ASCII Obfuscation

0.0.1

Generated by Doxygen 1.8.11

Contents

Chapter 1

Todo List

Global calculate_consonant_translation (char consonant, unsigned char *p_out_buffer, unsigned int *p_offset)

look for a workaround for this case

Global calculate_vowel_translation (char vowel, unsigned char *p_out_buffer, unsigned int *p_offset)

look for a workaround for this case

Global main (int argC, char **argV)

Add a check on return code for 'translate_into_obscure' function (and take action depending on it)

Remove following lines after debug

remove 'goto' ASAP => goes against coding style

rework following lines : coding style chosen is not to have 1 exit point per function & forbids 'goto' usage

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 deprecated_functions.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "deprecated_functions.h"
```

Functions

- void shift_letters (void)

  Shift letters from upper case to lower case or from lower case to upper case.
- void character_bit_shifting (void)

  Bit shifts a string of character to make it unreadable.

Variables

- const char input_table [ ] = "I think you know my point about inline if operations : it only obfuscates the code.\n"
- const char bit_shifted_char_table [ ]

3.1.1 Function Documentation

3.1.1.1 void character_bit_shifting (void)

Bit shifts a string of character to make it unreadable.

Definition at line 89 of file deprecated_functions.c.

References input_table.

3.1.1.2 void shift_letters (void)

Shift letters from upper case to lower case or from lower case to upper case.

Definition at line 40 of file deprecated_functions.c.

References SHIFT_VAL.

3.1.2 Variable Documentation

3.1.2.1 const char bit_shifted_char_table[ ]

Initial value:

```
=
{
    0x92, 0x40, 0xe8, 0xd0, 0xd2, 0xdc, 0xd6, 0x40, 0xf2, 0xde, 0xea, 0x40, 0xd6, 0xdc, 0xde, 0xee, 0x40,
      0xda, 0xf2,
    0x40, 0xe0, 0xde, 0xd2, 0xdc, 0xe8, 0x40, 0xc2, 0xc4, 0xde, 0xea, 0xe8, 0x40, 0xd2, 0xdc, 0xd8, 0xd2,
      0xdc, 0xca,
    0x40, 0xd2, 0xcc, 0x40, 0xde, 0xe0, 0xca, 0xe4, 0xc2, 0xe8, 0xd2, 0xde, 0xdc, 0xe6, 0x40, 0x74, 0x40,
      0xd2, 0xe8,
    0x40, 0xde, 0xdc, 0xd8, 0xf2, 0x40, 0xde, 0xc4, 0xcc, 0xea, 0xe6, 0xc6, 0xc2, 0xe8, 0xca, 0xe6, 0x40,
      0xe8, 0xd0,
    0xca, 0x40, 0xc6, 0xde, 0xc8, 0xca, 0x5c
}
```

Definition at line 18 of file deprecated_functions.c.

3.1.2.2 const char input_table[ ] = "I think you know my point about inline if operations : it only obfuscates the code.\n"

Definition at line 17 of file deprecated_functions.c.

Referenced by character_bit_shifting().

3.2 deprecated_functions.h File Reference

Macros

- #define SHIFT_VAL ('a'-'A')

Functions

- void shift_letters (void)

  Shift letters from upper case to lower case or from lower case to upper case.
- void character_bit_shifting (void)

  Bit shifts a string of character to make it unreadable.

3.2.1 Macro Definition Documentation

3.2.1.1 #define SHIFT_VAL ('a'-'A')

Definition at line 6 of file deprecated_functions.h.

Referenced by shift_letters().

3.2.2 Function Documentation

3.2.2.1 void character_bit_shifting (void)

Bit shifts a string of character to make it unreadable.

Definition at line 89 of file deprecated_functions.c.

References input_table.

3.2.2.2 void shift_letters (void)

Shift letters from upper case to lower case or from lower case to upper case.

Definition at line 40 of file deprecated_functions.c.

References SHIFT_VAL.

3.3 main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <iso646.h>
#include <windows.h>
```

Macros

- #define UPPERCASE_LOWERCASE_SHIFT ('a'-'A')
- #define ever (;;)
- #define forever for ever

Functions

- bool is_a_vowel (char letter_to_analyse)

  Tells if the character given in argument is a vowel or not.
- int shift_letter (char character_to_shift, char *p_result_character)

  Shift letters from upper case to lower case or from lower case to upper case.
- int calculate_vowel_translation (char vowel, unsigned char *p_out_buffer, unsigned int *p_offset)

  Calculate the translation of a vowel into consonant.
- int calculate_consonant_translation (char consonant, unsigned char *p_out_buffer, unsigned int *p_offset)

  Calculate the translation of a consonant into another one.
- int translate_into_obscure (char *input, unsigned int input_length, unsigned char *output, unsigned int *p_output_length)

  Translation loop function.
- int main (int argC, char **argV)

  Main program function.

Variables

- const char vowels [ ] = { 'a', 'e', 'i', 'o', 'u', 'y' }
- const char consonants [ ] = { 'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'z' }
- const char message_str [ ] = "i think you know my point about inline if operations : it only obfuscates the code."

3.3.1 Macro Definition Documentation

3.3.1.1 #define ever (;;)

Definition at line 24 of file main.c.

3.3.1.2 #define forever for ever

Definition at line 25 of file main.c.

Referenced by main().

3.3.1.3 #define UPPERCASE_LOWERCASE_SHIFT ('a'-'A')

Definition at line 21 of file main.c.

Referenced by shift_letter().

3.3.2 Function Documentation

3.3.2.1 int calculate_consonant_translation (char *consonant,* unsigned char * *p_out_buffer,* unsigned int * *p_offset*)

Calculate the translation of a consonant into another one.

Parameters

in	*consonant*	Vowel consonant character to translate
out	*p_out_buffer*	Pointer onto output buffer
out	*p_offset*	Pointer onto offset value to set

Returns

An error value :

- 0 if OK
- (-1) if failure

Remarks

Clang-Tidy complains about '#' init value being not used

Todo look for a workaround for this case

Note

`unsigned_consonant` is used instead of `consonant` to avoid Clang-Tidy warnings about wrong implicit casts or value size, during value manipulation and bitwise operations

Determine value of upper_half

Note

operation could be written as follow

```
upper_half = (unsigned char)( ( consonant >> 4U ) & 0x0FU );
```

but it triggers a Clang-Tidy warning, which we want to avoid at the moment
an idea that came while writting the code : if resulting char is a vowel, why not shifting it in addition ?

Definition at line 153 of file main.c.

Referenced by translate_into_obscure().

3.3.2.2 int calculate_vowel_translation (char *vowel*, unsigned char * *p_out_buffer*, unsigned int * *p_offset*)

Calculate the translation of a vowel into consonant.

Parameters

in	*vowel*	Vowel character to translate into consonant character
out	*p_out_buffer*	Pointer onto output buffer
out	*p_offset*	Pointer onto offset value to set

Returns

An error value :

- 0 if OK
- (-1) if failure

Note

The translation of a vowel will not always give 2 consonants but is likely to output one consonant and one vowel from time to time. It has been observed that the translation from 'i' is 'FI' i.e. second character is a vowel. It might be of interest to translate vowels only into consonants i.e. 'i' giving 'FJ' for example (and maybe have consonants only give consonants)

Remarks

Clang-Tidy complains about '#' init value being not used

Todo look for a workaround for this case

Note

`unsigned_vowel` is used instead of `vowel` to avoid Clang-Tidy warnings about wrong implicit casts or value size, during value manipulation and bitwise operations

Determine value of upper_half

Note

operation could be written as follow

```
upper_half = (unsigned char)( ( vowel >> 4U ) & 0x0FU );
```

but it triggers a Clang-Tidy warning, which we want to avoid at the moment

Definition at line 95 of file main.c.

Referenced by translate_into_obscure().

3.3.2.3 bool is_a_vowel (char *letter_to_analyse*)

Tells if the character given in argument is a vowel or not.

Parameters

in	*letter_to_analyse*	Character that we will tell if its a vowel or not

Returns

A boolean value :

- true : letter_to_analyse is a vowel
- false : letter_to_analyse is a consonant

Since 'vowels' table only contains lowercase letters, and we need to handle uppercase we will shift the 'letter_to_↩ analyse' value and use it in the analyse itself afterward

Error case handling :
'shift_letter' fail means that 'letter_to_analyse' is not within the alphabet (see 'shift_letter' return codes)

Parse the vowel table within a loop

Check that letter_to_analyse is value is the same as one in 'vowels[]' table

Comparison success => vowel found => exit with true

Definition at line 211 of file main.c.

References shift_letter(), and vowels.

Referenced by translate_into_obscure().

3.3.2.4 int main (int *argC,* char ** *argV*)

Main program function.

Parameters

in	*argC*	Argument number
in	*argV*	Pointer onto input argument strings

Returns

0 in case of successful execution

Conditional behavior of program:

- if there are arguments in the program call, just translates the input string and exit (or print error message)
- if there is no argument, go to the infinite loop to use the program until asked to quit

Remarks

show how many arguments are passed to the program (following line is only here for debug)
- Just print the input string for now (translation part of the program is not finished yet)

- Perhaps keep a print of which option has been chosen in the end ('encode' or 'decode' for example)

Todo Add a check on return code for 'translate_into_obscure' function (and take action depending on it)

Todo Remove following lines after debug

Remarks

Lines are left as commented code to allow multiple use cases, depending on IDE used for development

Todo remove 'goto' ASAP => goes against coding style

Todo rework following lines : coding style chosen is not to have 1 exit point per function & forbids 'goto' usage

Definition at line 308 of file main.c.

References forever, and translate_into_obscure().

3.3.2.5 int shift_letter (char *character_to_shift*, char * *p_result_character*)

Shift letters from upper case to lower case or from lower case to upper case.

Parameters

in	*character_to_shift*	Character to shift
out	*p_result_character*	Pointer onto resulting character

Returns

An int error code value :

- 0 if OK
- (-1) if failure

Definition at line 54 of file main.c.

References UPPERCASE_LOWERCASE_SHIFT.

Referenced by is_a_vowel().

3.3.2.6 int translate_into_obscure (char * *input*, unsigned int *input_length*, unsigned char * *output*, unsigned int * *p_output_length*)

Translation loop function.

Parameters

in	*input*	String that contains the original message
in	*input_length*	Length of input string
out	*output*	Buffer in which we would write the resulting string
out	*p_output_length*	Pointer to length of output string

Returns

An int value :

- 0 if everything is OK
- (-1) if process ends in error

If 'output' buffer is 'NULL' we might get in trouble trying to set values to a random memory location
=> We would better exit the function as soon as possible with an error code to inform the caller

If there is no input buffer passed in argument, use default string 'message_str'

Definition at line 259 of file main.c.

References calculate_consonant_translation(), calculate_vowel_translation(), is_a_vowel(), and message_str.

Referenced by main().

3.3.3 Variable Documentation

3.3.3.1 const char consonants[ ] = { 'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'z' }

Definition at line 29 of file main.c.

3.3.3.2 const char message_str[ ] = "i think you know my point about inline if operations : it only obfuscates the code."

Definition at line 31 of file main.c.

Referenced by translate_into_obscure().

3.3.3.3 const char vowels[ ] = { 'a', 'e', 'i', 'o', 'u', 'y' }

Definition at line 28 of file main.c.

Referenced by is_a_vowel().

Index