

# Almog String Manipulation

Generated by Doxygen 1.9.1



---

<b>1 File Index</b>	<b>1</b>
1.1 File List . . . . .	1
<b>2 File Documentation</b>	<b>3</b>
2.1 Almog_String_Manipulation.h File Reference . . . . .	3
2.1.1 Detailed Description . . . . .	4
2.1.2 Macro Definition Documentation . . . . .	5
2.1.2.1 asm_dprintCHAR . . . . .	5
2.1.2.2 asm_dprintINT . . . . .	5
2.1.2.3 asm_dprintSIZE_T . . . . .	6
2.1.2.4 asm_dprintSTRING . . . . .	6
2.1.2.5 ASM_MAX_LEN_LINE . . . . .	6
2.1.2.6 ASM_MAXDIR . . . . .	7
2.1.3 Function Documentation . . . . .	7
2.1.3.1 asm_copy_array_by_indesies() . . . . .	7
2.1.3.2 asm_get_line() . . . . .	8
2.1.3.3 asm_get_next_word_from_line() . . . . .	8
2.1.3.4 asm_get_word_and_cut() . . . . .	9
2.1.3.5 asm_length() . . . . .	10
2.1.3.6 asm_str_in_str() . . . . .	10
2.1.3.7 asm_strncmp() . . . . .	11
2.2 Almog_String_Manipulation.h . . . . .	11
2.3 striped_Almog_String_Manipulation.h File Reference . . . . .	13
2.3.1 Macro Definition Documentation . . . . .	14
2.3.1.1 asm_dprintCHAR . . . . .	14
2.3.1.2 asm_dprintINT . . . . .	14
2.3.1.3 asm_dprintSIZE_T . . . . .	14
2.3.1.4 asm_dprintSTRING . . . . .	15
2.3.1.5 ASM_MAX_LEN_LINE . . . . .	15
2.3.1.6 ASM_MAXDIR . . . . .	15
2.3.2 Function Documentation . . . . .	15
2.3.2.1 asm_copy_array_by_indesies() . . . . .	15
2.3.2.2 asm_get_line() . . . . .	15
2.3.2.3 asm_get_next_word_from_line() . . . . .	16
2.3.2.4 asm_get_word_and_cut() . . . . .	16
2.3.2.5 asm_length() . . . . .	16
2.3.2.6 asm_str_in_str() . . . . .	16
2.3.2.7 asm_strncmp() . . . . .	17
2.4 striped_Almog_String_Manipulation.h . . . . .	17
2.5 temp.c File Reference . . . . .	18
2.5.1 Macro Definition Documentation . . . . .	19
2.5.1.1 ALMOG_STRING_MANIPULATION_IMPLEMENTATION . . . . .	19

2.5.2 Function Documentation . . . . .	19
2.5.2.1 main() . . . . .	19
2.6 temp.c . . . . .	20
<b>Index</b>	<b>21</b>

# Chapter 1

## File Index

### 1.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Almog_String_Manipulation.h</a>	Lightweight string and line manipulation helpers . . . . .	3
<a href="#">striped_Almog_String_Manipulation.h</a>		13
<a href="#">temp.c</a>		18



## Chapter 2

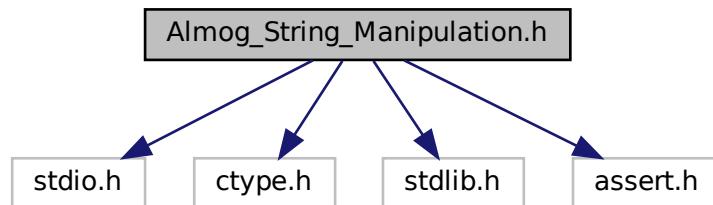
# File Documentation

## 2.1 Almog\_String\_Manipulation.h File Reference

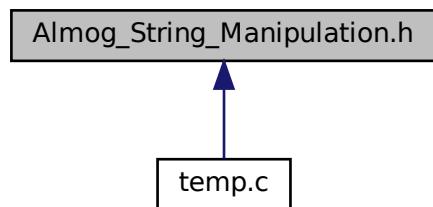
Lightweight string and line manipulation helpers.

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
```

Include dependency graph for Almog\_String\_Manipulation.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define ASM_MAXDIR 100`  
*Generic maximum directory length constant (not used by the functions in this header but available to callers).*
- `#define ASM_MAX_LEN_LINE (int)1e3`  
*Maximum number of characters read by `asm_get_line` (excluding the terminating null).*
- `#define asm_dprintSTRING(expr) printf(#expr " = %s\n", expr)`  
*Debug print a C string expression as "expr = value\n".*
- `#define asm_dprintCHAR(expr) printf(#expr " = %c\n", expr)`  
*Debug print a character expression as "expr = c\n".*
- `#define asm_dprintINT(expr) printf(#expr " = %d\n", expr)`  
*Debug print an integer expression as "expr = n\n".*
- `#define asm_dprintSIZE_T(expr) printf(#expr " = %zu\n", expr)`  
*Debug print a size\_t expression as "expr = n\n".*

## Functions

- `int asm_get_line (FILE *fp, char *dst)`  
*Read a single line from a stream into a buffer.*
- `int asm_length (char *str)`  
*Compute the length of a null-terminated C string.*
- `int asm_get_next_word_from_line (char *dst, char *src, char separator)`  
*Extract the next word from a line without modifying the source.*
- `void asm_copy_array_by_indesies (char *target, int start, int end, char *src)`  
*Copy a substring [start, end] from src into target and null-terminate.*
- `int asm_get_word_and_cut (char *dst, char *src, char separator)`  
*Get the next word and cut the source string at that point.*
- `int asm_str_in_str (char *src, char *word2search)`  
*Count occurrences of a substring within a string.*
- `int asm_strncmp (const char *s1, const char *s2, const int N)`  
*Compare up to N characters for equality (boolean result).*

### 2.1.1 Detailed Description

Lightweight string and line manipulation helpers.

This single-header module provides small utilities for working with C strings:

- Reading a single line from a FILE stream
- Measuring string length
- Extracting the next "word" (token) from a line using a separator
- Cutting the extracted word from the source buffer
- Copying a substring by indices
- Counting occurrences of a substring
- A boolean-style strncmp (returns 1 on equality, 0 otherwise)

## Usage

- In exactly one translation unit, define ALMOG\_STRING\_MANIPULATION\_IMPLEMENTATION before including this header to compile the implementation.
- In all other files, include the header without the macro to get declarations only.

## Notes and limitations

- All destination buffers must be large enough; functions do not grow or allocate buffers.
- `asm_get_line` enforces `MAX_LEN_LINE` characters (not counting the terminating '\0'). Longer lines cause a fatal error via `exit(1)`.
- `asm_strncmp` differs from the standard C `strncmp`: this version returns 1 if equal and 0 otherwise.
- These functions are not locale-aware unless otherwise noted (`isspace` is used for whitespace handling).

Definition in file [Almog\\_String\\_Manipulation.h](#).

## 2.1.2 Macro Definition Documentation

### 2.1.2.1 `asm_dprintCHAR`

```
#define asm_dprintCHAR(  
    expr ) printf(#expr " = %c\n", expr)
```

Debug print a character expression as "expr = c\n".

#### Parameters

<code>expr</code>	An expression that yields a character promoted to int.
-------------------	--

Definition at line 72 of file [Almog\\_String\\_Manipulation.h](#).

### 2.1.2.2 `asm_dprintINT`

```
#define asm_dprintINT(  
    expr ) printf(#expr " = %d\n", expr)
```

Debug print an integer expression as "expr = n\n".

#### Parameters

<code>expr</code>	An expression that yields an int.
-------------------	-----------------------------------

Definition at line 79 of file [Almog\\_String\\_Manipulation.h](#).

### 2.1.2.3 `asm_dprintSIZE_T`

```
#define asm_dprintSIZE_T(  
    expr ) printf(#expr " = %zu\n", expr)
```

Debug print a `size_t` expression as "expr = n\n".

#### Parameters

<code>expr</code>	An expression that yields a <code>size_t</code> .
-------------------	---

Definition at line 86 of file [Almog\\_String\\_Manipulation.h](#).

### 2.1.2.4 `asm_dprintSTRING`

```
#define asm_dprintSTRING(  
    expr ) printf(#expr " = %s\n", expr)
```

Debug print a C string expression as "expr = value\n".

#### Parameters

<code>expr</code>	An expression that yields a pointer to char (const or non-const).
-------------------	---

Definition at line 65 of file [Almog\\_String\\_Manipulation.h](#).

### 2.1.2.5 `ASM_MAX_LEN_LINE`

```
#define ASM_MAX_LEN_LINE (int)1e3
```

Maximum number of characters read by `asm_get_line` (excluding the terminating null).

If an input line exceeds this value before encountering '  
' or EOF, `asm_get_line` prints an error to stderr and terminates the process with `exit(1)`.

Definition at line 58 of file [Almog\\_String\\_Manipulation.h](#).

### 2.1.2.6 ASM\_MAXDIR

```
#define ASM_MAXDIR 100
```

Generic maximum directory length constant (not used by the functions in this header but available to callers).

Definition at line 47 of file [Almog\\_String\\_Manipulation.h](#).

## 2.1.3 Function Documentation

### 2.1.3.1 asm\_copy\_array\_by\_indesies()

```
void asm_copy_array_by_indesies (
    char * target,
    int start,
    int end,
    char * src )
```

Copy a substring [start, end) from src into target and null-terminate.

Copies characters with indices  $i = start, start+1, \dots, end-1$  from src into target, then writes a terminating '\0'.

#### Parameters

<i>target</i>	Destination buffer. Must be large enough to hold $(end - start)$ characters plus the null terminator.
<i>start</i>	Inclusive start index within src (0-based).
<i>end</i>	Exclusive end index within src (must satisfy $end \geq start$ ).
<i>src</i>	Source string buffer.

#### Warning

No bounds checking is performed. The caller must ensure valid indices and sufficient target capacity.

#### Note

This routine supports in-place "left-shift" usage where  $target == src$  and  $start > 0$  (used by `asm_get_word_and_cut`).

Definition at line 232 of file [Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_get\\_word\\_and\\_cut\(\)](#).

### 2.1.3.2 `asm_get_line()`

```
int asm_get_line (
    FILE * fp,
    char * dst )
```

Read a single line from a stream into a buffer.

Reads characters from the FILE stream until a newline ('  
') or EOF is encountered. The newline, if present, is not copied. The result is always null-terminated.

#### Parameters

<code>fp</code>	Input stream (must be non-NULL).
<code>dst</code>	Destination buffer. Must have capacity of at least MAX_LEN_LINE + 1 bytes.

#### Returns

Number of characters stored in dst (excluding the terminating null).

#### Return values

-1	EOF was encountered before any character was read.
----	--

#### Note

If the line exceeds MAX\_LEN\_LINE characters before a newline or EOF, the function prints an error and calls exit(1).

An empty line returns 0 (not -1).

Definition at line 119 of file [Almog\\_String\\_Manipulation.h](#).

References [ASM\\_MAX\\_LEN\\_LINE](#).

### 2.1.3.3 `asm_get_next_word_from_line()`

```
int asm_get_next_word_from_line (
    char * dst,
    char * src,
    char separator )
```

Extract the next word from a line without modifying the source.

Skips leading whitespace in src (as determined by isspace), then copies characters into dst until one of the following is seen: the separator, a newline ('  
''), or the string terminator ('\0'). The copied word in dst is null-terminated and is never empty on success.

Special case:

- If the very first character in src (at index 0, without leading whitespace) is the separator, ''', or '\0', that single character is returned as a one-character "word".

**Parameters**

<i>dst</i>	Destination buffer for the extracted word. Must be large enough to hold the token plus the null terminator.
<i>src</i>	Source C string to parse (not modified by this function).
<i>separator</i>	Separator character to stop at (spelling as in the API).

**Returns**

The number of characters consumed from *src* (i.e., the index of the first unconsumed character).

**Return values**

-1	No word was found (e.g., only whitespace before a delimiter or end-of-string).
----	--

**Note**

The source buffer is not altered. To both extract and advance/cut the source, see `asm_get_word_and_cut()`.

Definition at line 182 of file [Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_get\\_word\\_and\\_cut\(\)](#).

**2.1.3.4 `asm_get_word_and_cut()`**

```
int asm_get_word_and_cut (
    char * dst,
    char * src,
    char separator )
```

Get the next word and cut the source string at that point.

Extracts the next word from *src* (per `asm_get_next_word_from_line` semantics) into *dst*. On success, *src* is modified in-place to remove the consumed prefix. The new *src* begins at the stopping character (the separator, newline, or terminator).

Example: For *src* = "abc,def", *separator* = ','

- *dst* becomes "abc"
- *src* becomes ",def" (note the leading separator remains)

**Parameters**

<i>dst</i>	Destination buffer for the extracted word (large enough for the token and terminating null).
<i>src</i>	Source buffer. Modified in-place if a word is found.
<i>separator</i>	Separator character to stop at (spelling as in the API).

**Returns**

1 if a word was extracted and src adjusted, 0 otherwise.

Definition at line 260 of file [Almog\\_String\\_Manipulation.h](#).

References [asm\\_copy\\_array\\_by\\_indesies\(\)](#), [asm\\_get\\_next\\_word\\_from\\_line\(\)](#), and [asm\\_length\(\)](#).

Referenced by [main\(\)](#).

### 2.1.3.5 **asm\_length()**

```
int asm_length (
    char * str )
```

Compute the length of a null-terminated C string.

**Parameters**

<i>str</i>	Null-terminated string (must be non-NULL).
------------	--

**Returns**

The number of characters before the terminating null byte.

Definition at line 146 of file [Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_get\\_word\\_and\\_cut\(\)](#), and [asm\\_str\\_in\\_str\(\)](#).

### 2.1.3.6 **asm\_str\_in\_str()**

```
int asm_str_in_str (
    char * src,
    char * word2search )
```

Count occurrences of a substring within a string.

Counts how many times word2search appears in src. Occurrences may overlap.

**Parameters**

<i>src</i>	The string to search in (must be null-terminated).
<i>word2search</i>	The substring to find (must be null-terminated).

**Returns**

The number of (possibly overlapping) occurrences found.

Definition at line 285 of file [Almog\\_String\\_Manipulation.h](#).

References [asm\\_length\(\)](#), and [asm\\_strncmp\(\)](#).

**2.1.3.7 asm\_strncmp()**

```
int asm_strncmp (
    const char * s1,
    const char * s2,
    const int N )
```

Compare up to N characters for equality (boolean result).

Returns 1 if the first N characters of s1 and s2 are all equal; otherwise returns 0. Unlike the standard C strncmp, which returns 0 on equality and a non-zero value on inequality/order, this function returns a boolean-like result (1 == equal, 0 == different).

**Parameters**

<i>s1</i>	First string (may be shorter than N).
<i>s2</i>	Second string (may be shorter than N).
<i>N</i>	Number of characters to compare.

**Returns**

1 if equal for the first N characters, 0 otherwise.

Definition at line 310 of file [Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_str\\_in\\_str\(\)](#).

**2.2 Almog\_String\_Manipulation.h**

```
00001
00034 #ifndef ALMOG_STRING_MANIPULATION_H_
00035 #define ALMOG_STRING_MANIPULATION_H_
00036
00037 #include <stdio.h>
00038 #include <ctype.h>
00039 #include <stdlib.h>
00040 #include <assert.h>
00041
00047 #define ASM_MAXDIR 100
00048
00058 #define ASM_MAX_LEN_LINE (int)1e3
00059
00065 #define asm_dprintSTRING(expr) printf(#expr " = %s\n", expr)
00066
00072 #define asm_dprintCHAR(expr) printf(#expr " = %c\n", expr)
00073
00079 #define asm_dprintINT(expr) printf(#expr " = %d\n", expr)
00080
```

```
00086 #define asm_dprintSIZE_T(expr) printf(#expr " = %zu\n", expr)
00087
00088 int asm_get_line(FILE *fp, char *dst);
00089 int asm_length(char *str);
00090 int asm_get_next_word_from_line(char *dst, char *src, char seperator);
00091 void asm_copy_array_by_indexies(char *target, int start, int end, char *src);
00092 int asm_get_word_and_cut(char *dst, char *src, char seperator);
00093 int asm_str_in_str(char *src, char *word2search);
00094 int asm_strncmp(const char *s1, const char *s2, const int N);
00095
00096 #endif /*ALMOG_STRING_MANIPULATION_H*/
00097
00098 #ifdef ALMOG_STRING_MANIPULATION_IMPLEMENTATION
00099 #undef ALMOG_STRING_MANIPULATION_IMPLEMENTATION
00100
00101
00119 int asm_get_line(FILE *fp, char *dst)
00120 {
00121     int i = 0;
00122     char c;
00123
00124     while ((c = fgetc(fp)) != '\n' && c != EOF) {
00125         dst[i] = c;
00126         i++;
00127         if (i >= ASM_MAX_LEN_LINE) {
00128             fprintf(stderr, "ERROR: line too long\n");
00129             exit(1);
00130         }
00131     }
00132     dst[i] = '\0';
00133     if (c == EOF && i == 0) {
00134         return -1;
00135     }
00136     return i;
00137 }
00138
00146 int asm_length(char *str)
00147 {
00148     char c;
00149     int i = 0;
00150
00151     while ((c = str[i]) != '\0') {
00152         i++;
00153     }
00154     return i++;
00155 }
00156
00182 int asm_get_next_word_from_line(char *dst, char *src, char seperator)
00183 {
00184     int i = 0, j = 0;
00185     char c;
00186
00187     while (isspace((c = src[i]))) {
00188         i++;
00189     }
00190
00191     while ((c = src[i]) != seperator &&
00192             c != '\n' &&
00193             c != '\0') {
00194         dst[j] = src[i];
00195         i++;
00196         j++;
00197     }
00198
00199     if ((c == seperator ||
00200          c == '\n' ||
00201          c == '\0') && i == 0) {
00202         dst[j++] = c;
00203         i++;
00204     }
00205
00206     dst[j] = '\0';
00207
00208     if (j == 0) {
00209         return -1;
00210     }
00211     return i;
00212
00213 }
00214
00232 void asm_copy_array_by_indexies(char *target, int start, int end, char *src)
00233 {
00234     int j = 0;
00235     for (int i = start; i < end; i++) {
00236         target[j] = src[i];
00237         j++;
00238     }
}
```

```

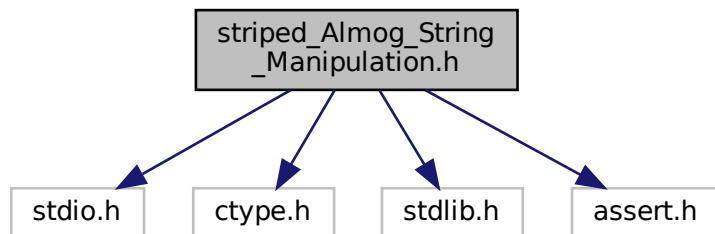
00239     target[j] = '\0';
00240 }
00241
00260 int asm_get_word_and_cut(char *dst, char *src, char separator)
00261 {
00262     int last_pos;
00263
00264     if (src[0] == '\0') {
00265         return 0;
00266     }
00267     last_pos = asm_get_next_word_from_line(dst, src, separator);
00268     if (last_pos == -1) {
00269         return 0;
00270     }
00271     asm_copy_array_by_indeisies(src, last_pos, asm_length(src), src);
00272     return 1;
00273 }
00274
00285 int asm_str_in_str(char *src, char *word2search)
00286 {
00287     int i = 0, num_of_accur = 0;
00288     while (src[i] != '\0') {
00289         if (asm_strncmp(src+i, word2search, asm_length(word2search))) {
00290             num_of_accur++;
00291         }
00292         i++;
00293     }
00294     return num_of_accur;
00295 }
00296
00310 int asm_strncmp(const char *s1, const char *s2, const int N)
00311 {
00312     int i = 0;
00313     while (i < N) {
00314         if (s1[i] == '\0' && s2[i] == '\0') {
00315             break;
00316         }
00317         if (s1[i] != s2[i] || (s1[i] == '\0') || (s2[i] == '\0')) {
00318             return 0;
00319         }
00320         i++;
00321     }
00322     return 1;
00323 }
00324
00325
00326 #endif /*ALMOG_STRING_MANIPULATION_IMPLEMENTATION*/
00327

```

## 2.3 striped\_Almog\_String\_Manipulation.h File Reference

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
```

Include dependency graph for striped\_Almog\_String\_Manipulation.h:



## Macros

- #define ASM\_MAXDIR 100
- #define ASM\_MAX\_LEN\_LINE (int)1e3
- #define asm\_dprintSTRING(expr) printf(#expr " = %s\n", expr)
- #define asm\_dprintCHAR(expr) printf(#expr " = %c\n", expr)
- #define asm\_dprintINT(expr) printf(#expr " = %d\n", expr)
- #define asm\_dprintSIZE\_T(expr) printf(#expr " = %zu\n", expr)

## Functions

- int `asm_get_line` (FILE \*fp, char \*dst)
- int `asm_length` (char \*str)
- int `asm_get_next_word_from_line` (char \*dst, char \*src, char separator)
- void `asm_copy_array_by_indesies` (char \*target, int start, int end, char \*src)
- int `asm_get_word_and_cut` (char \*dst, char \*src, char separator)
- int `asm_str_in_str` (char \*src, char \*word2search)
- int `asm_strncmp` (const char \*s1, const char \*s2, const int N)

### 2.3.1 Macro Definition Documentation

#### 2.3.1.1 `asm_dprintCHAR`

```
#define asm_dprintCHAR(
    expr ) printf(#expr " = %c\n", expr)
```

Definition at line 10 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

#### 2.3.1.2 `asm_dprintINT`

```
#define asm_dprintINT(
    expr ) printf(#expr " = %d\n", expr)
```

Definition at line 11 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

#### 2.3.1.3 `asm_dprintSIZE_T`

```
#define asm_dprintSIZE_T(
    expr ) printf(#expr " = %zu\n", expr)
```

Definition at line 12 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

### 2.3.1.4 asm\_dprintSTRING

```
#define asm_dprintSTRING( expr ) printf(#expr " = %s\n", expr)
```

Definition at line 9 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

### 2.3.1.5 ASM\_MAX\_LEN\_LINE

```
#define ASM_MAX_LEN_LINE (int)1e3
```

Definition at line 8 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

### 2.3.1.6 ASM\_MAXDIR

```
#define ASM_MAXDIR 100
```

Definition at line 7 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

## 2.3.2 Function Documentation

### 2.3.2.1 asm\_copy\_array\_by\_indesies()

```
void asm_copy_array_by_indesies (
    char * target,
    int start,
    int end,
    char * src )
```

Definition at line 76 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_get\\_word\\_and\\_cut\(\)](#).

### 2.3.2.2 asm\_get\_line()

```
int asm_get_line (
    FILE * fp,
    char * dst )
```

Definition at line 23 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

References [ASM\\_MAX\\_LEN\\_LINE](#).

### 2.3.2.3 `asm_get_next_word_from_line()`

```
int asm_get_next_word_from_line (
    char * dst,
    char * src,
    char separator )
```

Definition at line 50 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_get\\_word\\_and\\_cut\(\)](#).

### 2.3.2.4 `asm_get_word_and_cut()`

```
int asm_get_word_and_cut (
    char * dst,
    char * src,
    char separator )
```

Definition at line 85 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

References [asm\\_copy\\_array\\_by\\_indesies\(\)](#), [asm\\_get\\_next\\_word\\_from\\_line\(\)](#), and [asm\\_length\(\)](#).

### 2.3.2.5 `asm_length()`

```
int asm_length (
    char * str )
```

Definition at line 41 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_get\\_word\\_and\\_cut\(\)](#), and [asm\\_str\\_in\\_str\(\)](#).

### 2.3.2.6 `asm_str_in_str()`

```
int asm_str_in_str (
    char * src,
    char * word2search )
```

Definition at line 98 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

References [asm\\_length\(\)](#), and [asm\\_strncmp\(\)](#).

### 2.3.2.7 asm\_strncmp()

```
int asm_strncmp (
    const char * s1,
    const char * s2,
    const int N )
```

Definition at line 109 of file [striped\\_Almog\\_String\\_Manipulation.h](#).

Referenced by [asm\\_str\\_in\\_str\(\)](#).

## 2.4 striped\_Almog\_String\_Manipulation.h

```
00001 #ifndef ALMOG_STRING_MANIPULATION_H_
00002 #define ALMOG_STRING_MANIPULATION_H_
00003 #include <stdio.h>
00004 #include <ctype.h>
00005 #include <stdlib.h>
00006 #include <assert.h>
00007 #define ASM_MAXDIR 100
00008 #define ASM_MAX_LEN_LINE (int)1e3
00009 #define asm_dprintSTRING(expr) printf(#expr " = %s\n", expr)
00010 #define asm_dprintCHAR(expr) printf(#expr " = %c\n", expr)
00011 #define asm_dprintINT(expr) printf(#expr " = %d\n", expr)
00012 #define asm_dprintSIZE_T(expr) printf(#expr " = %zu\n", expr)
00013 int asm_get_line(FILE *fp, char *dst);
00014 int asm_length(char *str);
00015 int asm_get_next_word_from_line(char *dst, char *src, char seperator);
00016 void asm_copy_array_by_indexes(char *target, int start, int end, char *src);
00017 int asm_get_word_and_cut(char *dst, char *src, char seperator);
00018 int asm_str_in_str(char *src, char *word2search);
00019 int asm_strncmp(const char *s1, const char *s2, const int N);
00020 #endif
00021 #ifdef ALMOG_STRING_MANIPULATION_IMPLEMENTATION
00022 #undef ALMOG_STRING_MANIPULATION_IMPLEMENTATION
00023 int asm_get_line(FILE *fp, char *dst)
00024 {
00025     int i = 0;
00026     char c;
00027     while ((c = fgetc(fp)) != '\n' && c != EOF) {
00028         dst[i] = c;
00029         i++;
00030         if (i >= ASM_MAX_LEN_LINE) {
00031             fprintf(stderr, "ERROR: line too long\n");
00032             exit(1);
00033         }
00034     }
00035     dst[i] = '\0';
00036     if (c == EOF && i == 0) {
00037         return -1;
00038     }
00039     return i;
00040 }
00041 int asm_length(char *str)
00042 {
00043     char c;
00044     int i = 0;
00045     while ((c = str[i]) != '\0') {
00046         i++;
00047     }
00048     return i++;
00049 }
00050 int asm_get_next_word_from_line(char *dst, char *src, char seperator)
00051 {
00052     int i = 0, j = 0;
00053     char c;
00054     while (isspace((c = src[i]))) {
00055         i++;
00056     }
00057     while ((c = src[i]) != seperator &&
00058            c != '\n' &&
00059            c != '\0') {
00060         dst[j] = src[i];
00061         i++;
00062         j++;
00063     }
00064     if ((c == seperator ||
00065          c == '\n' ||
```

```

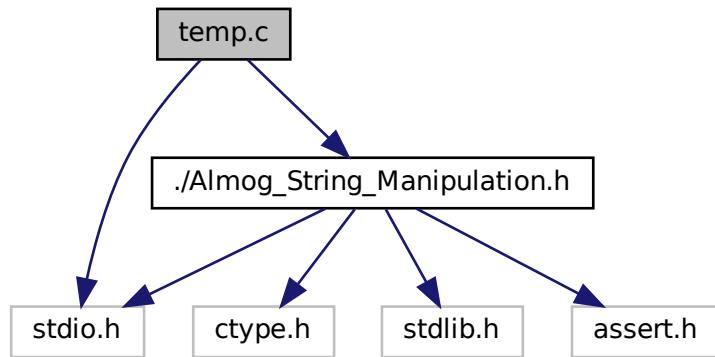
00066         c == '\0') && i == 0) {
00067             dst[j++] = c;
00068             i++;
00069     }
00070     dst[j] = '\0';
00071     if (j == 0) {
00072         return -1;
00073     }
00074     return i;
00075 }
00076 void asm_copy_array_by_indesies(char *target, int start, int end, char *src)
00077 {
00078     int j = 0;
00079     for (int i = start; i < end; i++) {
00080         target[j] = src[i];
00081         j++;
00082     }
00083     target[j] = '\0';
00084 }
00085 int asm_get_word_and_cut(char *dst, char *src, char separator)
00086 {
00087     int last_pos;
00088     if (src[0] == '\0') {
00089         return 0;
00090     }
00091     last_pos = asm_get_next_word_from_line(dst, src, separator);
00092     if (last_pos == -1) {
00093         return 0;
00094     }
00095     asm_copy_array_by_indesies(src, last_pos, asm_length(src), src);
00096     return 1;
00097 }
00098 int asm_str_in_str(char *src, char *word2search)
00099 {
00100     int i = 0, num_of_accur = 0;
00101     while (src[i] != '\0') {
00102         if (asm_strncmp(src+i, word2search, asm_length(word2search))) {
00103             num_of_accur++;
00104         }
00105         i++;
00106     }
00107     return num_of_accur;
00108 }
00109 int asm_strncmp(const char *s1, const char *s2, const int N)
00110 {
00111     int i = 0;
00112     while (i < N) {
00113         if (s1[i] == '\0' && s2[i] == '\0') {
00114             break;
00115         }
00116         if (s1[i] != s2[i] || (s1[i] == '\0') || (s2[i] == '\0')) {
00117             return 0;
00118         }
00119         i++;
00120     }
00121     return 1;
00122 }
00123 #endif

```

## 2.5 temp.c File Reference

```
#include <stdio.h>
#include "./Almog_String_Manipulation.h"
```

Include dependency graph for temp.c:



## Macros

- `#define ALMOG_STRING_MANIPULATION_IMPLEMENTATION`

## Functions

- `int main (void)`

### 2.5.1 Macro Definition Documentation

#### 2.5.1.1 ALMOG\_STRING\_MANIPULATION\_IMPLEMENTATION

```
#define ALMOG_STRING_MANIPULATION_IMPLEMENTATION
```

Definition at line 2 of file [temp.c](#).

### 2.5.2 Function Documentation

#### 2.5.2.1 main()

```
int main (
    void )
```

Definition at line 5 of file [temp.c](#).

References [asm\\_get\\_word\\_and\\_cut\(\)](#).

## 2.6 temp.c

```
00001 #include <stdio.h>
00002 #define ALMOG_STRING_MANIPULATION_IMPLEMENTATION
00003 #include "./Almog_String_Manipulation.h"
00004
00005 int main(void)
00006 {
00007     char str[] = "almog dobreșcu";
00008     char word[256];
00009
00010     asm_get_word_and_cut(word, str, ' ');
00011
00012     printf("str: %s\n", str);
00013
00014     printf("word: %s\n", word);
00015
00016
00017     return 0;
00018 }
```