**Funções de exame de strings:**

**• strlen: size\_t strlen( const char \*str );**

Essa função recebe um ponteiro que indica uma string e retorna quantos caracteres essa string possui.

**• strcmp: int strcmp( const char \*lhs, const char \*rhs );**

Essa função compara duas strings segundo sua ordem alfabética e retorna um inteiro. Se esse inteiro for negativo, é porque a primeira string é menor que a segunda. Se retornar um inteiro positivo, é porque a segunda string é maior que a segunda. Se retornar 0, é porque as strings são idênticas.

**• strncmp: int strncmp( const char \*lhs, const char \*rhs, size\_t count );**

Faz a mesma comparação da strcmp, mas ao invés de comparar toda a string, compara somente os 'count' primeiros caracteres.

**• strchr: char \*strchr( const char \*str, int ch );**

Retorna um ponteiro para a localização em que o caractere 'ch' aparece na string pela primeira vez na string apontada por \*str, ou NULL se não encontrar.

**• strrchr: char \*strrchr( const char \*str, int ch );**

Faz a mesma coisa da função anterior, mas ao invés de localizar a primeira ocorrência de 'ch', localiza e retorna a última ocorrência.

**• strspn: size\_t strspn( const char \*dest, const char \*src );**

Retorna o tamanho máximo do primeiro segmento na string 'dest' que consiste de elementos da string 'src'.

**• strcspn: size\_t strcspn( const char \*dest, const char \*src );**

É o contrário da anterior, ou seja, retorna o tamanho máximo do segmento inicial na string 'dest' que consiste somente de elementos que NÃO ESTÃO na string 'src'.

**• strpbrk: char\* strpbrk( const char\* dest, const char\* str );**

Retorna o primeiro caractere que está em ambas strings recebidas, ou NULL se não existe caractere comum.

**• strstr: char \*strstr( const char\* str, const char\* substr );**

Retorna um ponteiro que indica a primeira ocorrencia da string 'substr' na string 'str'.

**Funções de manipulação de strings**

**• strcpy: char \*strcpy( char \*dest, const char \*src );**

Copia a string 'src' para a string 'dest', e retorna um ponteiro para 'dest'.

**• strncpy: char \*strncpy( char \*dest, const char \*src, size\_t count );**

Faz a mesma coisa da strcpy, mas em vez de copiar todos os elementos, copia somente os 'count' primeiros elementos da 'src' para a 'dest'.

**• strcat: char \*strcat( char \*dest, const char \*src );**

Copia (concatena) a string 'src' ao final da string 'dest'.

**• strncat: char \*strncat( char \*dest, const char \*src, size\_t count );**

Copia (concatena) 'count' elementos da string 'src' no final da string 'dest'.

**• memset: void\* memset( void\* dest, int ch, size\_t count );**

Coloca o caractere 'ch' nas 'count' primeiras posições da string 'dest'.

**• memcpy: void\* memcpy( void\* dest, const void\* src, size\_t count );**

Copia os 'count' primeiros caracteres da string 'src' e coloca nas primeiras 'count' da string 'dest'.

**• memcmp: int memcmp( const void\* lhs, const void\* rhs, size\_t count );**

Compara os 'count' primeiros caracteres das strings. Retorna negativo se os 'count' primeiros caracteres da primeira string forem menores que a segunda. Retorna positivo se os 'count' primeiros caracteres da primeira string forem maiores que a segunda. Retorna 0 se os 'count' primeiros caracteres das duas strings são iguais.

**• memchr: void\* memchr( const void\* ptr, int ch, size\_t count );**

Acha a primeira ocorrência do caractere 'ch' nos primeiras 'count' elementos da string 'ptr' e devolve o endereço da ocorrência ou NULL.