Graphs

Generated by Doxygen 1.8.18

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 Expression Struct Reference

`#include <rec_desc.h>`

Data Fields

- unsigned char * string_form
  string with the expression
- unsigned char * curpointer
  pointer to the current symbol(used internally)

3.1.1 Detailed Description

structure of an infex expression

3.1.2 Field Documentation

3.1.2.1 curpointer

`unsigned char* Expression::curpointer`

pointer to the current symbol(used internally)

3.1.2.2 string_form

```
unsigned char* Expression::string_form
```

string with the expression

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

- rec_desc.h

3.2 RPN Struct Reference

```
#include <RPN.h>
```

Data Fields

- void * data
  input data of rpn
- size_t data_size
  size of data_size
- size_t occupied
  tracking of the last written element

3.2.1 Detailed Description

struct of an RPN

3.2.2 Field Documentation

3.2.2.1 data

```
void* RPN::data
```

input data of rpn

3.2.2.2 data_size

```
size_t RPN::data_size
```

size of data_size

3.2.2.3 occupied

```
size_t RPN::occupied
```

tracking of the last written element

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

- RPN.h

3.3 Stack Struct Reference

structure of a stack

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

Data Fields

- void * data
  data of a stack
- void * stack_top
  pointer to the end of a valid stack
- size_t stack_size
  size of a stack(not always to the last element)
- size_t cur_size
  control of the current sie of stack(used internally)

3.3.1 Detailed Description

structure of a stack

3.3.2 Field Documentation

3.3.2.1 cur_size

```
size_t Stack::cur_size
```

control of the current sie of stack(used internally)

3.3.2.2 data

```
void* Stack::data
```

data of a stack

3.3.2.3 stack_size

```
size_t Stack::stack_size
```

size of a stack(not always to the last element)

3.3.2.4 stack_top

```
void* Stack::stack_top
```

pointer to the end of a valid stack

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

- stack.h

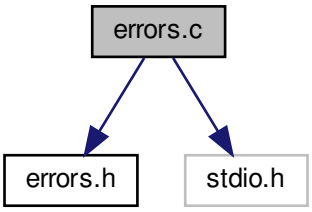
Chapter 4

File Documentation

4.1 errors.c File Reference

```
#include "errors.h"
#include <stdio.h>
```

Include dependency graph for errors.c:



Functions

- void err_print (int err)

4.1.1 Function Documentation

4.1.1.1 err_print()

```
void err_print (
            int err )
```

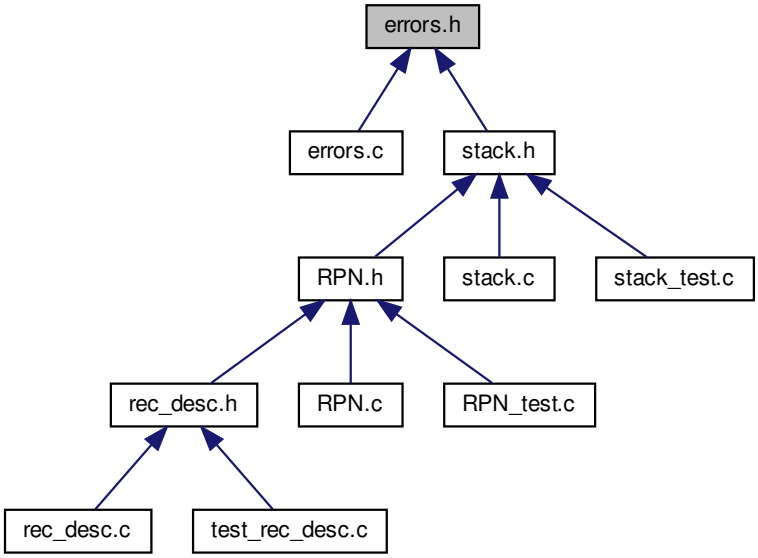
print an error message according to the error code

Parameters

err	error code

4.2 errors.h File Reference

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



Enumerations

- enum {
  E_MEM_ALLOC = 1, E_OVERFLOW = 2, E_UNDERFLOW = 3, E_UNEXPECTED_SYMBOL = 4,
  E_ZERO_DIVISION = 5, E_UNKNOWN_VAR = 6, E_UNBALANCED_BRACKET = 7 }

Functions

- void err_print (int err)

4.2.1 Enumeration Type Documentation

4.2.1.1 anonymous enum

`anonymous enum`

enum of all errors possible in rec_desc and all of its dependancies

Enumerator

E_MEM_ALLOC	error in allocating memory
E_OVERFLOW	error in dynamic memory overflow
E_UNDERFLOW	error in dynamic memory underflow
E_UNEXPECTED_SYMBOL	error in case of unexpected symbol
E_ZERO_DIVISION	arithmetic exception in case of division on zero
E_UNKNOWN_VAR	error in case variable lookup
E_UNBALANCED_BRACKET	error in case of an unbalanced bracket

4.2.2 Function Documentation

4.2.2.1 err_print()

```
void err_print (
            int err )
```

print an error message according to the error code

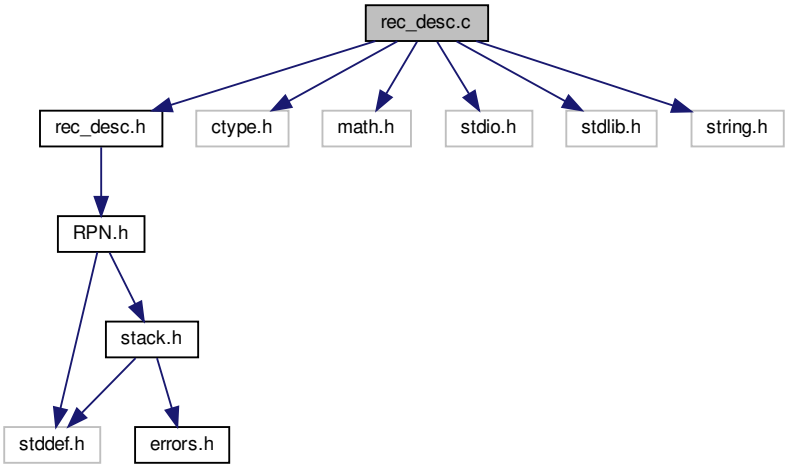
Parameters

err	error code

4.3 rec_desc.c File Reference

```
#include "rec_desc.h"
#include <ctype.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for rec_desc.c:



Macros

- #define SAFE(call)

Functions

- int parse_fact (Expression *expr, RPN *stack_mach)
- int parse_product (Expression *expr, RPN *stack_mach)
- int parse_sum (Expression *expr, RPN *stack_mach)
- int parse_literal (Expression *expr, RPN *stack_mach)
- int parse_variable (Expression *expr, RPN *stack_mach)
- int add_elem (const void *elem, Size_elem size, Stack *stack)
- int sum_double (const void *elem, Size_elem size, Stack *stack)
- int sub_double (const void *elem, Size_elem size, Stack *stack)
- int mult_double (const void *elem, Size_elem size, Stack *stack)
- int div_double (const void *elem, Size_elem size, Stack *stack)
- int neg (const void *elem, Size_elem size, Stack *stack)
- int lookup_var (const void *elem, Size_elem size, Stack *stack)
- int put_var_in_RPN (RPN *expression, char *name, int name_size)
- int put_number_in_RPN (RPN *expression, double elem)
- int put_func_in_RPN (RPN *expression, Calculate_elem func_in)
- int compute_expression (Expression *expr, double *res)
- int init_expression (Expression *expr, char *input)
- void finalize_expression (Expression *expr)

4.3.1 Macro Definition Documentation

4.3.1.1 SAFE

```
#define SAFE(
               call )
```

Value:

```
    do {                                            \
        if ((flag = call) != 0) return flag; \
    } while (0)
```

4.3.2 Function Documentation

4.3.2.1 add_elem()

```
int add_elem (
             const void * elem,
             Size_elem size,
             Stack * stack )
```

4.3.2.2 compute_expression()

```
int compute_expression (
             Expression * expr,
             double * res )
```

compute an expression

Parameters

expr	expression to compute
res	result of the computation

Exceptions

E_UNEXPECTED_SYMBOL	Thrown in case of error in parsing a string

Returns

error code

4.3.2.3 div_double()

```
int div_double (
             const void * elem,
```

```
            Size_elem size,
            Stack * stack )
```

4.3.2.4 finalize_expression()

```
void finalize_expression (
            Expression * expr )
```

finalize an expression

Parameters

expr	expression to finalize

4.3.2.5 init_expression()

```
int init_expression (
            Expression * expr,
            char * input )
```

initialize an expression

Parameters

expr	which expression to initialize
input	string with which to initialize the expression

Exceptions

E_MEM_ALLOC	Thrown in case of memory allocation error

Returns

error code

4.3.2.6 lookup_var()

```
int lookup_var (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

4.3.2.7 mult_double()

```
int mult_double (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

4.3.2.8 neg()

```
int neg (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

4.3.2.9 parse_fact()

```
int parse_fact (
            Expression * expr,
            RPN * stack_mach )
```

4.3.2.10 parse_literal()

```
int parse_literal (
            Expression * expr,
            RPN * stack_mach )
```

4.3.2.11 parse_product()

```
int parse_product (
            Expression * expr,
            RPN * stack_mach )
```

4.3.2.12 parse_sum()

```
int parse_sum (
            Expression * expr,
            RPN * stack_mach )
```

4.3.2.13 parse_variable()

```
int parse_variable (
            Expression * expr,
            RPN * stack_mach )
```

4.3.2.14 put_func_in_RPN()

```
int put_func_in_RPN (
            RPN * expression,
            Calculate_elem func_in )
```

4.3.2.15 put_number_in_RPN()

```
int put_number_in_RPN (
            RPN * expression,
            double elem )
```

4.3.2.16 put_var_in_RPN()

```
int put_var_in_RPN (
            RPN * expression,
            char * name,
            int name_size )
```

4.3.2.17 sub_double()

```
int sub_double (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

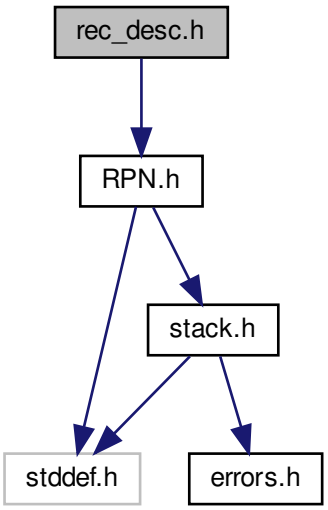
4.3.2.18 sum_double()

```
int sum_double (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

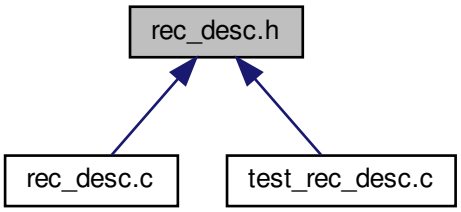
4.4 rec_desc.h File Reference

```
#include "RPN.h"
```

Include dependency graph for rec_desc.h:



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



Data Structures

- struct Expression

Functions

- int compute_expression (Expression *expr, double *res)
- int init_expression (Expression *expr, char *input)
- void finalize_expression (Expression *expr)

4.4.1 Function Documentation

4.4.1.1 compute_expression()

```
int compute_expression (
            Expression * expr,
            double * res )
```

compute an expression

Parameters

expr	expression to compute
res	result of the computation

Exceptions

E_UNEXPECTED_SYMBOL	Thrown in case of error in parsing a string

Returns

error code

4.4.1.2 finalize_expression()

```
void finalize_expression (
            Expression * expr )
```

finalize an expression

Parameters

expr	expression to finalize

4.4.1.3 init_expression()

```
int init_expression (
            Expression * expr,
            char * input )
```

initialize an expression

Parameters

expr	which expression to initialize
input	string with which to initialize the expression

Exceptions

E_MEM_ALLOC	Thrown in case of memory allocation error

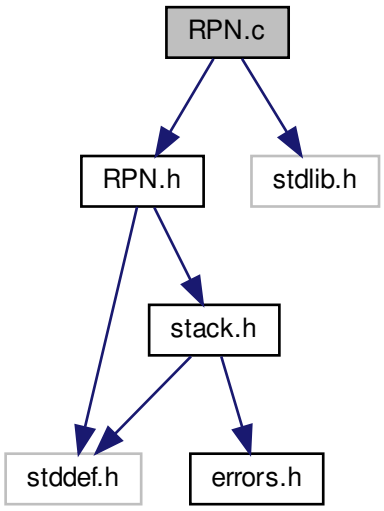
Returns

error code

4.5 RPN.c File Reference

```
#include "RPN.h"
#include <stdlib.h>
```

Include dependency graph for RPN.c:



Macros

- #define SAFE(call)

Functions

- int RPN_compute (RPN *notation, void *res, size_t res_size)
- int RPN_init (RPN *notation, size_t size)
- void RPN_finalize (RPN *notation)

4.5.1 Macro Definition Documentation

4.5.1.1 SAFE

```
#define SAFE(
              call )
```

Value:

```
    do {                            \
        if ((flag = call) != 0) {   \
            stack_finalize(&stack); \
            return flag;            \
        }                           \
    } while (0)
```

4.5.2 Function Documentation

4.5.2.1 RPN_compute()

```
int RPN_compute (
             RPN * notation,
             void * res,
             size_t res_size )
```

compute the RPN

Parameters

notation	notation to compute
res	result of the computation
res_size	size of res

Returns

error code

4.5.2.2 RPN_finalize()

```
void RPN_finalize (
             RPN * notation )
```

finalize an RPN

Parameters

notation	RPN to finalize

4.5.2.3 RPN_init()

```
int RPN_init (
            RPN * notation,
            size_t size )
```

initialize an RPN

Parameters

notation	RPN to initialize
size	size of the notation

Exceptions

E_MEM_ALLOC	Thrown in case of memory allocation error

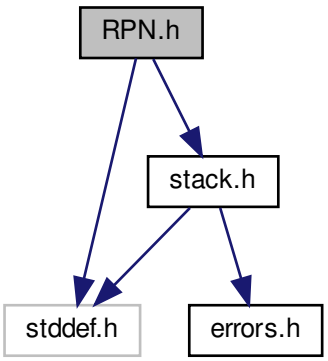
Returns

error code

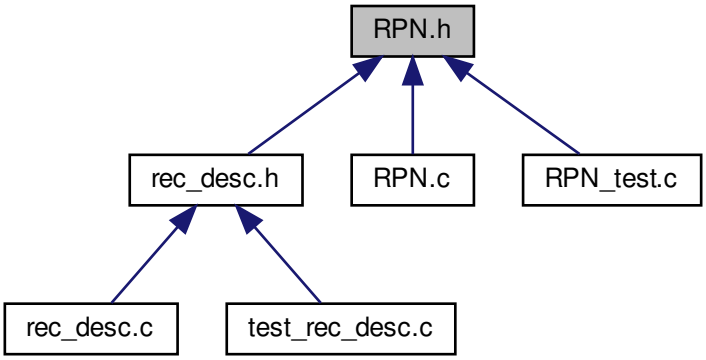
4.6 RPN.h File Reference

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

Include dependency graph for RPN.h:



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



Data Structures

- struct RPN

Typedefs

- typedef int(∗ Calculate_elem) (const void ∗elem, Size_elem size, Stack ∗stack)

Functions

- int RPN_compute (RPN ∗notation, void ∗res, size_t res_size)
- int RPN_init (RPN ∗notation, size_t size)
- void RPN_finalize (RPN ∗notation)

4.6.1 Typedef Documentation

4.6.1.1 Calculate_elem

```
typedef int(* Calculate_elem) (const void *elem, Size_elem size, Stack *stack)
```

4.6.2 Function Documentation

4.6.2.1 RPN_compute()

```
int RPN_compute (
            RPN * notation,
            void * res,
            size_t res_size )
```

compute the RPN

Parameters

notation	notation to compute
res	result of the computation
res_size	size of res

Returns

error code

4.6.2.2 RPN_finalize()

```
void RPN_finalize (
            RPN * notation )
```

finalize an RPN

Parameters

notation	RPN to finalize

4.6.2.3 RPN_init()

```
int RPN_init (
            RPN * notation,
            size_t size )
```

initialize an RPN

Parameters

notation	RPN to initialize
size	size of the notation

Exceptions

E_MEM_ALLOC	Thrown in case of memory allocation error

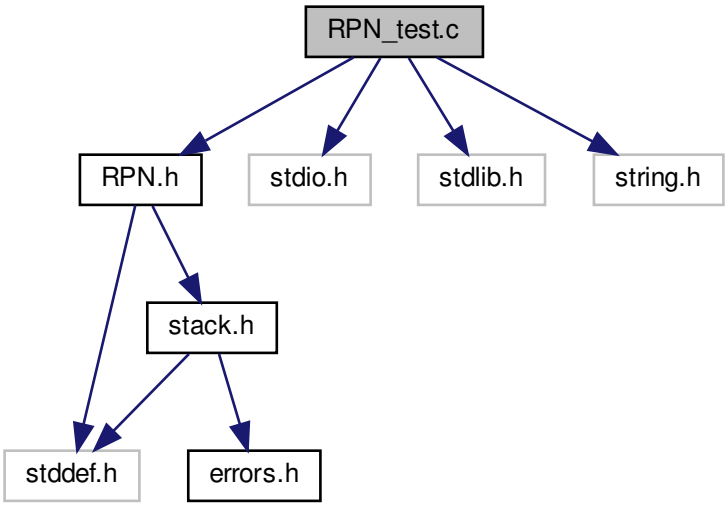
Returns

error code

4.7 RPN_test.c File Reference

```
#include "RPN.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for RPN_test.c:



Functions

- int add_double (const void ∗elem, Size_elem size, Stack ∗stack)
- int sum_double (const void ∗elem, Size_elem size, Stack ∗stack)
- int mult_double (const void ∗elem, Size_elem size, Stack ∗stack)
- int main (void)

4.7.1 Function Documentation

4.7.1.1 add_double()

```
int add_double (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

4.7.1.2 main()

```
int main (
            void  )
```

4.7.1.3 mult_double()

```
int mult_double (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

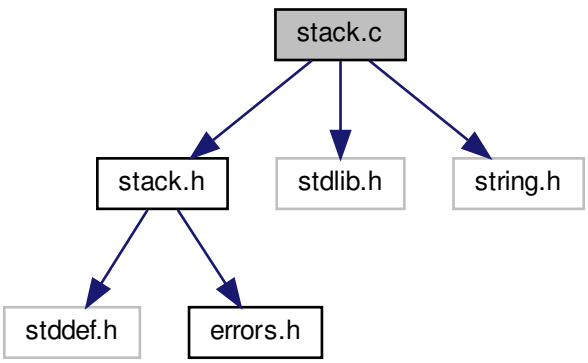
4.7.1.4 sum_double()

```
int sum_double (
            const void * elem,
            Size_elem size,
            Stack * stack )
```

4.8 stack.c File Reference

```
#include "stack.h"
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for stack.c:



Macros

- #define MEM_SAFE(call)

Functions

- int stack_init (Stack *stack, size_t data_size)
- void stack_finalize (Stack *stack)
- int stack_pop (Stack *stack, void *resp, size_t size_res)
- int stack_push (Stack *stack, const void *resp, size_t size_res)

4.8.1 Macro Definition Documentation

4.8.1.1 MEM_SAFE

```
#define MEM_SAFE(
                call )
```

Value:

```
    do {                            \
        if ((call) == NULL) {       \
            stack_finalize(stack); \
            return E_MEM_ALLOC;     \
        }                           \
    } while (0)
```

4.8.2 Function Documentation

4.8.2.1 stack_finalize()

```
void stack_finalize (
              Stack * stack )
```

finalize a stack

Parameters

stack	stack to finalize

4.8.2.2 stack_init()

```
int stack_init (
```

```
            Stack * stack,
            size_t data_size )
```

initialize a stack

Parameters

stack	stack to initialize
data_size	maximum size of a stack

Exceptions

E_MEM_ALLOC	Thrown in case of memory allocation error

Returns

error code

4.8.2.3 stack_pop()

```
int stack_pop (
            Stack * stack,
            void * resp,
            size_t size_res )
```

pop an element to the stack

Parameters

stack	stack from which to pop
resp	array to which write the result
size_res	size of resp. In case of overflow returns an error

Exceptions

E_UNDERFLOW	Thrown in case of popping empty or corrupted stack
E_OVERFLOW	Thrown if the popped memory overflows resp

Returns

error code

4.8.2.4 stack_push()

```
int stack_push (
            Stack * stack,
```

```
const void * resp,
size_t size_res )
```

push an element to the stack

Parameters

stack	stack to which to push
resp	array to which write the result
size_res	size of resp. In case of overflow returns an error

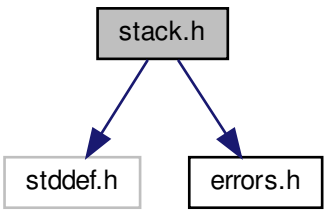
Exceptions

E_OVERFLOW	Thrown in case ov overflow of allocated memory

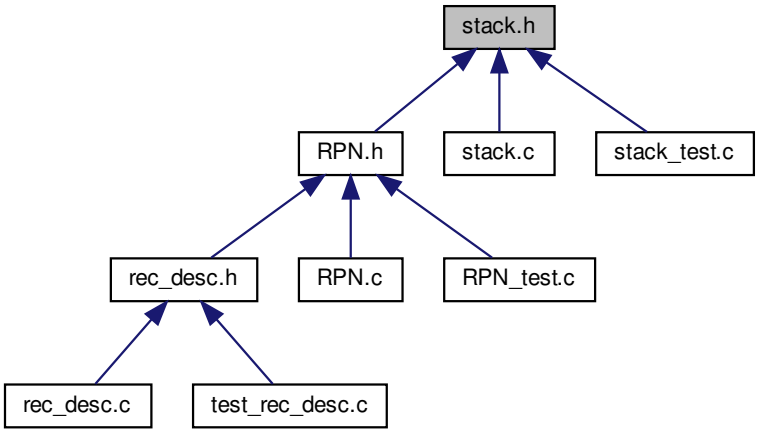
4.9 stack.h File Reference

```
#include <stddef.h>
#include "errors.h"
```

Include dependency graph for stack.h:



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



Data Structures

- struct Stack

  structure of a stack

Typedefs

- typedef char Size_elem

Functions

- int stack_init (Stack *stack, size_t data_size)
- void stack_finalize (Stack *stack)
- int stack_pop (Stack *stack, void *resp, size_t size_res)
- int stack_push (Stack *stack, const void *resp, size_t size_res)

4.9.1 Typedef Documentation

4.9.1.1 Size_elem

```
typedef char Size_elem
```

4.9.2 Function Documentation

4.9.2.1 stack_finalize()

```
void stack_finalize (
            Stack * stack )
```

finalize a stack

Parameters

stack	stack to finalize

4.9.2.2 stack_init()

```
int stack_init (
            Stack * stack,
            size_t data_size )
```

initialize a stack

Parameters

stack	stack to initialize
data_size	maximum size of a stack

Exceptions

E_MEM_ALLOC	Thrown in case of memory allocation error

Returns

error code

4.9.2.3 stack_pop()

```
int stack_pop (
            Stack * stack,
            void * resp,
            size_t size_res )
```

pop an element to the stack

Parameters

stack	stack from which to pop
resp	array to which write the result
size_res	size of resp. In case of overflow returns an error

Exceptions

E_UNDERFLOW	Thrown in case of popping empty or corrupted stack
E_OVERFLOW	Thrown if the popped memory overflows resp

Returns

error code

4.9.2.4 stack_push()

```
int stack_push (
            Stack * stack,
            const void * resp,
            size_t size_res )
```

push an element to the stack

Parameters

stack	stack to which to push
resp	array to which write the result
size_res	size of resp. In case of overflow returns an error

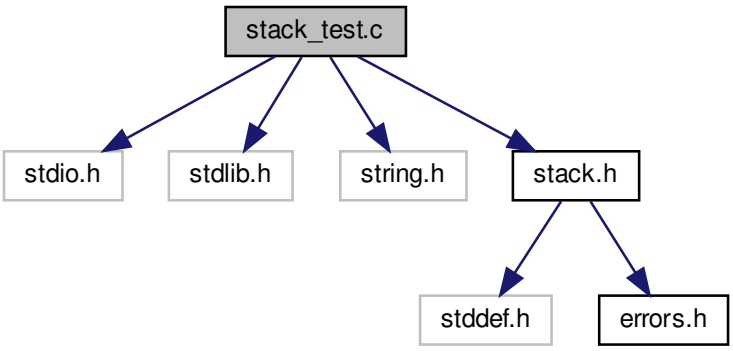
Exceptions

E_OVERFLOW	Thrown in case ov overflow of allocated memory

4.10 stack_test.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "stack.h"
```

Include dependency graph for stack_test.c:



Functions

- int main (void)

4.10.1 Function Documentation

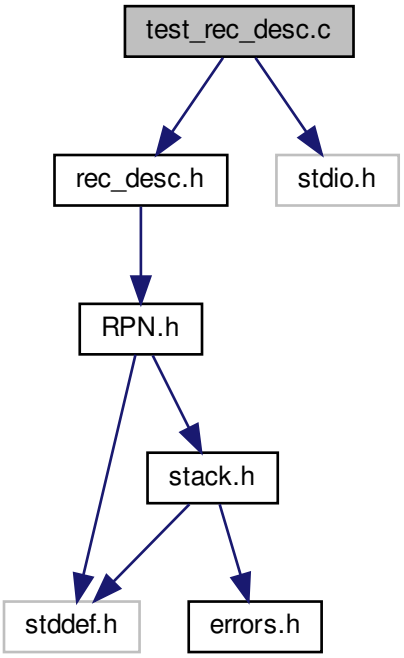
4.10.1.1 main()

```
int main (
            void  )
```

4.11 test_rec_desc.c File Reference

```
#include "rec_desc.h"
#include <stdio.h>
```

Include dependency graph for test_rec_desc.c:



Functions

- int main (void)

4.11.1 Function Documentation

4.11.1.1 main()

```
int main (
            void  )
```

Index