number.

**strncpy**
char *strncpy(char *s1, const char *s2, size_t n);

The function copies the string s2, not including its terminating null character, to successive elements of the array of char whose first element has the address s1. It copies no more than n characters from s2. The function then stores zero or more null characters in the next elements to be altered in s1 until it stores a total of n characters. It returns s1.

**strpbrk**
char *strpbrk(const char *s1, const char *s2); [not in C++]
const char *strpbrk(const char *s1, const char *s2); [C++ only]
char *strpbrk(char *s1, const char *s2); [C++ only]

The function searches for the first element s1[i] in the string s1 that equals any one of the elements of the string s2. It considers each terminating null character as part of its string. If s1[i] is not the terminating null character, the function returns &s1[i]; otherwise, it returns a null pointer.

**strrchr**
char *strrchr(const char *s, int c); [not in C++]
const char *strrchr(const char *s, int c); [C++ only]
char *strrchr(char *s, int c); [C++ only]

The function searches for the last element of the string s that equals (char)c. It considers the terminating null character as part of the string. If successful, the function returns the address of the matching element; otherwise, it returns a null pointer.

**strspn**
size_t strspn(const char *s1, const char *s2);

The function searches for the first element s1[i] in the string s1 that equals none of the elements of the string s2 and returns i. It considers the terminating null character as part of the string s1 only.

**strstr**
char *strstr(const char *s1, const char *s2); [not in C++]
const char *strstr(const char *s1, const char *s2); [C++ only]
char *strstr(char *s1, const char *s2); [C++ only]

The function searches for the first sequence of elements in the string s1 that matches the sequence of elements in the string s2, not including its terminating null character. If

successful, the function returns the address of the matching first element; otherwise, it returns a null pointer.

**strtok**
char *strtok(char *s1, const char *s2);

If s1 is not a null pointer, the function begins a search of the string s1. Otherwise, it begins a search of the string whose address was last stored in an internal static-duration object on an earlier call to the function, as described below. The search proceeds as follows:

The function searches the string for begin, the address of the first element that equals none of the elements of the string s2 (a set of token separators). It considers the terminating null character as part of the search string only.

If the search does not find an element, the function stores the address of the terminating null character in the internal static-duration object (so that a subsequent search beginning with that address will fail) and returns a null pointer. Otherwise, the function searches from begin for end, the address of the first element that equals any one of the elements of the string s2. It again considers the terminating null character as part of the search string only.

If the search does not find an element, the function stores the address of the terminating null character in the internal static-duration object. Otherwise, it stores a null character in the element whose address is end. Then it stores the address of the next element after end in the internal static-duration object (so that a subsequent search beginning with that address will continue with the remaining elements of the string) and returns begin.

**strxfrm**
size_t strxfrm(char *s1, const char *s2, size_t n);

The function stores a string in the array of char whose first element has the address s1. It stores no more than n characters, including the terminating null character, and returns the number of characters needed to represent the entire string, not including the terminating null character. If the value returned is n or greater, the values stored in the array are indeterminate. (If n is zero, s1 can be a null pointer.)

strxfrm generates the string it stores from the string s2 by using a transformation rule that depends on the current locale. For example, if x is a transformation of s1 and y is a transformation of s2, then strcmp(x, y) returns the same value as strcoll(s1, s2).