Exercism: Isogram in C

Eric Bailey

February 22, 2018 [1]

[1] Last updated February 23, 2018

An *isogram* is a word or phrase without a repeating letter.

Contents

The `is_isogram` function

To determine a `phrase` is an isogram, there are two failing conditions we must disprove.

1b ⟨*Define the* `is_isogram` *function.* 1b⟩≡

```
bool is_isogram(const char phrase[])
{
    ⟨If phrase is NULL, return false. 2a⟩

    ⟨If any letter in phrase appears more than once, return false. 2b⟩

    ⟨Otherwise, phrase is an isogram. 3c⟩
}
```

This code is used in chunk 1a.
Defines:
seen, used in chunks 2 and 3.

1a ⟨* 1a⟩≡

```
#include "isogram.h"
⟨Include headers. 3d⟩

⟨Define the is_isogram function. 1b⟩
```

Root chunk (not used in this document).

NULL is not an isogram

The implementation first condition is self-explanatory.

2a ⟨*If* phrase *is NULL, return false.* 2a⟩≡
```
if (phrase == NULL)
    return false;
```
This code is used in chunk 1b.
Uses NULL 3d.

Determining whether a word is an isogram

To determine whether phrase is an isogram, we need to keep track of letters we've already seen. For that, use an unsigned 32-bit integer (uint32_t).

2b ⟨*If any letter in* phrase *appears more than once, return false.* 2b⟩≡
```
uint32_t seen = 0;
```
This definition is continued in chunk 2c.
This code is used in chunk 1b.
Uses seen 1b and uint32_t 3e.

Next, iterate through each letter in phrase until NUL, short-circuiting if we've seen one before.

2c ⟨*If any letter in* phrase *appears more than once, return false.* 2b⟩+≡
```
for (size_t i = 0; phrase[i] != '\0'; ++i) {
    ⟨Ignore nonalphabetic characters. 2d⟩

    ⟨If we've already seen the current letter, return false. 2e⟩

    ⟨Otherwise, mask the bit for the current letter on in seen. 3b⟩
}
```
This code is used in chunk 1b.

Since we don't care about nonalphabetic characters, ignore them.

2d ⟨*Ignore nonalphabetic characters.* 2d⟩≡
```
if (!isalpha(phrase[i]))
    continue;
```
This code is used in chunk 2c.
Uses isalpha 3f.

To determine, case-insensitively, if we've seen a letter already, convert it to uppercase and subtract 'A', e.g 'a' - 'A' == 0 and 'e' - 'A' == 4. Mask the corresponding bit on and store the result.

2e ⟨*If we've already* seen *the current letter, return false.* 2e⟩≡
```
uint32_t bit_mask = 1 « (toupper(phrase[i]) - 'A');
```
This definition is continued in chunk 3a.
This code is used in chunk 2c.
Defines:
  bit_mask, used in chunk 3.
Uses toupper 3f and uint32_t 3e.

If the result of bitwise `seen` `AND` `bit_mask` is nonzero, we've seen this letter before and `phrase` is not an isogram.

3a ⟨*If we've already* `seen` *the current letter, return false.* 2e⟩+≡
```
if (seen & bit_mask)
    return false;
```
This code is used in chunk 2c.
Uses `bit_mask` 2e and `seen` 1b.

If this is a new letter, mask its bit (see `bit_mask`) on in `seen` and proceed to the next letter.

3b ⟨*Otherwise, mask the bit for the current letter on in* `seen`. 3b⟩≡
```
seen |= bit_mask;
```
This code is used in chunk 2c.
Uses `bit_mask` 2e and `seen` 1b.

Double negation

If `phrase` is not not an isogram, then it is an isogram.

3c ⟨*Otherwise,* `phrase` *is an isogram.* 3c⟩≡
```
return true;
```
This code is used in chunk 1b.

✱2·14. $\vdash.\sim(\sim p)\supset p$
(Whitehead and Russell)

Include headers

From `stddef.h` import the `NULL` macro.

3d ⟨*Include headers.* 3d⟩≡
```
#include <stddef.h>
```
This definition is continued in chunk 3.
This code is used in chunk 1a.
Defines:
`NULL`, used in chunk 2a.

From `stdint.h` import the 32-bit unsigned integer type, `uint32_t`.

3e ⟨*Include headers.* 3d⟩+≡
```
#include <stdint.h>
```
This code is used in chunk 1a.
Defines:
`uint32_t`, used in chunk 2.

From `ctype.h` import the functions `isalpha`, to determine whether a character is alphabetic, and `toupper`, to convert a letter to uppercase.

3f ⟨*Include headers.* 3d⟩+≡
```
#include <ctype.h>
```
This code is used in chunk 1a.
Defines:
`isalpha`, used in chunk 2d.
`toupper`, used in chunk 2e.

Full Listing

Listing 1: isogram.h

```h
#ifndef ISOGRAM_H
#define ISOGRAM_H

#include <stdbool.h>

bool is_isogram(const char phrase[]);

#endif
```

Listing 2: isogram.c

```c
#include "isogram.h"
#include <stddef.h>
#include <stdint.h>
#include <ctype.h>


bool is_isogram(const char phrase[])
{
    if (phrase == NULL)
        return false;

    uint32_t seen = 0;

    for (size_t i = 0; phrase[i] != '\0'; ++i) {
        if (!isalpha(phrase[i]))
            continue;

        uint32_t bit_mask = 1 << (toupper(phrase[i]) - 'A');
        if (seen & bit_mask)
            return false;

        seen |= bit_mask;
    }

    return true;
}
```

Chunks

Index

References

Alfred North Whitehead and Bertrand Russell. *Principia mathematica*. Cambridge University Press, 1910-. URL http://name.umdl.umich.edu/AAT3201.0001.001.