Kalkulator ONP

Wygenerowano przez Doxygen 1.8.15

Rozdział 1

Indeks struktur danych

1.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

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

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); \
                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

destination	wskaźnik na pamięć, która będzie alokowana
source	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

ptr	string do sprawdzenia
d	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

exp	przetwarzany string
top	uchwyt stosu
config	konfiguracja programu
f_out	strumień wyjścia programu

Zwraca

true = powodzenie

4.2.2.7 read_text()

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

funkcja

Parametry

config	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 occured. 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

4.3.1 Dokumentacja definicji

4.3.1.1 asrt

```
#define asrt(
              z )
```

Wartość:

```
if (!z) {                                                       \
        ++asrt_count;                                               \
        fprintf(stderr, error_msg);                                 \
        fprintf(stderr, error@ %s, line %d\n
, __FILE__, __LINE__); \
        return 0;                                                   \
    }
```

inicjalizacja w main.c

makro wychwytujące błędy krytyczne

4.3.2 Dokumentacja definicji typów

4.3.2.1 args

```
typedef struct args args
```

argumenty wiersza poleceń

4.3.3 Dokumentacja funkcji

4.3.3.1 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

destination	wskaźnik na pamięć, która będzie alokowana
source	ścieżka dostępu do pliku

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

ptr	string do sprawdzenia
d	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

exp	przetwarzany string
top	uchwyt stosu
config	konfiguracja programu
f_out	strumień wyjścia programu

Zwraca

true = powodzenie

4.3.3.7 read_text()

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

funkcja

Parametry

config	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

top	uchwyt stosu
num_of_operands	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

str	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

top	uchwyt stosu
op	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

name	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

top	uchwyt stosu
num_of_operands	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

str	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

top	uchwyt stosu
op	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

top	uchwyt stosu

Zwraca

wartość liczby ze szczytu stosu

4.6.1.2 pop()

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

funkcja zdejmuje liczbę ze szczytu stosu

Parametry

top	uchwyt stosu
d	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

top	uchwyt stosu

4.6.1.4 push()

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

funkcja wstawia liczbę na szczyt stosu

Parametry

top	uchwyt stosu
d	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

top	uchwyt stosu

Zwraca

wartość liczby ze szczytu stosu

4.7.2.2 pop()

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

funkcja zdejmuje liczbę ze szczytu stosu

Parametry

top	uchwyt stosu
d	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

top	uchwyt stosu

4.7.2.4 push()

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

funkcja wstawia liczbę na szczyt stosu

Parametry

top	uchwyt stosu
d	liczba wstawiana na stos

Zwraca

true = powodzenie

Indeks