

Exercism: Acronym in C

Eric Bailey

February 25, 2018 ¹

¹ Last updated February 25, 2018

Contents

<i>The abbreviate function</i>	1
<i>Count the words in a phrase</i>	2
<i>Determining the start of a word</i>	2
<i>Include headers</i>	2
<i>Full Listing</i>	4
<i>Chunks</i>	6
<i>Index</i>	6

1a `<* 1a>≡`
`#include "acronym.h"`
`<Include headers. 2c>`

`<Define the is_word_start function. 2b>`

`<Define the word_count function. 2a>`

`<Define the abbreviate function. 1b>`
Root chunk (not used in this document).

WRITE ME

The abbreviate function

1b `<Define the abbreviate function. 1b>≡`
`char *abbreviate(const char *phrase)`
`{`
 `if (phrase == NULL || phrase[0] == '\0')`
 `return NULL;`

 `char *acronym = NULL;`
 `if (!(acronym = calloc(word_count(phrase) + 1, sizeof phrase[0])))`
 `return NULL;`

 `acronym[0] = toupper(phrase[0]);`

 `for (size_t i = 1, j = 1; phrase[i] != '\0'; ++i) {`
 `if (is_word_start(phrase[i], phrase[i - 1]))`
 `acronym[j++] = toupper(phrase[i]);`
 `}`

 `return acronym;`
`}`

This code is used in chunk 1a.

Defines:

abbreviate, never used.

Uses calloc 3, is_word_start 2b 2b, NULL 3, size_t 3, toupper 2c, and word_count 2a 2a.

Count the words in a phrase

To determine the number of words in a phrase, count the number of characters for which `is_word_start` holds.

2a *<Define the word_count function. 2a>*≡

```
static int word_count(const char phrase[])
{
    if (phrase == NULL)
        return 0;

    int count = 1;

    for (size_t i = 1; phrase[i] != '\0'; ++i) {
        if (is_word_start(phrase[i], phrase[i - 1]))
            count++;
    }

    return count;
}
```

This code is used in chunk 1a.

Defines:

`word_count`, used in chunk 1b.

Uses `is_word_start` 2b 2b, `NULL` 3, and `size_t` 3.

Determining the start of a word

The current character starts a word if, and only if, it is alphabetic and the previous character is not.

2b *<Define the is_word_start function. 2b>*≡

```
static int is_word_start(char current, char previous)
{
    return isalpha(current) && (!isalpha(previous));
}
```

This code is used in chunk 1a.

Defines:

`is_word_start`, used in chunks 1b and 2a.

Uses `isalpha` 2c.

Include headers

2c *<Include headers. 2c>*≡

```
#include <ctype.h>
```

This definition is continued in chunk 3.

This code is used in chunk 1a.

Defines:

`isalpha`, used in chunk 2b.

`toupper`, used in chunk 1b.

```
3  <Include headers. 2c>+≡  
    #include <stdlib.h>
```

This code is used in chunk 1a.

Defines:

 calloc, used in chunk 1b.

 NULL, used in chunks 1b and 2a.

 size_t, used in chunks 1b and 2a.

*Full Listing*Listing 1: `acronym.h`

```
1  #ifndef ACRONYM_H
2  #define ACRONYM_H
3
4  char *abbreviate(const char *phrase);
5
6  #endif
```

Listing 2: `acronym.c`

```

1  #include "acronym.h"
2  #include <ctype.h>
3  #include <stdlib.h>
4
5
6  static int is_word_start(char current, char previous)
7  {
8      return isalpha(current) && (!isalpha(previous));
9  }
10
11
12 static int word_count(const char phrase[])
13 {
14     if (phrase == NULL)
15         return 0;
16
17     int count = 1;
18
19     for (size_t i = 1; phrase[i] != '\0'; ++i) {
20         if (is_word_start(phrase[i], phrase[i - 1]))
21             count++;
22     }
23
24     return count;
25 }
26
27
28 char *abbreviate(const char *phrase)
29 {
30     if (phrase == NULL || phrase[0] == '\0')
31         return NULL;
32
33     char *acronym = NULL;
34     if (!(acronym = calloc(word_count(phrase) + 1, sizeof phrase[0])))
35         return NULL;
36
37     acronym[0] = toupper(phrase[0]);
38
39     for (size_t i = 1, j = 1; phrase[i] != '\0'; ++i) {
40         if (is_word_start(phrase[i], phrase[i - 1]))
41             acronym[j++] = toupper(phrase[i]);
42     }
43
44     return acronym;
45 }

```

Chunks

`< * 1a> 1a`
`< Define the abbreviate function. 1b> 1a, 1b`
`< Define the is_word_start function. 2b> 1a, 2b`
`< Define the word_count function. 2a> 1a, 2a`
`< Include headers. 2c> 1a, 2c, 3`

Index

abbreviate: 1b
calloc: 1b, 3
is_word_start: 1b, 2a, 2b, 2b
isalpha: 2b, 2c
NULL: 1b, 2a, 3
size_t: 1b, 2a, 3
toupper: 1b, 2c
word_count: 1b, 2a, 2a