

Graphs

Generated by Doxygen 1.8.18

| | |
|---------------------------------------|----------|
| 1 Data Structure Index | 1 |
| 1.1 Data Structures | 1 |
| 2 File Index | 3 |
| 2.1 File List | 3 |
| 3 Data Structure Documentation | 5 |
| 3.1 Expression Struct Reference | 5 |
| 3.1.1 Detailed Description | 5 |
| 3.1.2 Field Documentation | 5 |
| 3.1.2.1 curpointer | 5 |
| 3.1.2.2 string_form | 6 |
| 3.2 RPN Struct Reference | 6 |
| 3.2.1 Detailed Description | 6 |
| 3.2.2 Field Documentation | 6 |
| 3.2.2.1 data | 6 |
| 3.2.2.2 data_size | 6 |
| 3.2.2.3 occupied | 7 |
| 3.3 Stack Struct Reference | 7 |
| 3.3.1 Detailed Description | 7 |
| 3.3.2 Field Documentation | 7 |
| 3.3.2.1 cur_size | 7 |
| 3.3.2.2 data | 8 |
| 3.3.2.3 stack_size | 8 |
| 3.3.2.4 stack_top | 8 |
| 4 File Documentation | 9 |
| 4.1 errors.c File Reference | 9 |
| 4.1.1 Function Documentation | 9 |
| 4.1.1.1 err_print() | 9 |
| 4.2 errors.h File Reference | 10 |
| 4.2.1 Enumeration Type Documentation | 10 |
| 4.2.1.1 anonymous enum | 10 |
| 4.2.2 Function Documentation | 11 |
| 4.2.2.1 err_print() | 11 |
| 4.3 rec_desc.c File Reference | 11 |
| 4.3.1 Macro Definition Documentation | 12 |
| 4.3.1.1 SAFE | 13 |
| 4.3.2 Function Documentation | 13 |
| 4.3.2.1 add_elem() | 13 |
| 4.3.2.2 compute_expression() | 13 |
| 4.3.2.3 div_double() | 13 |
| 4.3.2.4 finalize_expression() | 14 |

| | |
|--------------------------------------|----|
| 4.3.2.5 init_expression() | 14 |
| 4.3.2.6 lookup_var() | 14 |
| 4.3.2.7 mult_double() | 15 |
| 4.3.2.8 neg() | 15 |
| 4.3.2.9 parse_fact() | 15 |
| 4.3.2.10 parse_literal() | 15 |
| 4.3.2.11 parse_product() | 15 |
| 4.3.2.12 parse_sum() | 15 |
| 4.3.2.13 parse_variable() | 16 |
| 4.3.2.14 put_func_in_RPN() | 16 |
| 4.3.2.15 put_number_in_RPN() | 16 |
| 4.3.2.16 put_var_in_RPN() | 16 |
| 4.3.2.17 sub_double() | 16 |
| 4.3.2.18 sum_double() | 16 |
| 4.4 rec_desc.h File Reference | 17 |
| 4.4.1 Function Documentation | 18 |
| 4.4.1.1 compute_expression() | 18 |
| 4.4.1.2 finalize_expression() | 18 |
| 4.4.1.3 init_expression() | 18 |
| 4.5 RPN.c File Reference | 19 |
| 4.5.1 Macro Definition Documentation | 20 |
| 4.5.1.1 SAFE | 20 |
| 4.5.2 Function Documentation | 20 |
| 4.5.2.1 RPN_compute() | 20 |
| 4.5.2.2 RPN_finalize() | 20 |
| 4.5.2.3 RPN_init() | 21 |
| 4.6 RPN.h File Reference | 21 |
| 4.6.1 Typedef Documentation | 22 |
| 4.6.1.1 Calculate_elem | 22 |
| 4.6.2 Function Documentation | 22 |
| 4.6.2.1 RPN_compute() | 22 |
| 4.6.2.2 RPN_finalize() | 23 |
| 4.6.2.3 RPN_init() | 23 |
| 4.7 RPN_test.c File Reference | 24 |
| 4.7.1 Function Documentation | 24 |
| 4.7.1.1 add_double() | 24 |
| 4.7.1.2 main() | 25 |
| 4.7.1.3 mult_double() | 25 |
| 4.7.1.4 sum_double() | 25 |
| 4.8 stack.c File Reference | 25 |
| 4.8.1 Macro Definition Documentation | 26 |
| 4.8.1.1 MEM_SAFE | 26 |

| | |
|-------------------------------------|-----------|
| 4.8.2 Function Documentation | 26 |
| 4.8.2.1 stack_finalize() | 26 |
| 4.8.2.2 stack_init() | 26 |
| 4.8.2.3 stack_pop() | 27 |
| 4.8.2.4 stack_push() | 27 |
| 4.9 stack.h File Reference | 28 |
| 4.9.1 Typedef Documentation | 29 |
| 4.9.1.1 Size_elem | 29 |
| 4.9.2 Function Documentation | 29 |
| 4.9.2.1 stack_finalize() | 30 |
| 4.9.2.2 stack_init() | 30 |
| 4.9.2.3 stack_pop() | 30 |
| 4.9.2.4 stack_push() | 31 |
| 4.10 stack_test.c File Reference | 31 |
| 4.10.1 Function Documentation | 32 |
| 4.10.1.1 main() | 32 |
| 4.11 test_rec_desc.c File Reference | 32 |
| 4.11.1 Function Documentation | 33 |
| 4.11.1.1 main() | 33 |
| Index | 35 |

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|----------------------|---|
| Expression | 5 |
| RPN | 6 |
| Stack | |
| Structure of a stack | 7 |

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

| | |
|-----------------|----|
| errors.c | 9 |
| errors.h | 10 |
| rec_desc.c | 11 |
| rec_desc.h | 17 |
| RPN.c | 19 |
| RPN.h | 21 |
| RPN_test.c | 24 |
| stack.c | 25 |
| stack.h | 28 |
| stack_test.c | 31 |
| test_rec_desc.c | 32 |

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:

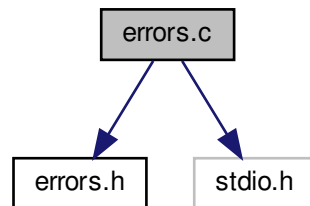
- [stack.h](#)

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 )
```

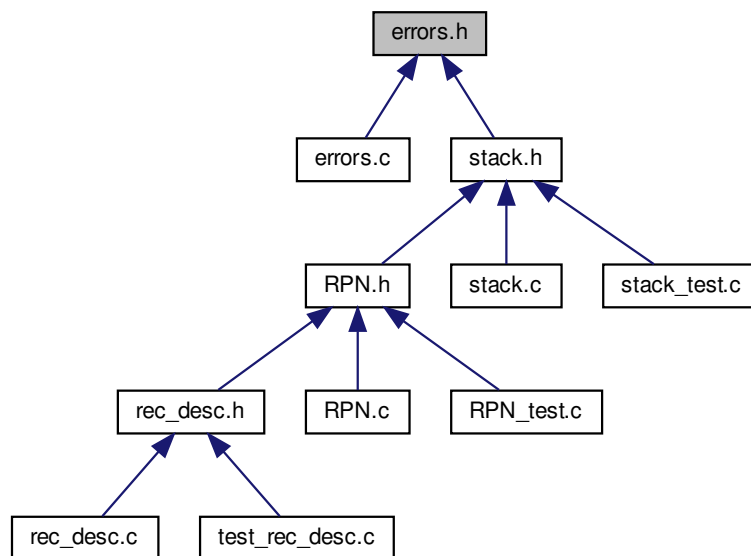
print an error message according to the error code

Parameters

| | |
|------------|------------|
| <i>err</i> | 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 dependencies

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

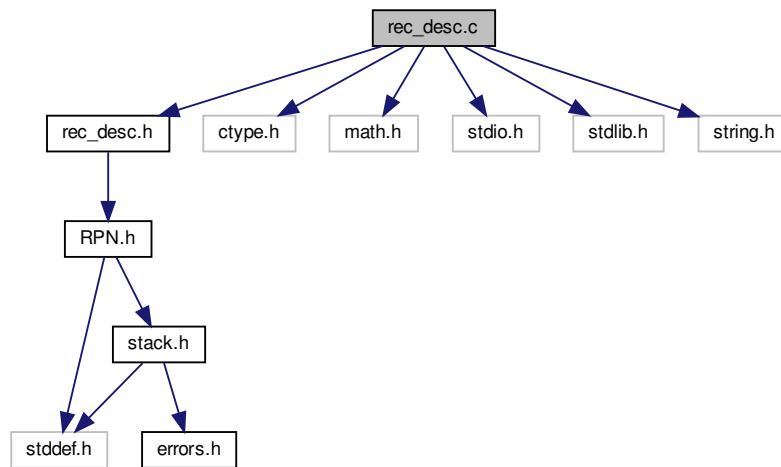
Parameters

| | |
|------------|------------|
| <i>err</i> | 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

| | |
|-------------|---------------------------|
| <i>expr</i> | expression to compute |
| <i>res</i> | result of the computation |

Exceptions

| | |
|----------------------------|---|
| <i>E_UNEXPECTED_SYMBOL</i> | 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

| | |
|-------------|------------------------|
| <i>expr</i> | expression to finalize |
|-------------|------------------------|

4.3.2.5 init_expression()

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

initialize an expression

Parameters

| | |
|--------------|--|
| <i>expr</i> | which expression to initialize |
| <i>input</i> | string with which to initialize the expression |

Exceptions

| | |
|--------------------|---|
| <i>E_MEM_ALLOC</i> | 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 )
```

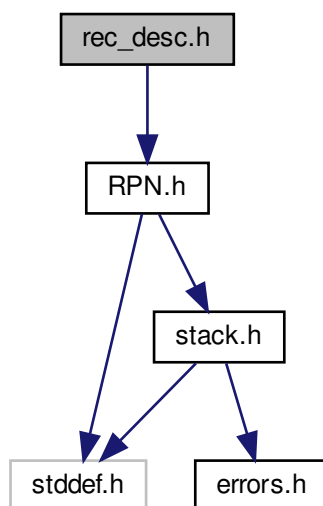
4.3.2.18 sum_double()

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

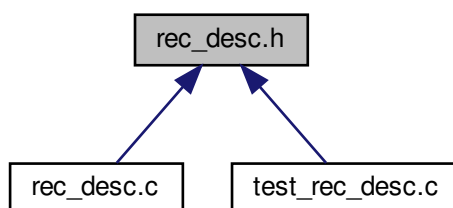
4.4 rec_desc.h File Reference

```
#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

| | |
|-------------|---------------------------|
| <i>expr</i> | expression to compute |
| <i>res</i> | result of the computation |

Exceptions

| | |
|----------------------------|---|
| <i>E_UNEXPECTED_SYMBOL</i> | 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

| | |
|-------------|------------------------|
| <i>expr</i> | expression to finalize |
|-------------|------------------------|

4.4.1.3 init_expression()

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

initialize an expression

Parameters

| | |
|--------------|--|
| <i>expr</i> | which expression to initialize |
| <i>input</i> | string with which to initialize the expression |

Exceptions

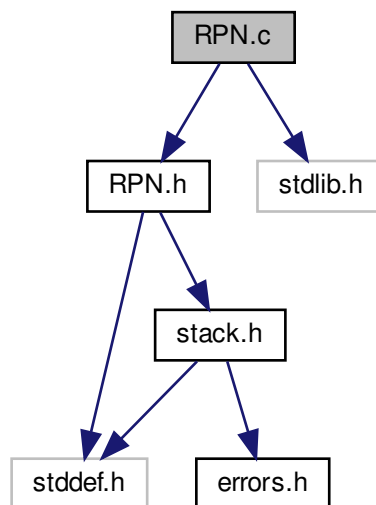
| | |
|--------------------|---|
| <i>E_MEM_ALLOC</i> | Thrown in case of memory allocation error |
|--------------------|---|

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

| | |
|-----------------|---------------------------|
| <i>notation</i> | notation to compute |
| <i>res</i> | result of the computation |
| <i>res_size</i> | size of res |

Returns

error code

4.5.2.2 RPN_finalize()

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

finalize an *RPN*

Parameters

| | |
|-----------------|-----------------|
| <i>notation</i> | RPN to finalize |
|-----------------|-----------------|

4.5.2.3 RPN_init()

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

initialize an RPN

Parameters

| | |
|-----------------|----------------------|
| <i>notation</i> | RPN to initialize |
| <i>size</i> | size of the notation |

Exceptions

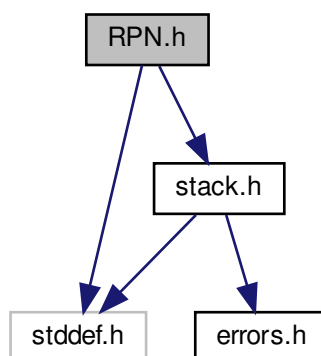
| | |
|--------------------|---|
| <i>E_MEM_ALLOC</i> | Thrown in case of memory allocation error |
|--------------------|---|

Returns

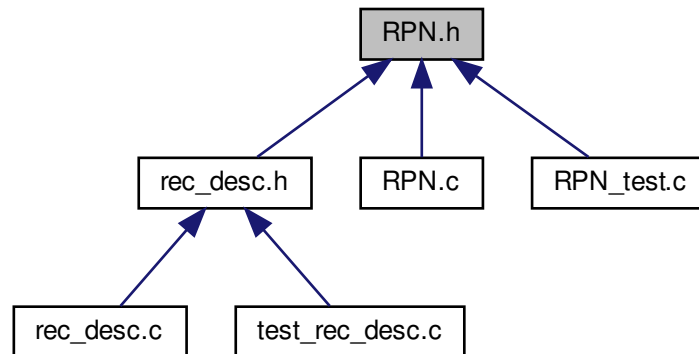
error code

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

| | |
|-----------------|---------------------------|
| <i>notation</i> | notation to compute |
| <i>res</i> | result of the computation |
| <i>res_size</i> | size of res |

Returns

error code

4.6.2.2 RPN_finalize()

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

finalize an [RPN](#)

Parameters

| | |
|-----------------|---------------------------------|
| <i>notation</i> | RPN to finalize |
|-----------------|---------------------------------|

4.6.2.3 RPN_init()

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

initialize an [RPN](#)

Parameters

| | |
|-----------------|-----------------------------------|
| <i>notation</i> | RPN to initialize |
| <i>size</i> | size of the notation |

Exceptions

| | |
|--------------------|---|
| <i>E_MEM_ALLOC</i> | Thrown in case of memory allocation error |
|--------------------|---|

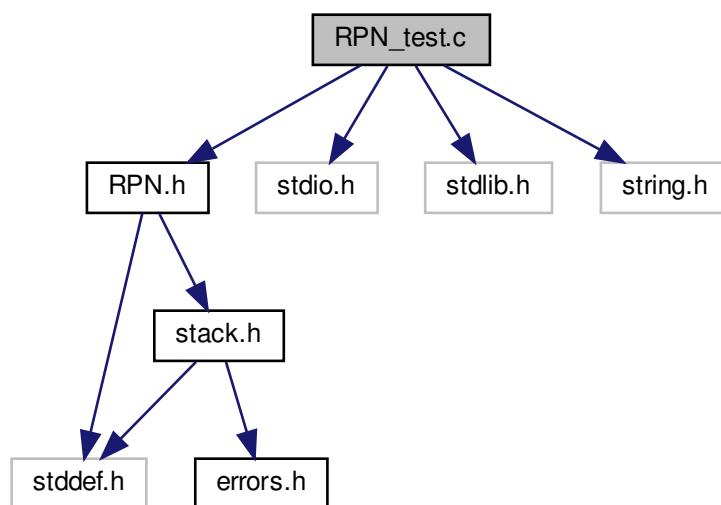
Returns

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()

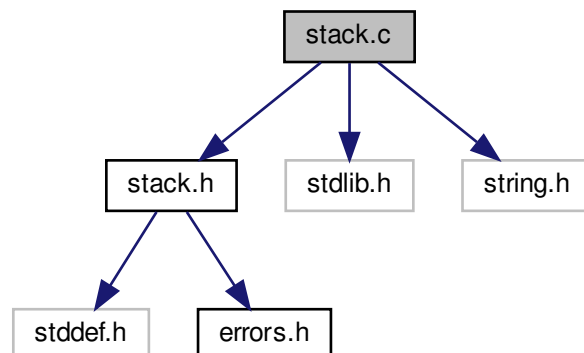
```
int mult_double (  
    const void * elem,  
    Size_elem size,  
    Stack * stack )
```

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

| | |
|--------------------|-------------------|
| <code>stack</code> | stack to finalize |
|--------------------|-------------------|

4.8.2.2 stack_init()

```
int stack_init (  
    size_t data_size )
```



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

initialize a stack

Parameters

| | |
|------------------|-------------------------|
| <i>stack</i> | stack to initialize |
| <i>data_size</i> | maximum size of a stack |

Exceptions

| | |
|--------------------|---|
| <i>E_MEM_ALLOC</i> | 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

| | |
|-----------------|--|
| <i>stack</i> | stack from which to pop |
| <i>resp</i> | array to which write the result |
| <i>size_res</i> | size of resp. In case of overflow returns an error |

Exceptions

| | |
|--------------------|--|
| <i>E_UNDERFLOW</i> | Thrown in case of popping empty or corrupted stack |
| <i>E_OVERFLOW</i> | 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

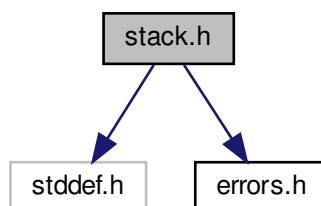
| | |
|-----------------|--|
| <i>stack</i> | stack to which to push |
| <i>resp</i> | array to which write the result |
| <i>size_res</i> | size of resp. In case of overflow returns an error |

Exceptions

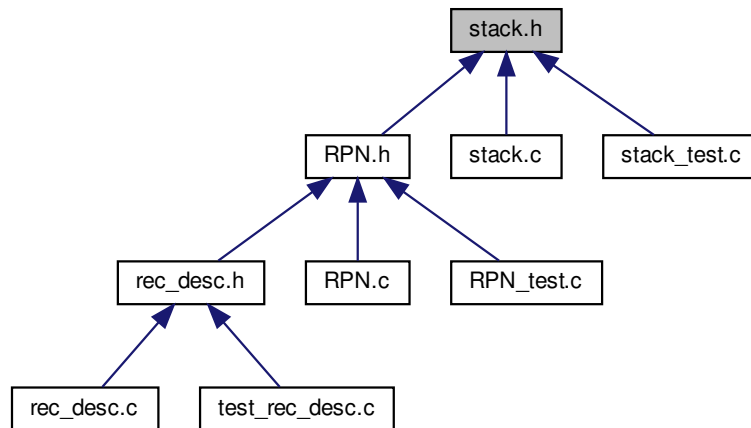
| | |
|-------------------|--|
| <i>E_OVERFLOW</i> | Thrown in case of overflow of allocated memory |
|-------------------|--|

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

| | |
|--------------|-------------------|
| <i>stack</i> | stack to finalize |
|--------------|-------------------|

4.9.2.2 stack_init()

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

initialize a stack

Parameters

| | |
|------------------|-------------------------|
| <i>stack</i> | stack to initialize |
| <i>data_size</i> | maximum size of a stack |

Exceptions

| | |
|--------------------|---|
| <i>E_MEM_ALLOC</i> | 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

| | |
|-----------------|--|
| <i>stack</i> | stack from which to pop |
| <i>resp</i> | array to which write the result |
| <i>size_res</i> | size of resp. In case of overflow returns an error |

Exceptions

| | |
|--------------------|--|
| <i>E_UNDERFLOW</i> | Thrown in case of popping empty or corrupted stack |
| <i>E_OVERFLOW</i> | 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

| | |
|-----------------|--|
| <i>stack</i> | stack to which to push |
| <i>resp</i> | array to which write the result |
| <i>size_res</i> | size of resp. In case of overflow returns an error |

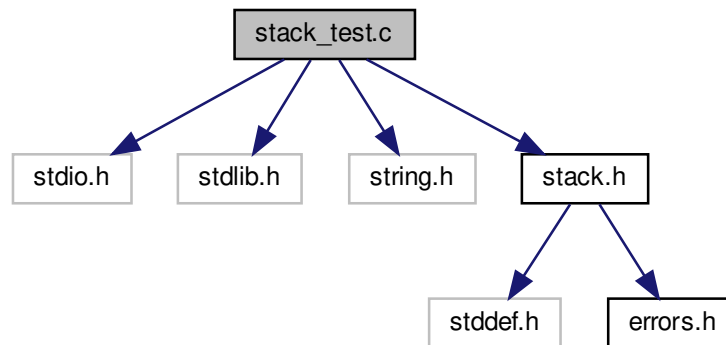
Exceptions

| | |
|-------------------|--|
| <i>E_OVERFLOW</i> | Thrown in case of 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

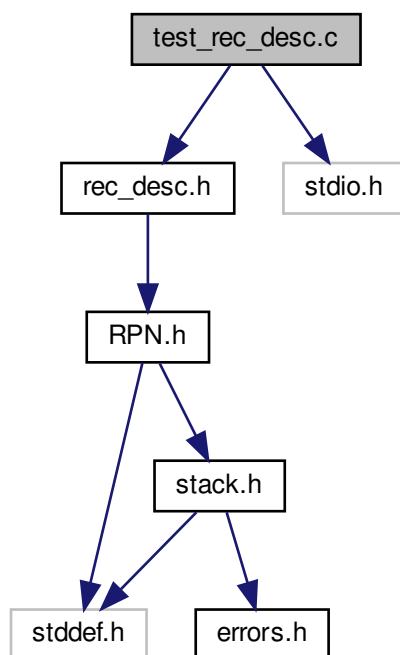
4.10.1.1 `main()`

```
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

- add_double
 - RPN_test.c, [24](#)
- add_elem
 - rec_desc.c, [13](#)
- Calculate_elem
 - RPN.h, [22](#)
- compute_expression
 - rec_desc.c, [13](#)
 - rec_desc.h, [18](#)
- cur_size
 - Stack, [7](#)
- curpointer
 - Expression, [5](#)
- data
 - RPN, [6](#)
 - Stack, [7](#)
- data_size
 - RPN, [6](#)
- div_double
 - rec_desc.c, [13](#)
- E_MEM_ALLOC
 - errors.h, [11](#)
- E_OVERFLOW
 - errors.h, [11](#)
- E_UNBALANCED_BRACKET
 - errors.h, [11](#)
- E_UNDERFLOW
 - errors.h, [11](#)
- E_UNEXPECTED_SYMBOL
 - errors.h, [11](#)
- E_UNKNOWN_VAR
 - errors.h, [11](#)
- E_ZERO_DIVISION
 - errors.h, [11](#)
- err_print
 - errors.c, [9](#)
 - errors.h, [11](#)
- errors.c, [9](#)
 - err_print, [9](#)
- errors.h, [10](#)
 - E_MEM_ALLOC, [11](#)
 - E_OVERFLOW, [11](#)
 - E_UNBALANCED_BRACKET, [11](#)
 - E_UNDERFLOW, [11](#)
 - E_UNEXPECTED_SYMBOL, [11](#)
 - E_UNKNOWN_VAR, [11](#)
 - E_ZERO_DIVISION, [11](#)
- err_print, [11](#)
- Expression, [5](#)
 - curpointer, [5](#)
 - string_form, [5](#)
- finalize_expression
 - rec_desc.c, [14](#)
 - rec_desc.h, [18](#)
- init_expression
 - rec_desc.c, [14](#)
 - rec_desc.h, [18](#)
- lookup_var
 - rec_desc.c, [14](#)
- main
 - RPN_test.c, [24](#)
 - stack_test.c, [32](#)
 - test_rec_desc.c, [33](#)
- MEM_SAFE
 - stack.c, [26](#)
- mult_double
 - rec_desc.c, [14](#)
 - RPN_test.c, [25](#)
- neg
 - rec_desc.c, [15](#)
- occupied
 - RPN, [6](#)
- parse_fact
 - rec_desc.c, [15](#)
- parse_literal
 - rec_desc.c, [15](#)
- parse_product
 - rec_desc.c, [15](#)
- parse_sum
 - rec_desc.c, [15](#)
- parse_variable
 - rec_desc.c, [15](#)
- put_func_in_RPN
 - rec_desc.c, [16](#)
- put_number_in_RPN
 - rec_desc.c, [16](#)
- put_var_in_RPN
 - rec_desc.c, [16](#)
- rec_desc.c, [11](#)
 - add_elem, [13](#)

- compute_expression, 13
- div_double, 13
- finalize_expression, 14
- init_expression, 14
- lookup_var, 14
- mult_double, 14
- neg, 15
- parse_fact, 15
- parse_literal, 15
- parse_product, 15
- parse_sum, 15
- parse_variable, 15
- put_func_in_RPN, 16
- put_number_in_RPN, 16
- put_var_in_RPN, 16
- SAFE, 12
- sub_double, 16
- sum_double, 16
- rec_desc.h, 17
 - compute_expression, 18
 - finalize_expression, 18
 - init_expression, 18
- RPN, 6
 - data, 6
 - data_size, 6
 - occupied, 6
- RPN.c, 19
 - RPN_compute, 20
 - RPN_finalize, 20
 - RPN_init, 21
 - SAFE, 20
- RPN.h, 21
 - Calculate_elem, 22
 - RPN_compute, 22
 - RPN_finalize, 23
 - RPN_init, 23
- RPN_compute
 - RPN.c, 20
 - RPN.h, 22
- RPN_finalize
 - RPN.c, 20
 - RPN.h, 23
- RPN_init
 - RPN.c, 21
 - RPN.h, 23
- RPN_test.c, 24
 - add_double, 24
 - main, 24
 - mult_double, 25
 - sum_double, 25
- SAFE
 - rec_desc.c, 12
 - RPN.c, 20
- Size_elem
 - stack.h, 29
- Stack, 7
 - cur_size, 7
 - data, 7
 - stack_size, 8
 - stack_top, 8
- stack.c, 25
 - MEM_SAFE, 26
 - stack_finalize, 26
 - stack_init, 26
 - stack_pop, 27
 - stack_push, 27
- stack.h, 28
 - Size_elem, 29
 - stack_finalize, 29
 - stack_init, 30
 - stack_pop, 30
 - stack_push, 31
- stack_finalize
 - stack.c, 26
 - stack.h, 29
- stack_init
 - stack.c, 26
 - stack.h, 30
- stack_pop
 - stack.c, 27
 - stack.h, 30
- stack_push
 - stack.c, 27
 - stack.h, 31
- stack_size
 - Stack, 8
- stack_test.c, 31
 - main, 32
- stack_top
 - Stack, 8
- string_form
 - Expression, 5
- sub_double
 - rec_desc.c, 16
- sum_double
 - rec_desc.c, 16
 - RPN_test.c, 25
- test_rec_desc.c, 32
 - main, 33