

Kalkulator ONP

Wygenerowano przez Doxygen 1.8.15



<b>1 Indeks struktur danych</b>	<b>1</b>
1.1 Struktury danych	1
<b>2 Indeks plików</b>	<b>3</b>
2.1 Lista plików	3
<b>3 Dokumentacja struktur danych</b>	<b>5</b>
3.1 Dokumentacja struktury args	5
3.1.1 Opis szczegółowy	5
3.2 Dokumentacja struktury handle	6
3.2.1 Opis szczegółowy	6
3.3 Dokumentacja struktury operation	6
3.3.1 Opis szczegółowy	6
3.4 Dokumentacja struktury stack	6
3.4.1 Opis szczegółowy	6
<b>4 Dokumentacja plików</b>	<b>7</b>
4.1 Dokumentacja pliku main.c	7
4.1.1 Dokumentacja zmiennych	7
4.1.1.1 asrt_count	7
4.2 Dokumentacja pliku misc.c	7
4.2.1 Dokumentacja definicji	8
4.2.1.1 arg_char	8
4.2.2 Dokumentacja funkcji	8
4.2.2.1 copy_path()	8
4.2.2.2 get_help()	9
4.2.2.3 is_number()	9
4.2.2.4 new_stack()	9
4.2.2.5 parse_args()	10
4.2.2.6 parse_exp()	10
4.2.2.7 read_text()	10
4.3 Dokumentacja pliku misc.h	11
4.3.1 Dokumentacja definicji	12
4.3.1.1 asrt	12
4.3.2 Dokumentacja definicji typów	12
4.3.2.1 args	12
4.3.3 Dokumentacja funkcji	12
4.3.3.1 copy_path()	12
4.3.3.2 get_help()	13
4.3.3.3 is_number()	13
4.3.3.4 new_stack()	13
4.3.3.5 parse_args()	13
4.3.3.6 parse_exp()	14

4.3.3.7 read_text()	14
4.3.4 Dokumentacja zmiennych	14
4.3.4.1 asrt_count	14
4.4 Dokumentacja pliku operations.c	15
4.4.1 Dokumentacja funkcji	15
4.4.1.1 get_operands()	15
4.4.1.2 get_operation()	15
4.4.1.3 memory_operation()	16
4.5 Dokumentacja pliku operations.h	16
4.5.1 Dokumentacja definicji	17
4.5.1.1 op_function_declr	17
4.5.2 Dokumentacja definicji typów	17
4.5.2.1 operation	17
4.5.3 Dokumentacja funkcji	17
4.5.3.1 get_operands()	17
4.5.3.2 get_operation()	18
4.5.3.3 memory_operation()	18
4.5.3.4 op_function_declr() [1/5]	19
4.5.3.5 op_function_declr() [2/5]	19
4.5.3.6 op_function_declr() [3/5]	19
4.5.3.7 op_function_declr() [4/5]	19
4.5.3.8 op_function_declr() [5/5]	19
4.6 Dokumentacja pliku stack.c	19
4.6.1 Dokumentacja funkcji	20
4.6.1.1 peek()	20
4.6.1.2 pop()	20
4.6.1.3 pulverize()	20
4.6.1.4 push()	21
4.7 Dokumentacja pliku stack.h	21
4.7.1 Dokumentacja definicji typów	22
4.7.1.1 handle	22
4.7.1.2 stack	22
4.7.2 Dokumentacja funkcji	22
4.7.2.1 peek()	22
4.7.2.2 pop()	22
4.7.2.3 pulverize()	23
4.7.2.4 push()	23

# Rozdział 1

## Indeks struktur danych

### 1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

<a href="#">args</a>	5
<a href="#">handle</a>	6
<a href="#">operation</a>	6
<a href="#">stack</a>	6



## Rozdział 2

# Indeks plików

### 2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

<a href="#">main.c</a>	7
<a href="#">misc.c</a>	7
<a href="#">misc.h</a>	11
<a href="#">operations.c</a>	15
<a href="#">operations.h</a>	16
<a href="#">stack.c</a>	19
<a href="#">stack.h</a>	21





## Rozdział 3

# Dokumentacja struktur danych

### 3.1 Dokumentacja struktury args

```
#include <misc.h>
```

#### Pola danych

- int [should\\_exit](#)  
*informacja o błędnych argumentach*
- char \* [infile](#)  
*ścieżka do pliku wejściowego.*
- char \* [outfile](#)  
*ścieżka do pliku wyjściowego.*
- char [whitespace](#)  
*znak odstępu między kolejnymi operatorami/operandami*
- char [comment](#)  
*znak/operator rozpoczęcia komentarza*
- char [quit](#)  
*znak/operator zamknięcia programu*
- char [precision](#)  
*długość rozwinięcia dziesiętnego dla wyjścia programu*
- char [deleter](#)  
*znak/operator czyszczenia stosu*
- char [memory](#)  
*znak/operator rozpoczynający operatory pamięciowe*

#### 3.1.1 Opis szczegółowy

argumenty wiersza poleceń

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [misc.h](#)

## 3.2 Dokumentacja struktury handle

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

### Pola danych

- [stack](#) \* **head**
- double **memory**
- unsigned int **stacksize**

### 3.2.1 Opis szczegółowy

uchwyt stosu

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [stack.h](#)

## 3.3 Dokumentacja struktury operation

```
#include <operations.h>
```

### Pola danych

- double(\* **fn\_ptr**)(const double operands[])
- const char \* **tag**
- unsigned int **num\_of\_operands**

### 3.3.1 Opis szczegółowy

element tablicy wskaźników na funkcje

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [operations.h](#)

## 3.4 Dokumentacja struktury stack

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

### Pola danych

- double **value**
- struct [stack](#) \* **next**

### 3.4.1 Opis szczegółowy

element stosu

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [stack.h](#)

## Rozdział 4

# Dokumentacja plików

### 4.1 Dokumentacja pliku main.c

```
#include "misc.h"  
#include "operations.h"  
#include "stack.h"
```

#### Funkcje

- int **main** (int argc, char \*\*argv)

#### Zmienne

- unsigned int `asrt_count` = 0

#### 4.1.1 Dokumentacja zmiennych

##### 4.1.1.1 asrt\_count

```
unsigned int asrt_count = 0
```

#### Autor

Tomasz Sitek

### 4.2 Dokumentacja pliku misc.c

```
#include <ctype.h>  
#include "misc.h"
```

## Definicje

- #define `arg_char`(`struct_member`)

## Funkcje

- void `get_help` ()
- bool `copy_path` (char \*\*`destination`, const char \*`source`)
- `args` `parse_args` (int `argc`, char \*\*`argv`)
- `handle` \* `new_stack` ()
- bool `is_number` (const char \*`ptr`, double \*`d`)
- bool `parse_exp` (char \*`exp`, `handle` \*const `top`, const `args` `config`, FILE \*`f_out`)
- bool `read_text` (const `args` `config`)

### 4.2.1 Dokumentacja definicji

#### 4.2.1.1 `arg_char`

```
#define arg_char(  
    struct_member )
```

##### Wartość:

```
if (strlen(argv[++i]) != 1) {  
    config.should_exit = true;  
} else {  
    if (strchr(restricted_characters, argv(i, 0)))  
        config.should_exit = true;  
    else {  
        char* ptr = strchr(config_chars, config.struct_member);  
        if (!ptr) {  
            fprintf(stderr, "FATAL ERROR: arg_char _%c_\n",  
                config.struct_member); \br/>            config.should_exit = true;  
        } else {  
            *ptr = argv(i, 0);  
            config.struct_member = argv(i, 0);  
        }  
    }  
}
```

makro wczytujące wartość argumentu wiersza poleceń będącego pojedynczym znakiem do konfiguracji programu

### 4.2.2 Dokumentacja funkcji

#### 4.2.2.1 `copy_path()`

```
bool copy_path (  
    char ** destination,  
    const char * source )
```

funkcja sprawdza poprawność podanej ścieżki dostępu do pliku, po czym alokuje pamięć i kopiuje ścieżkę

**Parametry**

<i>destination</i>	wskaźnik na pamięć, która będzie alokowana
<i>source</i>	ścieżka dostępu do pliku

**Zwraca**

true = powodzenie

**4.2.2.2 get\_help()**

```
void get_help ( )
```

funkcja wyświetla pomoc

**4.2.2.3 is\_number()**

```
bool is_number (
    const char * ptr,
    double * d )
```

funkcja sprawdza, czy podany string zawiera poprawną liczbę i zwraca ją

**Parametry**

<i>ptr</i>	string do sprawdzenia
<i>d</i>	wskaźnik na zmienną, do której zostanie zwrócona liczba

**Zwraca**

true = string zawiera liczbę

**4.2.2.4 new\_stack()**

```
handle* new_stack ( )
```

funkcja alokuje i inicjalizuje nowy uchwyt stosu

**Zwraca**

uchwyt stosu lub NULL w przypadku niepowodzenia alokacji

#### 4.2.2.5 parse\_args()

```
args parse_args (
    int argc,
    char ** argv )
```

funkcja przetwarza argumenty wiersza poleceń

##### Zwraca

struktura args przechowująca informacje istotne dla programu

#### 4.2.2.6 parse\_exp()

```
bool parse_exp (
    char * exp,
    handle *const top,
    const args config,
    FILE * f_out )
```

funkcja przetwarza wyrażenie w ONP podane w stringu

##### Parametry

<i>exp</i>	przetwarzany string
<i>top</i>	uchwyt stosu
<i>config</i>	konfiguracja programu
<i>f_out</i>	strumień wyjścia programu

##### Zwraca

true = powodzenie

#### 4.2.2.7 read\_text()

```
bool read_text (
    const args config )
```

funkcja

##### Parametry

<i>config</i>	konfiguracja programu
---------------	-----------------------

#### Zwraca

true = powodzenie

## 4.3 Dokumentacja pliku misc.h

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

### Struktury danych

- struct [args](#)

### Definicje

- #define **exp\_len\_max** 1024
- #define **restricted\_characters** "0123456789.+\*/^"
- #define **error\_msg** "You should not be seeing this message. \If you do, a critical error has occurred. Please contact the app developer.\n"
- #define **argv**(x, y) argv[x][y]
- #define **eq**(str1, str2) !strcmp(str1, str2)
- #define **asrt**(z)  
*inicjalizacja w [main.c](#)*

### Definicje typów

- typedef struct [args](#) [args](#)

### Funkcje

- void [get\\_help](#) ()
- bool [copy\\_path](#) (char \*\*destination, const char \*source)
- [args](#) [parse\\_args](#) (int argc, char \*\*argv)
- [handle](#) \* [new\\_stack](#) ()
- bool [is\\_number](#) (const char \*ptr, double \*d)
- bool [parse\\_exp](#) (char \*exp, [handle](#) \*const top, const [args](#) config, FILE \*f\_out)
- bool [read\\_text](#) (const [args](#) config)

### Zmienne

- unsigned int [asrt\\_count](#)





**Zwraca**

true = powodzenie

**4.3.3.2 get\_help()**

```
void get_help ( )
```

funkcja wyświetla pomoc

**4.3.3.3 is\_number()**

```
bool is_number (
    const char * ptr,
    double * d )
```

funkcja sprawdza, czy podany string zawiera poprawną liczbę i zwraca ją

**Parametry**

<i>ptr</i>	string do sprawdzenia
<i>d</i>	wskaźnik na zmienną, do której zostanie zwrócona liczba

**Zwraca**

true = string zawiera liczbę

**4.3.3.4 new\_stack()**

```
handle* new_stack ( )
```

funkcja alokuje i inicjalizuje nowy uchwyt stosu

**Zwraca**

uchwyt stosu lub NULL w przypadku niepowodzenia alokacji

**4.3.3.5 parse\_args()**

```
args parse_args (
    int argc,
    char ** argv )
```

funkcja przetwarza argumenty wiersza poleceń

**Zwraca**

struktura args przechowująca informacje istotne dla programu

#### 4.3.3.6 parse\_exp()

```
bool parse_exp (
    char * exp,
    handle *const top,
    const args config,
    FILE * f_out )
```

funkcja przetwarza wyrażenie w ONP podane w stringu

##### Parametry

<i>exp</i>	przetwarzany string
<i>top</i>	uchwyt stosu
<i>config</i>	konfiguracja programu
<i>f_out</i>	strumień wyjścia programu

##### Zwraca

true = powodzenie

#### 4.3.3.7 read\_text()

```
bool read_text (
    const args config )
```

funkcja

##### Parametry

<i>config</i>	konfiguracja programu
---------------	-----------------------

##### Zwraca

true = powodzenie

### 4.3.4 Dokumentacja zmiennych

#### 4.3.4.1 asrt\_count

```
unsigned int asrt_count
```

##### Autor

Tomasz Sitek

## 4.4 Dokumentacja pliku operations.c

```
#include <math.h>
#include "misc.h"
#include "operations.h"
```

### Funkcje

- double **op\_add** (const double operands[2])
- double **op\_subtract** (const double operands[2])
- double **op\_multiply** (const double operands[2])
- double **op\_divide** (const double operands[2])
- double **op\_pow** (const double operands[2])
- bool **memory\_operation** (**handle** \*const top, const char \*op)
- const **operation** \* **get\_operation** (const char \*str)
- double \* **get\_operands** (**handle** \*const top, unsigned int num\_of\_operands)

### 4.4.1 Dokumentacja funkcji

#### 4.4.1.1 get\_operands()

```
double* get_operands (
    handle *const top,
    unsigned int num_of_operands )
```

funkcja zwraca tablicę operandów

#### Parametry

<i>top</i>	uchwyt stosu
<i>num_of_operands</i>	żądana liczba operandów (a także rozmiar zwracanej tablicy)

#### Zwraca

wskaźnik na pierwszy element tablicy

#### 4.4.1.2 get\_operation()

```
const operation* get_operation (
    const char * str )
```

funkcja zwraca wskaźnik na funkcję realizującą odpowiednią operację

#### Parametry

<i>str</i>	operator
------------	----------

#### Zwraca

wskaźnik na funkcję lub NULL, jeżeli funkcja o podanym operatorze nie istnieje

#### 4.4.1.3 memory\_operation()

```
bool memory_operation (
    handle *const top,
    const char * op )
```

funkcja realizuje operację na pamięci kalkulatora w oparciu o podany operator pamięciowy

#### Parametry

<i>top</i>	uchwyt stosu
<i>op</i>	string zawierający operator pamięciowy

#### Zwraca

true = powodzenie

## 4.5 Dokumentacja pliku operations.h

```
#include "stack.h"
```

### Struktury danych

- struct `operation`

### Definicje

- #define `n_operations` 5
- #define `op_function_declr`(name) op\_##name(const double operands[ ])

### Definicje typów

- typedef struct `operation` `operation`

## Funkcje

- double `op_function_declr` (add)
- double `op_function_declr` (subtract)
- double `op_function_declr` (multiply)
- double `op_function_declr` (divide)
- double `op_function_declr` (pow)
- bool `memory_operation` (`handle` \*const top, const char \*op)
- const `operation` \* `get_operation` (const char \*str)
- double \* `get_operands` (`handle` \*const top, unsigned int num\_of\_operands)

### 4.5.1 Dokumentacja definicji

#### 4.5.1.1 `op_function_declr`

```
#define op_function_declr(  
    name ) op_##name(const double operands[])
```

makro deklarujące następujące funkcje

#### Parametry

<i>name</i>	nazwa funkcji
-------------	---------------

### 4.5.2 Dokumentacja definicji typów

#### 4.5.2.1 `operation`

```
typedef struct operation operation
```

element tablicy wskaźników na funkcje

### 4.5.3 Dokumentacja funkcji

#### 4.5.3.1 `get_operands()`

```
double* get_operands (  
    handle *const top,  
    unsigned int num_of_operands )
```

funkcja zwraca tablicę operandów

**Parametry**

<i>top</i>	uchwyt stosu
<i>num_of_operands</i>	żądana liczba operandów (a także rozmiar zwracanej tablicy)

**Zwraca**

wskaźnik na pierwszy element tablicy

**4.5.3.2 get\_operation()**

```
const operation* get_operation (
    const char * str )
```

funkcja zwraca wskaźnik na funkcję realizującą odpowiednią operację

**Parametry**

<i>str</i>	operator
------------	----------

**Zwraca**

wskaźnik na funkcję lub NULL, jeżeli funkcja o podanym operatorze nie istnieje

**4.5.3.3 memory\_operation()**

```
bool memory_operation (
    handle *const top,
    const char * op )
```

funkcja realizuje operację na pamięci kalkulatora w oparciu o podany operator pamięciowy

**Parametry**

<i>top</i>	uchwyt stosu
<i>op</i>	string zawierający operator pamięciowy

**Zwraca**

true = powodzenie

#### 4.5.3.4 op\_function\_declr() [1/5]

```
double op_function_declr (
    add )
```

funkcja realizująca dodawanie

#### 4.5.3.5 op\_function\_declr() [2/5]

```
double op_function_declr (
    subtract )
```

funkcja realizująca odejmowanie

#### 4.5.3.6 op\_function\_declr() [3/5]

```
double op_function_declr (
    multiply )
```

funkcja realizująca mnożenie

#### 4.5.3.7 op\_function\_declr() [4/5]

```
double op_function_declr (
    divide )
```

funkcja realizująca dzielenie

#### 4.5.3.8 op\_function\_declr() [5/5]

```
double op_function_declr (
    pow )
```

funkcja realizująca potęgowanie

## 4.6 Dokumentacja pliku stack.c

```
#include "misc.h"
#include "stack.h"
```

### Funkcje

- bool [push](#) ([handle](#) \*const top, const double d)
- bool [pop](#) ([handle](#) \*const top, double \*d)
- double [peek](#) ([handle](#) \*const top)
- void [pulverize](#) ([handle](#) \*const top)

## 4.6.1 Dokumentacja funkcji

### 4.6.1.1 peek()

```
double peek (
    handle *const top )
```

funkcja zwraca wartość liczby ze szczytu stosu bez zdejmowania jej

#### Parametry

<i>top</i>	uchwyt stosu
------------	--------------

#### Zwraca

wartość liczby ze szczytu stosu

### 4.6.1.2 pop()

```
bool pop (
    handle *const top,
    double * d )
```

funkcja zdejmuję liczbę ze szczytu stosu

#### Parametry

<i>top</i>	uchwyt stosu
<i>d</i>	wskaźnik na zmienną, do której zwracana jest liczba zdjęta ze stosu

#### Zwraca

true = powodzenie

### 4.6.1.3 pulverize()

```
void pulverize (
    handle *const top )
```

funkcja niszczy stos



## Parametry

<i>top</i>	uchwyt stosu
------------	--------------

## 4.6.1.4 push()

```
bool push (
    handle *const top,
    const double d )
```

funkcja wstawia liczbę na szczyt stosu

## Parametry

<i>top</i>	uchwyt stosu
<i>d</i>	liczba wstawiana na stos

## Zwraca

true = powodzenie

## 4.7 Dokumentacja pliku stack.h

```
#include <stdbool.h>
```

## Struktury danych

- struct [stack](#)
- struct [handle](#)

## Definicje typów

- typedef struct [stack](#) [stack](#)
- typedef struct [handle](#) [handle](#)

## Funkcje

- bool [push](#) ([handle](#) \*const top, const double d)
- bool [pop](#) ([handle](#) \*const top, double \*d)
- double [peek](#) ([handle](#) \*const top)
- void [pulverize](#) ([handle](#) \*const top)

## 4.7.1 Dokumentacja definicji typów

### 4.7.1.1 handle

```
typedef struct handle handle
```

uchwyt stosu

### 4.7.1.2 stack

```
typedef struct stack stack
```

element stosu

## 4.7.2 Dokumentacja funkcji

### 4.7.2.1 peek()

```
double peek (  
    handle *const top )
```

funkcja zwraca wartość liczby ze szczytu stosu bez zdejmowania jej

#### Parametry

<i>top</i>	uchwyt stosu
------------	--------------

#### Zwraca

wartość liczby ze szczytu stosu

### 4.7.2.2 pop()

```
bool pop (  
    handle *const top,  
    double * d )
```

funkcja zdejmuję liczbę ze szczytu stosu

## Parametry

<i>top</i>	uchwyt stosu
<i>d</i>	wskaźnik na zmienną, do której zwracana jest liczba zdjęta ze stosu

## Zwraca

true = powodzenie

## 4.7.2.3 pulverize()

```
void pulverize (
    handle *const top )
```

funkcja niszczy stos

## Parametry

<i>top</i>	uchwyt stosu
------------	--------------

## 4.7.2.4 push()

```
bool push (
    handle *const top,
    const double d )
```

funkcja wstawia liczbę na szczyt stosu

## Parametry

<i>top</i>	uchwyt stosu
<i>d</i>	liczba wstawiana na stos

## Zwraca

true = powodzenie



# Indeks

- arg\_char
  - misc.c, 8
- args, 5
  - misc.h, 12
- asrt
  - misc.h, 12
- asrt\_count
  - main.c, 7
  - misc.h, 14
- copy\_path
  - misc.c, 8
  - misc.h, 12
- get\_help
  - misc.c, 9
  - misc.h, 13
- get\_operands
  - operations.c, 15
  - operations.h, 17
- get\_operation
  - operations.c, 15
  - operations.h, 18
- handle, 6
  - stack.h, 22
- is\_number
  - misc.c, 9
  - misc.h, 13
- main.c, 7
  - asrt\_count, 7
- memory\_operation
  - operations.c, 16
  - operations.h, 18
- misc.c, 7
  - arg\_char, 8
  - copy\_path, 8
  - get\_help, 9
  - is\_number, 9
  - new\_stack, 9
  - parse\_args, 9
  - parse\_exp, 10
  - read\_text, 10
- misc.h, 11
  - args, 12
  - asrt, 12
  - asrt\_count, 14
  - copy\_path, 12
  - get\_help, 13
  - is\_number, 13
  - new\_stack, 13
  - parse\_args, 13
  - parse\_exp, 13
  - read\_text, 14
- new\_stack
  - misc.c, 9
  - misc.h, 13
- op\_function\_declr
  - operations.h, 17–19
- operation, 6
  - operations.h, 17
- operations.c, 15
  - get\_operands, 15
  - get\_operation, 15
  - memory\_operation, 16
- operations.h, 16
  - get\_operands, 17
  - get\_operation, 18
  - memory\_operation, 18
  - op\_function\_declr, 17–19
  - operation, 17
- parse\_args
  - misc.c, 9
  - misc.h, 13
- parse\_exp
  - misc.c, 10
  - misc.h, 13
- peek
  - stack.c, 20
  - stack.h, 22
- pop
  - stack.c, 20
  - stack.h, 22
- pulverize
  - stack.c, 20
  - stack.h, 23
- push
  - stack.c, 21
  - stack.h, 23
- read\_text
  - misc.c, 10
  - misc.h, 14
- stack, 6
  - stack.h, 22
- stack.c, 19

- peek, [20](#)
- pop, [20](#)
- pulverize, [20](#)
- push, [21](#)
- stack.h, [21](#)
  - handle, [22](#)
  - peek, [22](#)
  - pop, [22](#)
  - pulverize, [23](#)
  - push, [23](#)
  - stack, [22](#)