My Project

Generated by Doxygen 1.9.2

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Stack Struct Reference

```
#include <stack.h>
```

Public Attributes

- canary_t **canary_left**
- int **size**
- int **capacity**
- data_t * **data**
- hash_t **hash**
- canary_t **canary_right**

3.1.1 Detailed Description

Structure that defines the stack: size, capacity, array pointer, hash, canary left and right for protection.

Parameters

`in`	*canary_left*	canary_left-left canary for protection
`in`	*size*	size-stack size
`in`	*capacity*	capacity-stack capacity
`in`	*data*	data-pointer to the beginning of the array
`in`	*hash*	b-coefficient
`in`	*canary_right*	canary_right-left right for protection

The documentation for this struct was generated from the following file:

- stack.h

Chapter 4

File Documentation

4.1 stack.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <limits.h>
```

Classes

- struct Stack

Macros

- #define FLOAT_DATA
- #define STACK_GENERAL_CHECK(check_function)
- #define STACK_RESIZE_ERROR_CHECK()
- #define STACK_POP_ERROR_CHECK()

Typedefs

- typedef double data_t
- typedef size_t canary_t
- typedef size_t hash_t

Enumerations

- enum StackErrors {
  STK_IS_NULL_PTR = 1 , **DATA_IS_NULL_PTR** = 2 , **STK_DESTROYED** = 4 , **STK_OVERFL** = 8 ,
  STK_UNDERFL = 16 , **STK_DOUBLE_CTED** = 32 , **STRCT_CANARY_BAD** = 64 , **DATA_CANARY_BAD** = 128 ,
  HASH_BAD = 256 , **CAPACITY_LARG_SIZE** = 512 }

Functions

- int StackCtor (Stack ∗stk)
- int StackDtor (Stack ∗stk)
- data_t ∗ StackResize (Stack ∗stk)
- int StackPush (Stack ∗stk, data_t value)
- data_t StackPop (Stack ∗stk)
- int StackErrorCheck (Stack ∗stk)
- int StackCtorCheck (Stack ∗stk)
- int StackDtorCheck (Stack ∗stk)
- void StackDump (Stack ∗stk, const char ∗current_file, const char ∗current_function)
- size_t StackHash (Stack ∗stk)
- size_t Hash (void ∗memory, size_t size_memory)
- int StackDestroy (Stack ∗stk)

4.1.1 Macro Definition Documentation

4.1.1.1 FLOAT_DATA

```
#define FLOAT_DATA
```

Define which specifies what type all elements on the stack are (in this case, the type is double).

4.1.1.2 STACK_GENERAL_CHECK

```
#define STACK_GENERAL_CHECK(
                check_function )
```

Value:

```
do                                                          \
{                                                           \
    Errors = 0;                                             \
                                                            \
    check_function;                                         \
                                                            \
    StackDump(stk, __FILE__, __FUNCTION__);                 \
                                                            \
    if (Errors != 0)                                        \
    {                                                       \
        return 1;                                           \
    }                                                       \
} while (0)
```

Define which describes the general check of the stack.

4.1.1.3 STACK_POP_ERROR_CHECK

```
#define STACK_POP_ERROR_CHECK( )
```

Value:

```
do                                                        \
{                                                         \
    Errors = 0;                                           \
                                                          \
    StackErrorCheck(stk);                                 \
                                                          \
    if (stk->size <= 0)                                   \
    {                                                     \
        Errors |= STK_UNDERFL;                            \
    }                                                     \
                                                          \
    StackDump(stk, __FILE__, __FUNCTION__);               \
                                                          \
    if (Errors != 0)                                      \
    {                                                     \
        return (data_t) 0xBEDABEDA;                         \
    }                                                     \
} while (0)
```

Define which describes the general check of the stack during the poping of the element.

4.1.1.4 STACK_RESIZE_ERROR_CHECK

```
#define STACK_RESIZE_ERROR_CHECK( )
```

Value:

```
do                                                        \
{                                                         \
    Errors = 0;                                           \
                                                          \
    StackErrorCheck(stk);                                 \
                                                          \
    StackDump(stk, __FILE__, __FUNCTION__);               \
                                                          \
    if (Errors != 0)                                      \
    {                                                     \
        return nullptr;                                   \
    }                                                     \
} while (0)
```

Define which describes a general stack check during stack resizing.

4.1.2 Typedef Documentation

4.1.2.1 canary_t

```
typedef size_t canary_t
```

Typedef which specifies that the type of all elements on the stack is int. Typedef which indicates what type of canary is.

4.1.2.2 data_t

```
typedef double data_t
```

Typedef which specifies that the type of all elements on the stack is float.

4.1.2.3 hash_t

```
typedef size_t hash_t
```

Typedef which indicates what type of hash is.

4.1.3 Enumeration Type Documentation

4.1.3.1 StackErrors

```
enum StackErrors
```

Enum which describes all error codes and their numbers.

Parameters

in	*STK_IS_NULL_PTR*	
in	*DATA_IS_NULL_PTR*	
in	*STK_DESTROYED*	
in	*STK_OVERFL*	
in	*STK_UNDERFL*	
in	*STK_DOUBLE_CTED*	
in	*STRCT_CANARY_BAD*	
in	*DATA_CANARY_BAD*	
in	*HASH_BAD*	
in	*CAPACITY_LARG_SIZE*	

4.1.4 Function Documentation

4.1.4.1 Hash()

```
size_t Hash (
             void * memory,
             size_t size_memory )
```

Function that describes the process of hashing a separate piece of memory.

Parameters

in	*memory*	memory-pointer to memory area
in	*size_memory*	size_memory-memory area size
out	*hash*	hash-memory area hash values

Returns

memory area hash values

4.1.4.2 StackCtor()

```
int StackCtor (
            Stack * stk )
```

Function that describes the stack constructor.

Parameters

in	*stk*	stk-stack pointer

Returns

zero

4.1.4.3 StackCtorCheck()

```
int StackCtorCheck (
            Stack * stk )
```

Function that describes the process of checking the stack while the constructor is running.

Parameters

in	*stk*	stk-stack pointer

Returns

zero

4.1.4.4 StackDestroy()

```
int StackDestroy (
            Stack * stk )
```

Function that describes the stack breaking process.

Parameters

in	*stk*	stk-stack pointer

Returns

nothing

4.1.4.5 StackDtor()

```
int StackDtor (
            Stack * stk )
```

Function that describes the stack destructor.

Parameters

in	*stk*	stk-stack pointer

Returns

zero

4.1.4.6 StackDtorCheck()

```
int StackDtorCheck (
            Stack * stk )
```

Function that describes the process of checking the stack while the destructor is running.

Parameters

in	*stk*	stk-stack pointer

Returns

zero

4.1.4.7 StackDump()

```
void StackDump (
            Stack * stk,
            const char * current_file,
            const char * current_function )
```

Function that writes all errors to the Dump.txt file.

Parameters

in	*stk*	stk-stack pointer
in	*current_file-the*	file in which the error occurred
in	*current_function-the*	function in which the error occurred

Returns

nothing

4.1.4.8 StackErrorCheck()

```
int StackErrorCheck (
            Stack * stk )
```

Function that describes the process of checking the stack.

Parameters

in	*stk*	stk-stack pointer
out	*Errors*	Error-number of errors in the stack

Returns

number of errors in the stack

4.1.4.9 StackHash()

```
size_t StackHash (
            Stack * stk )
```

Function that describes the stack hashing process.

Parameters

in	*stk*	stk-stack pointer
out	*hash*	hash-stack hash value

Returns

stack hash value

4.1.4.10 StackPop()

```
data_t StackPop (
            Stack * stk )
```

Function that describes the process of popping the value of an element from the top of the stack.

Parameters

in	*stk*	stk-stack pointer
out	*data_pop*	data_pop-value of the element that is popped from the top of the stack

Returns

value of the element that is popped from the top of the stack

4.1.4.11 StackPush()

```
int StackPush (
            Stack * stk,
            data_t value )
```

Function that describes the process of pushing an element onto the stack.

Parameters

in	*stk*	stk-stack pointer
in	*value*	value-value of the element that is pushed onto the stack

Returns

zero

4.1.4.12 StackResize()

```
data_t * StackResize (
              Stack * stk )
```

Function that describes the process of changing the stack size.

Parameters

in	*stk*	stk-stack pointer
out	*new_adress*	new_adress-new stack pointer

Returns

new stack pointer

4.2 stack.h

Go to the documentation of this file.

```

#pragma once

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <limits.h>

//-----------------------------------------------
//-----------------------------------------------

#define FLOAT_DATA

//-----------------------------------------------
//-----------------------------------------------

#ifdef FLOAT_DATA
typedef double data_t;
#endif

//-----------------------------------------------
//-----------------------------------------------

#ifdef INT_DATA
typedef int data_t;
#endif

//-----------------------------------------------
//-----------------------------------------------

typedef size_t canary_t;

//-----------------------------------------------
//-----------------------------------------------

typedef size_t hash_t;

//-----------------------------------------------
//-----------------------------------------------

struct Stack
{
    canary_t    canary_left;

    int         size;
    int         capacity;
    data_t*   data;
    hash_t    hash;

    canary_t    canary_right;
};
```

```

//------------------------------------------------
//------------------------------------------------

enum StackErrors
{
    STK_IS_NULL_PTR     = 1,
    DATA_IS_NULL_PTR    = 2,
    STK_DESTROYED       = 4,
    STK_OVERFL          = 8,
    STK_UNDERFL         = 16,
    STK_DOUBLE_CTED     = 32,
    STRCT_CANARY_BAD    = 64,
    DATA_CANARY_BAD     = 128,
    HASH_BAD            = 256,
    CAPACITY_LARG_SIZE = 512,
};

//enum ResizeTypes
//{
//  RESIZESMALLER = 0,
//  RESIZELARGER = 1
//};

//------------------------------------------------
//------------------------------------------------

#define STACK_GENERAL_CHECK(check_function)             \
do                                                      \
{                                                       \
    Errors = 0;                                         \
                                                        \
    check_function;                                     \
                                                        \
    StackDump(stk, __FILE__, __FUNCTION__);             \
                                                        \
    if (Errors != 0)                                    \
    {                                                   \
        return 1;                                       \
    }                                                   \
} while (0)

//------------------------------------------------
//------------------------------------------------

#define STACK_RESIZE_ERROR_CHECK()                      \
do                                                      \
{                                                       \
    Errors = 0;                                         \
                                                        \
    StackErrorCheck(stk);                               \
                                                        \
    StackDump(stk, __FILE__, __FUNCTION__);             \
                                                        \
    if (Errors != 0)                                    \
    {                                                   \
        return nullptr;                                 \
    }                                                   \
} while (0)

//------------------------------------------------
//------------------------------------------------

#define STACK_POP_ERROR_CHECK()                         \
do                                                      \
{                                                       \
    Errors = 0;                                         \
                                                        \
    StackErrorCheck(stk);                               \
                                                        \
    if (stk->size <= 0)                                 \
    {                                                   \
        Errors |= STK_UNDERFL;                          \
    }                                                   \
                                                        \
    StackDump(stk, __FILE__, __FUNCTION__);             \
                                                        \
    if (Errors != 0)                                    \
    {                                                   \
        return (data_t) 0xBEDABEDA;                     \
    }                                                   \
} while (0)

//------------------------------------------------
//------------------------------------------------

int StackCtor(Stack* stk);
```

```

//------------------------------------------------
//------------------------------------------------

int StackDtor(Stack* stk);

//------------------------------------------------
//------------------------------------------------

data_t* StackResize(Stack* stk);

//------------------------------------------------
//------------------------------------------------

int StackPush(Stack* stk, data_t value);

//------------------------------------------------
//------------------------------------------------

data_t StackPop(Stack* stk);

//------------------------------------------------
//------------------------------------------------

int StackErrorCheck(Stack* stk);

//------------------------------------------------
//------------------------------------------------

int StackCtorCheck (Stack* stk);

//------------------------------------------------
//------------------------------------------------

int StackDtorCheck (Stack* stk);

//------------------------------------------------
//------------------------------------------------

void StackDump (Stack* stk, const char* current_file, const char* current_function);

//------------------------------------------------
//------------------------------------------------

size_t StackHash (Stack* stk);

//------------------------------------------------
//------------------------------------------------

size_t Hash (void* memory, size_t size_memory);

//------------------------------------------------
//------------------------------------------------

int StackDestroy(Stack* stk);
```

Index