CE103 Algorithms and Programming I HW2

211401039

Generated by Doxygen 1.9.5

Chapter 1

CE103 HW-2 template without function body

Please check homework guide to complete this task

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 CE103HW2LibTest Namespace Reference

Functions

- TEST_CLASS (CE103HW2LibTest)

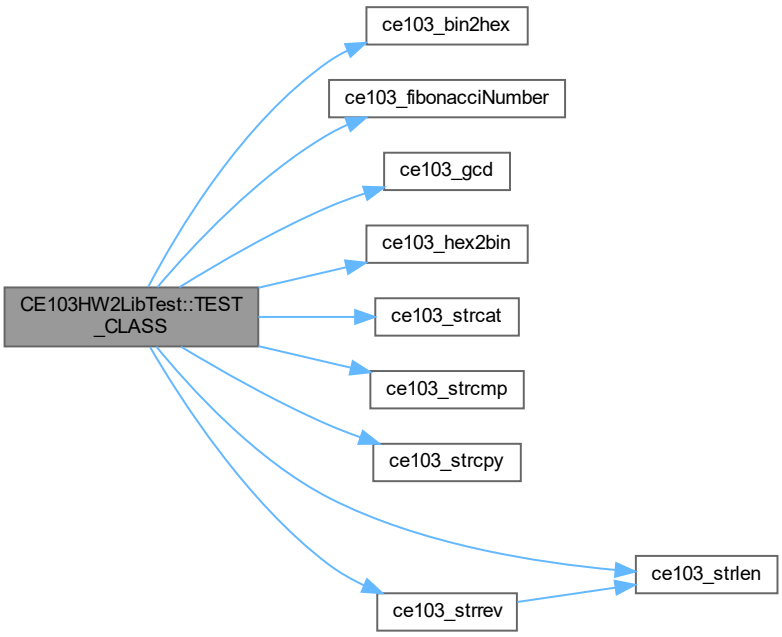
4.1.1 Function Documentation

4.1.1.1 TEST_CLASS()

```
CE103HW2LibTest::TEST_CLASS (
            CE103HW2LibTest  )
```

References ce103_bin2hex(), ce103_fibonacciNumber(), ce103_gcd(), ce103_hex2bin(), ce103_strcat(), ce103_strcmp(), ce103_strcpy(), ce103_strlen(), and ce103_strrev().

Here is the call graph for this function:



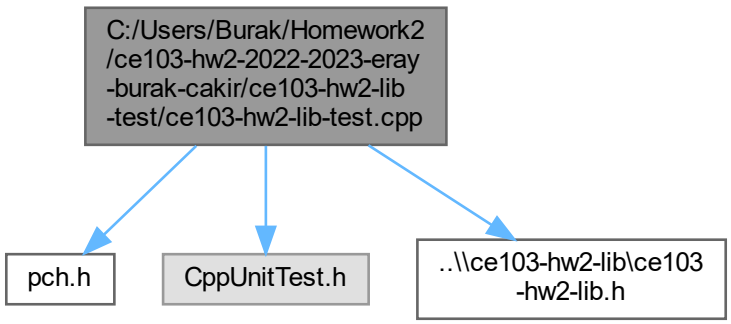
Chapter 5

File Documentation

5.1 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/ce103-hw2-lib-test/ce103-hw2-lib-test.cpp File Reference

```
#include "pch.h"
#include "CppUnitTest.h"
#include "..\\ce103-hw2-lib\ce103-hw2-lib.h"
```

Include dependency graph for ce103-hw2-lib-test.cpp:



Namespaces

- namespace CE103HW2LibTest

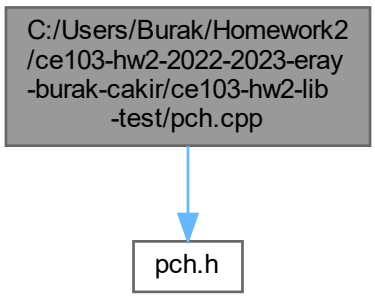
Functions

- CE103HW2LibTest::TEST_CLASS (CE103HW2LibTest)

5.2 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/ce103-hw2-lib-test/pch.cpp File Reference

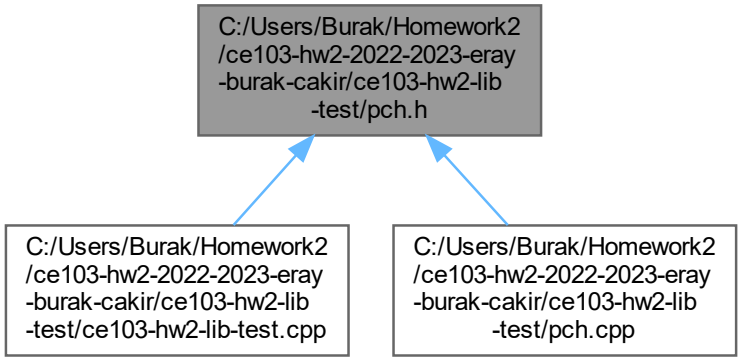
```
#include "pch.h"
```

Include dependency graph for pch.cpp:



5.3 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/ce103-hw2-lib-test/pch.h File Reference

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

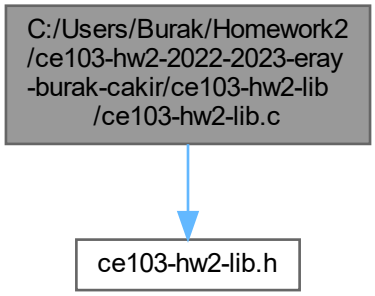


5.4 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/ce103-hw2-lib/ce103-hw2-lib.c File Reference

HW-2 Functions

```
#include "ce103-hw2-lib.h"
```

Include dependency graph for ce103-hw2-lib.c:



Functions

TestFunction(fnCE103HW2Lib)

Auto *Generated Test Function*

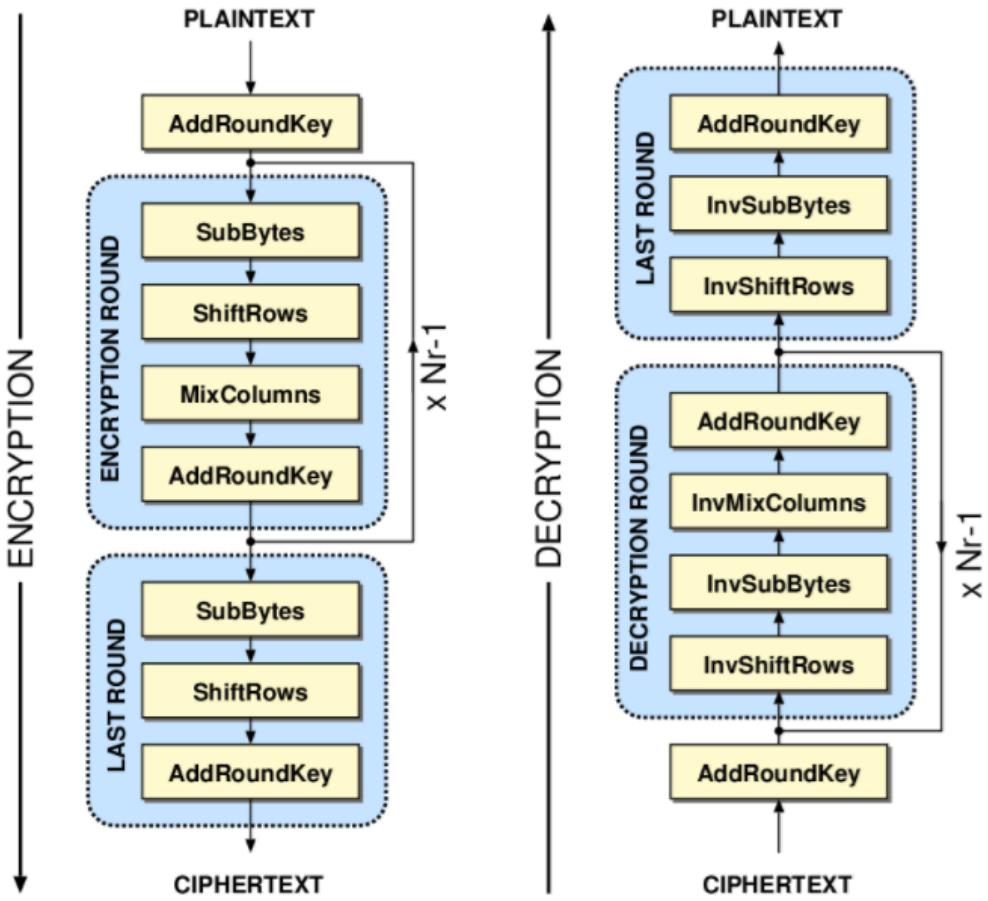
Auto Generated Test Function Has Doxygen and Plantuml Integration

Sample Web Page Link

See also

https://www.cplusplus.com/reference/cstring/strcpy/

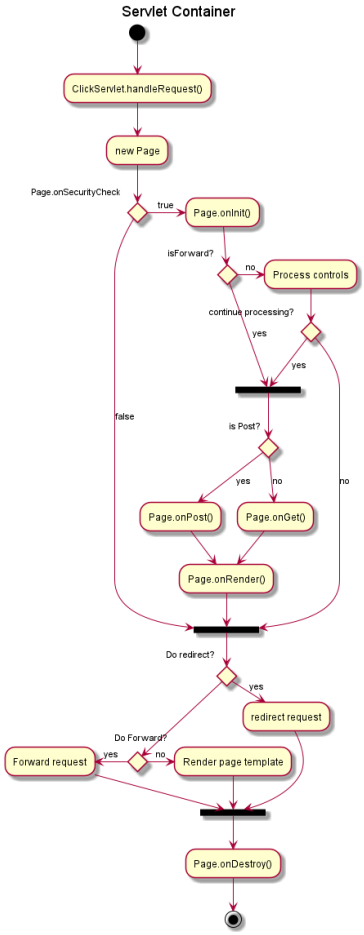
Sample Image AES Block Decryption Operation



Sample Related Function Link

See also

TestFunction (fnCE103HW2Lib) ***Plant UML Sample***



"ClickServlet.handleRequest()" --> "new Page"

if "Page.onSecurityCheck" then ->[true] "Page.onInit()"

if "isForward?" then ->[no] "Process controls"

if "continue processing?" then -->[yes] ===RENDERING=== else -->[no] ===REDIRECT_CHECK=== endif

else -->[yes] ===RENDERING=== endif

if "is Post?" then -->[yes] "Page.onPost()" --> "Page.onRender()" as render --> ===REDIRECT_CHECK=== else -->[no] "Page.onGet()" --> render endif

else -->[false] ===REDIRECT_CHECK=== endif

if "Do redirect?" then ->[yes] "redirect request" --> ==BEFORE_DESTROY=== else if "Do Forward?" then -left->[yes] "Forward request" --> ==BEFORE_DESTROY=== else -right->[no] "Render page template" --> ==BEFORE_DESTROY=== endif endif

--> "Page.onDestroy()" -->(*)

-->

Parameters

in	fia	*[**unsigned** char*] Binary Data Input*
in	fib	*[**int**] Binary Data Input Length*
out	fic	*[**char***] Hex String Output Array*

Return values

[	

b int] operation result success 0 fail 1

Example Usage :

```
#include <stdio.h>
#include <string.h>
int main (){
  unsigned char data[]="\x13\x13\x13x\13";
 int dataLength = 4;
  char dataHex[40];
  if(fnCE103HW2Lib(data,dataLength,dataHex)!=0)
 {
   printf("Operation Failed!");
   return -1;
 }
printf("Converted Data [%s]",dataHex);
  return 0;
}
```

- int fnCE103HW2Lib (unsigned char ∗fia, int fib, char ∗fic)

fibonacciNumber (ce103_fibonacciNumber)

Fibonacci Number Calculator

Calculates the fibonacci number in the given index and return as output

Parameters

in	fiIndex	***[int]** index of fibonacci number in the serie*

Return values

[	

b int] calculated fibonacci number

- int ce103_fibonacciNumber (int fiIndex)

strrev (ce103_strrev)

***Reverse** String*

Reverse given string

Parameters

in	fiStr	***[char∗]** The given string which is needed to be reversed.*

Return values

[	

b char∗] This function returns the string after reversing the given string

- char ∗ ce103_strrev (char ∗fiStr)

strlen (ce103_strlen)

***Get** string length*

Returns the length of the C string str.

The length of a C string is determined by the terminating null-character: A C string is as long as the number of characters between the beginning of the string and the terminating null character (without including the terminating null character itself).

see more `strlen reference 1` *see more* `strlen reference 2` *see more* `strlen reference 3`

Parameters

in	fiStr	*[**const** char∗] pointer to the null-terminated byte string to be examined*

Return values

[	

b int] The length of the null-terminated byte string str.

- int ce103_strlen (const char ∗fiStr)

strcat (ce103_strcat)

***Concatenate** strings*

Appends a copy of the null-terminated byte string pointed to by src to the end of the null-terminated byte string pointed to by dest

The character src[0] replaces the null terminator at the end of dest. The resulting byte string is null-terminated.

The behavior is undefined if the destination array is not large enough for the contents of both src and dest and the terminating null character. The behavior is undefined if the strings overlap. The behavior is undefined if either dest or src is not a pointer to a null-terminated byte string.

see more `strcat reference` *see more* `strcat reference`

Parameters

in	fiDest	*[**char**∗] pointer to the null-terminated byte string to append to*
in	fiSrc	*[**char**∗] pointer to the null-terminated byte string to copy from*

Return values

[	

b char∗] returns a copy of dest

- char ∗ ce103_strcat (char ∗fiDest, char ∗fiSrc)

strcmp (ce103_strcmp)

***Compare** two strings*

Compares two null-terminated byte strings lexicographically.

The sign of the result is the sign of the difference between the values of the first pair of characters (both interpreted as unsigned char) that differ in the strings being compared.The behavior is undefined if lhs or rhs are not pointers to null-terminated byte strings.

see more `strcmp reference` *see more* `strcmp reference`

Parameters

in	fiLhs	*[**const** char∗] pointers to the null-terminated byte strings to compare*
in	fiRhs	*[**const** char∗] pointers to the null-terminated byte strings to compare*

Return values

[	

b int] Negative value if lhs appears before rhs in lexicographical order. Zero if lhs and rhs compare equal. Positive value if lhs appears after rhs in lexicographical order.

- int ce103_strcmp (const char ∗fiLhs, const char ∗fiRhs)

strcpy (ce103_strcpy)

***Copy** string*

Copies the C string pointed by source into the array pointed by destination, including the terminating null character (and stopping at that point).

To avoid overflows, the size of the array pointed by destination shall be long enough to contain the same C string as source (including the terminating null character), and should not overlap in memory with source.

see more `strcpy reference 1` *see more* `strcpy reference 2`

Parameters

out	foDestination	*[**char**∗] Pointer to the destination array where the content is to be copied.*
in	fiSource	*[**const** char∗] C string to be copied.*

Return values

returns	*a copy of dest*

- char ∗ ce103_strcpy (char ∗foDestination, const char ∗fiSource)

hex2bin (ce103_hex2bin)

***Hexadecimal** to Binary (BCD) Conversion*

Hexadecimal to Binary (BCD) Conversion Packs hexadecimal string to packed binary array, Example: "AB1234" => 0xAB 0x12 0x34 If length of the input string is less than the fiHexLen,remaining bytes is not filled. If odd number of characters processed, last nibble is padded with 0

Parameters

in	fiHex	*[**unsigned** char∗] Ascii hex string.*
in	fiHexLen	*[**int**] Ascii data length.*
out	foBin	*[**char**∗] Convertion result as binary.*

- void ce103_hex2bin (char ∗fiHex, int fiHexLen, unsigned char ∗foBin)

bin2hex (ce103_bin2hex)

Binary *(BCD) to Hexadecimal Conversion*

Unpacks alpha numeric value, Example: 0x12 0x34 = "1234".

Parameters

in	fiBin	***[unsigned** char∗] Binary data to be converted.*
in	fiBinLen	***[int]** Binary data length.*
out	foHex	***[char∗]** Convertion result as ascii. Doubles the binary length.*

- void ce103_bin2hex (unsigned char ∗fiBin, int fiBinLen, char ∗foHex)

gcd (ce103_gcd)

Greatest *Common Divisor*

Calculates the greatest common divisor of two number in iterative way for example GCD of 98 and 56 is 14

Parameters

in	fiNum1	***[int]** First Number*
in	fiNum2	***[int]** Second Number*

Return values

[	

b int∗] GCD of numbers.

- int ce103_gcd (int fiNum1, int fiNum2)

5.4.1 Detailed Description

HW-2 Functions

Author

Ugur CORUH

Date

28 November 2022

HW-2 Sample Lib Functions

See also

http://bilgisayar.mmf.erdogan.edu.tr/en/

5.4.2 Function Documentation

5.4.2.1 ce103_bin2hex()

```
void ce103_bin2hex (
            unsigned char * fiBin,
            int fiBinLen,
            char * foHex )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

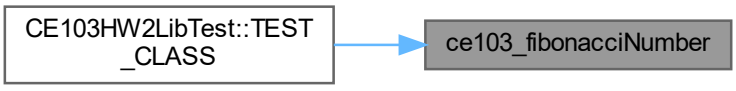
Here is the caller graph for this function:

5.4.2.2 ce103_fibonacciNumber()

```
int ce103_fibonacciNumber (
            int fiIndex )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

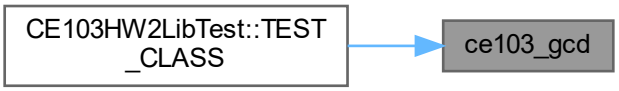


5.4.2.3 ce103_gcd()

```
int ce103_gcd (
            int fiNum1,
            int fiNum2 )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:



5.4.2.4 ce103_hex2bin()

```
void ce103_hex2bin (
            char * fiHex,
            int fiHexLen,
            unsigned char * foBin )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.4.2.5 ce103_strcat()

```
char * ce103_strcat (
            char * fiDest,
            char * fiSrc )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.4.2.6 ce103_strcmp()

```
int ce103_strcmp (
            const char * fiLhs,
            const char * fiRhs )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.4.2.7 ce103_strcpy()

```
char * ce103_strcpy (
            char * foDestination,
            const char * fiSource )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.4.2.8 ce103_strlen()

```
int ce103_strlen (
            const char * fiStr )
```

Referenced by ce103_strrev(), and CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

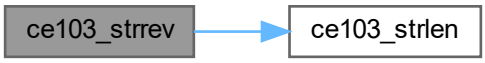
5.4.2.9 ce103_strrev()

```
char * ce103_strrev (
            char * fiStr )
```

References ce103_strlen().

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the call graph for this function:



Here is the caller graph for this function:

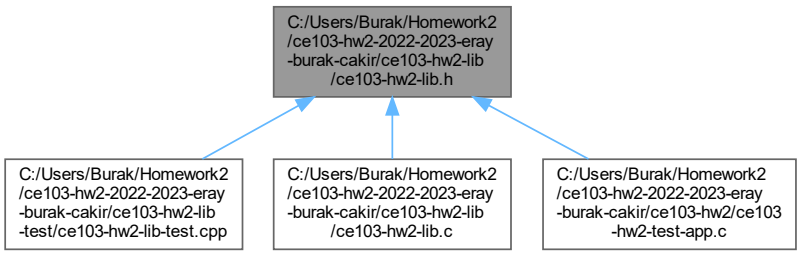
5.4.2.10 fnCE103HW2Lib()

```
int fnCE103HW2Lib (
            unsigned char * fia,
            int fib,
            char * fic )
```

5.5 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/ce103-hw2-lib/ce103-hw2-lib.h File Reference

HW-2 Functions

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



Macros

- #define WIN32_LEAN_AND_MEAN

Functions

TestFunction(fnCE103HW2Lib)

Auto *Generated Test Function*

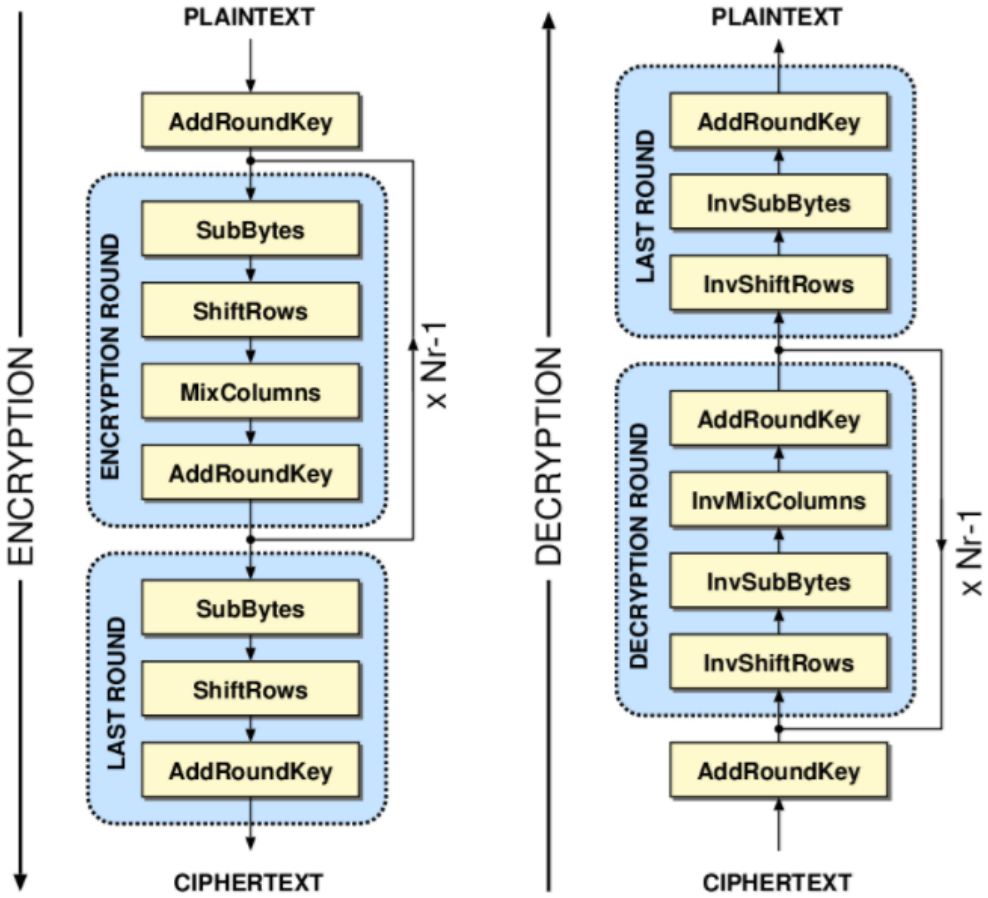
Auto Generated Test Function Has Doxygen and Plantuml Integration

Sample Web Page Link

See also

https://www.cplusplus.com/reference/cstring/strcpy/

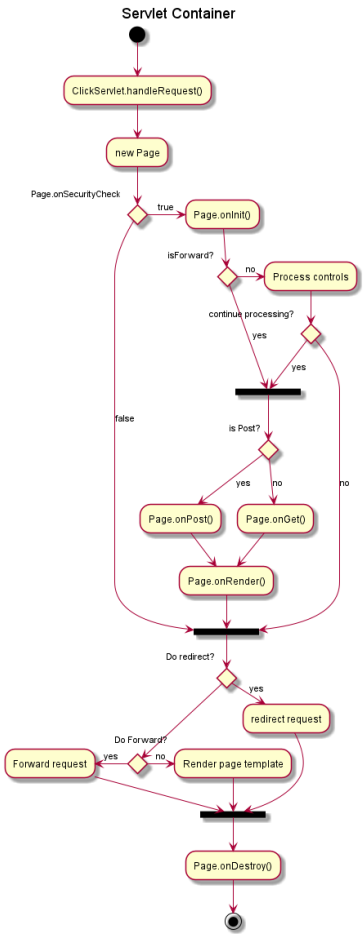
Sample Image AES Block Decryption Operation



Sample Related Function Link

See also

TestFunction (fnCE103HW2Lib) ***Plant UML Sample***



Parameters

in	fia	*[**unsigned** char∗] Binary Data Input*
in	fib	*[**int**] Binary Data Input Length*
out	fic	*[**char**∗] Hex String Output Array*

Return values

[	

b int] operation result success 0 fail 1

Example Usage :

```
#include <stdio.h>
#include <string.h>
 int main (){
unsigned char data[]="\x13\x13\x13x\13";
  int dataLength = 4;
char dataHex[40];
if(fnCE103HW2Lib(data,dataLength,dataHex)!=0)
  {
 printf("Operation Failed!");
 return -1;
  }
 printf("Converted Data [%s]",dataHex);
return 0;
}
```

- int fnCE103HW2Lib (unsigned char ∗fia, int fib, char ∗fic)

fibonacciNumber (ce103_fibonacciNumber)

Fibonacci Number Calculator

Calculates the fibonacci number in the given index and return as output

Parameters

in	fiIndex	*[**int**] index of fibonacci number in the serie*

Return values

[	

b int] calculated fibonacci number

- int ce103_fibonacciNumber (int fiIndex)

strrev (ce103_strrev)

***Reverse** String*

Reverse given string

Parameters

in	fiStr	*[**char**∗] The given string which is needed to be reversed.*

Return values

[	

b char∗] This function returns the string after reversing the given string

- char ∗ ce103_strrev (char ∗fiStr)

strlen (ce103_strlen)

***Get** string length*

Returns the length of the C string str.

The length of a C string is determined by the terminating null-character: A C string is as long as the number of characters between the beginning of the string and the terminating null character (without including the terminating null character itself).

see more `strlen reference 1` *see more* `strlen reference 2` *see more* `strlen reference 3`

Parameters

in	fiStr	*[**const** char∗] pointer to the null-terminated byte string to be examined*

Return values

Return values

[	

b int] The length of the null-terminated byte string str.

- int ce103_strlen (const char ∗fiStr)

strcat (ce103_strcat)

Concatenate *strings*

Appends a copy of the null-terminated byte string pointed to by src to the end of the null-terminated byte string pointed to by dest

The character src[0] replaces the null terminator at the end of dest. The resulting byte string is null-terminated.

The behavior is undefined if the destination array is not large enough for the contents of both src and dest and the terminating null character. The behavior is undefined if the strings overlap. The behavior is undefined if either dest or src is not a pointer to a null-terminated byte string.

see more `strcat reference` *see more* `strcat reference`

Parameters

in	fiDest	*[**char**∗] pointer to the null-terminated byte string to append to*
in	fiSrc	*[**char**∗] pointer to the null-terminated byte string to copy from*

Return values

[	

b char∗] returns a copy of dest

- char ∗ ce103_strcat (char ∗fiDest, char ∗fiSrc)

strcmp (ce103_strcmp)

Compare *two strings*

Compares two null-terminated byte strings lexicographically.

The sign of the result is the sign of the difference between the values of the first pair of characters (both interpreted as unsigned char) that differ in the strings being compared.The behavior is undefined if lhs or rhs are not pointers to null-terminated byte strings.

see more `strcmp reference` *see more* `strcmp reference`

Parameters

in	fiLhs	*[**const** char∗] pointers to the null-terminated byte strings to compare*
in	fiRhs	*[**const** char∗] pointers to the null-terminated byte strings to compare*

Return values

[	

b int] Negative value if lhs appears before rhs in lexicographical order. Zero if lhs and rhs compare equal. Positive value if lhs appears after rhs in lexicographical order.

- int ce103_strcmp (const char *fiLhs, const char *fiRhs)

strcpy (ce103_strcpy)

***Copy** string*

Copies the C string pointed by source into the array pointed by destination, including the terminating null character (and stopping at that point).

To avoid overflows, the size of the array pointed by destination shall be long enough to contain the same C string as source (including the terminating null character), and should not overlap in memory with source.

see more `strcpy reference 1` *see more* `strcpy reference 2`

Parameters

out	foDestination	*[**char***] Pointer to the destination array where the content is to be copied.*
in	fiSource	*[**const** char*] C string to be copied.*

Return values

returns	*a copy of dest*

- char * ce103_strcpy (char *foDestination, const char *fiSource)

hex2bin (ce103_hex2bin)

***Hexadecimal** to Binary (BCD) Conversion*

Hexadecimal to Binary (BCD) Conversion Packs hexadecimal string to packed binary array, Example: "AB1234" => 0xAB 0x12 0x34 If length of the input string is less than the fiHexLen,remaining bytes is not filled. If odd number of characters processed, last nibble is padded with 0

Parameters

in	fiHex	*[**unsigned** char*] Ascii hex string.*
in	fiHexLen	*[**int**] Ascii data length.*
out	foBin	*[**char***] Convertion result as binary.*

- void ce103_hex2bin (char *fiHex, int fiHexLen, unsigned char *foBin)

bin2hex (ce103_bin2hex)

***Binary** (BCD) to Hexadecimal Conversion*

Unpacks alpha numeric value, Example: 0x12 0x34 = "1234".

Parameters

in	fiBin	*[**unsigned** char*] Binary data to be converted.*
in	fiBinLen	*[**int**] Binary data length.*
out	foHex	*[**char***] Convertion result as ascii. Doubles the binary length.*

- void ce103_bin2hex (unsigned char *fiBin, int fiBinLen, char *foHex)

gcd (ce103_gcd)

***Greatest** Common Divisor*

Calculates the greatest common divisor of two number in iterative way for example GCD of 98 and 56 is 14

Parameters

in	fiNum1	***[int]** First Number*
in	fiNum2	***[int]** Second Number*

Return values

[	

b int] GCD of numbers.*

- int ce103_gcd (int fiNum1, int fiNum2)

5.5.1 Detailed Description

HW-2 Functions

Author

Ugur CORUH

Date

28 November 2022

HW-2 Sample Lib Functions Header

See also

http://bilgisayar.mmf.erdogan.edu.tr/en/

5.5.2 Macro Definition Documentation

5.5.2.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

5.5.3 Function Documentation

5.5.3.1 ce103_bin2hex()

```
void ce103_bin2hex (
            unsigned char * fiBin,
            int fiBinLen,
            char * foHex )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

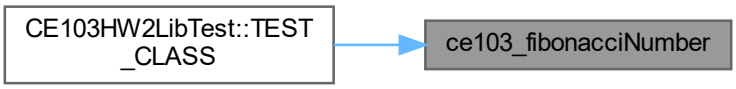
Here is the caller graph for this function:

5.5.3.2 ce103_fibonacciNumber()

```
int ce103_fibonacciNumber (
            int fiIndex )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

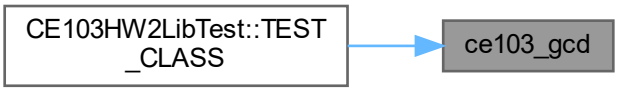


5.5.3.3 ce103_gcd()

```
int ce103_gcd (
            int fiNum1,
            int fiNum2 )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:



5.5.3.4 ce103_hex2bin()

```
void ce103_hex2bin (
            char * fiHex,
            int fiHexLen,
            unsigned char * foBin )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.5.3.5 ce103_strcat()

```
char * ce103_strcat (
            char * fiDest,
            char * fiSrc )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.5.3.6 ce103_strcmp()

```
int ce103_strcmp (
            const char * fiLhs,
            const char * fiRhs )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:

5.5.3.7 ce103_strcpy()

```
char * ce103_strcpy (
            char * foDestination,
            const char * fiSource )
```

Referenced by CE103HW2LibTest::TEST_CLASS().

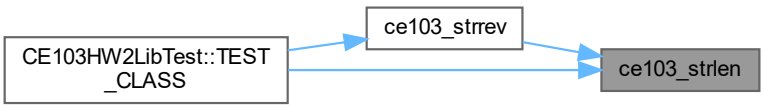
Here is the caller graph for this function:

5.5.3.8 ce103_strlen()

```
int ce103_strlen (
            const char * fiStr )
```

Referenced by ce103_strrev(), and CE103HW2LibTest::TEST_CLASS().

Here is the caller graph for this function:



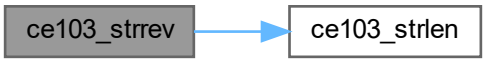
5.5.3.9 ce103_strrev()

```
char * ce103_strrev (
            char * fiStr )
```

References ce103_strlen().

Referenced by CE103HW2LibTest::TEST_CLASS().

Here is the call graph for this function:



Here is the caller graph for this function:

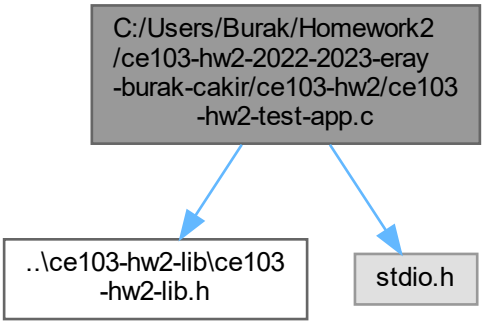
5.5.3.10 fnCE103HW2Lib()

```
int fnCE103HW2Lib (
            unsigned char * fia,
            int fib,
            char * fic )
```

5.6 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/ce103-hw2/ce103-hw2-test-app.c File Reference

```
#include "..\ce103-hw2-lib\ce103-hw2-lib.h"
#include <stdio.h>
```

Include dependency graph for ce103-hw2-test-app.c:



Functions

- int main ()

5.6.1 Function Documentation

5.6.1.1 main()

```
int main ( )
```

5.7 C:/Users/Burak/Homework2/ce103-hw2-2022-2023-eray-burak-cakir/README.md File Reference

Index