

Calculator

Generated by Doxygen 1.9.1

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 calculator_front.c File Reference	3
2.1.1 Macro Definition Documentation	3
2.1.1.1 ASCII_TO_NUM	4
2.1.1.2 IS_NUM	4
2.1.2 Function Documentation	4
2.1.2.1 clear_screen()	4
2.1.2.2 flush_stdin()	4
2.1.2.3 parse_input()	4
2.1.2.4 print_calculator_tips()	5
2.1.2.5 print_operation_result()	5
2.1.2.6 print_operators()	5
2.2 calculator_front.h File Reference	6
2.2.1 Macro Definition Documentation	6
2.2.1.1 OPERATOR_NOT_FOUND	6
2.2.2 Function Documentation	6
2.2.2.1 clear_screen()	6
2.2.2.2 flush_stdin()	6
2.2.2.3 parse_input()	7
2.2.2.4 print_calculator_tips()	7
2.2.2.5 print_operation_result()	7
2.2.2.6 print_operators()	8
2.3 main.c File Reference	8
2.3.1 Macro Definition Documentation	9
2.3.1.1 ACOS_CHAR	9
2.3.1.2 ASIN_CHAR	9
2.3.1.3 ATAN_CHAR	9
2.3.1.4 COS_CHAR	10
2.3.1.5 DIV_CHAR	10
2.3.1.6 EXIT_CHAR	10
2.3.1.7 FACT_CHAR	10
2.3.1.8 HELP_CHAR	10
2.3.1.9 MAX_OPERATORS	10
2.3.1.10 POW_CHAR	10
2.3.1.11 PROD_CHAR	11
2.3.1.12 SIN_CHAR	11
2.3.1.13 SUBS_CHAR	11
2.3.1.14 SUM_CHAR	11
2.3.1.15 TAN_CHAR	11

2.3.2 Function Documentation	11
2.3.2.1 add_operation()	11
2.3.2.2 cos_wrapper()	12
2.3.2.3 factorial_wrapper()	12
2.3.2.4 find_operator()	12
2.3.2.5 integer_power_wrapper()	13
2.3.2.6 main()	13
2.3.2.7 sin_wrapper()	13
2.3.3 Variable Documentation	14
2.3.3.1 actions	14
2.3.3.2 operators	14
2.4 operations.c File Reference	14
2.4.1 Macro Definition Documentation	15
2.4.1.1 TAYLOR_TERMS	15
2.4.2 Function Documentation	15
2.4.2.1 cos()	15
2.4.2.2 division()	15
2.4.2.3 factorial()	16
2.4.2.4 integer_power()	16
2.4.2.5 product()	16
2.4.2.6 sin()	17
2.4.2.7 subtraction()	17
2.4.2.8 sum()	18
2.5 operations.h File Reference	18
2.5.1 Macro Definition Documentation	18
2.5.1.1 PI	19
2.5.2 Function Documentation	19
2.5.2.1 cos()	19
2.5.2.2 division()	19
2.5.2.3 factorial()	19
2.5.2.4 integer_power()	21
2.5.2.5 product()	21
2.5.2.6 sin()	22
2.5.2.7 subtraction()	22
2.5.2.8 sum()	22
Index	25

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

calculator_front.c	3
calculator_front.h	6
main.c	8
operations.c	14
operations.h	18

Chapter 2

File Documentation

2.1 calculator_front.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "calculator_front.h"
```

Macros

- #define `IS_NUM(c)` (((c) >= '0') && ((c) <= '9'))
- #define `ASCII_TO_NUM(c)` ((c) - '0')

Functions

- int `parse_input` (double *op1, double *op2, char *operator)
Reads two operands and an operator from stdin.
- void `print_calculator_tips` (void)
Prints tips for calculator use.
- void `print_operators` (char operators[], int op_num)
Prints available operators.
- void `print_operation_result` (double op1, double op2, char operation, double result)
Prints the expression to resolve and the result.
- void `clear_screen` (void)
Clears terminal in an OS-specific basis.
- void `flush_stdin` (char limit)
Flushes the stdin buffer. Takes characters from stdin until limit is found.

2.1.1 Macro Definition Documentation

2.1.1.1 ASCII_TO_NUM

```
#define ASCII_TO_NUM(  
    c ) ((c) - '0')
```

2.1.1.2 IS_NUM

```
#define IS_NUM(  
    c ) (((c) >= '0') && ((c) <= '9'))
```

2.1.2 Function Documentation

2.1.2.1 clear_screen()

```
void clear_screen (  
    void )
```

Clears terminal in an OS-specific basis.

2.1.2.2 flush_stdin()

```
void flush_stdin (  
    char limit )
```

Flushes the stdin buffer. Takes characters from stdin until limit is found.

Parameters

<i>limit</i>	Character that indicates where to stop flushing
--------------	---

2.1.2.3 parse_input()

```
int parse_input (  
    double * op1,  
    double * op2,  
    char * operator )
```

Reads two operands and an operator from stdin.

Parameters

<i>op1</i>	Pointer to the variable that stores the first operand
<i>op2</i>	Pointer to the variable that stores the second operand
<i>operator</i>	Pointer to the variable that stores the operator char representation

Returns

0 if there was an error, 1 otherwise.

2.1.2.4 print_calculator_tips()

```
void print_calculator_tips (  
    void )
```

Prints tips for calculator use.

2.1.2.5 print_operation_result()

```
void print_operation_result (  
    double op1,  
    double op2,  
    char operation,  
    double result )
```

Prints the expression to resolve and the result.

Parameters

<i>op1</i>	Left operand
<i>op2</i>	Right operand
<i>operation</i>	Char representing operation
<i>result</i>	The operation result

2.1.2.6 print_operators()

```
void print_operators (  
    char operators[],  
    int op_num )
```

Prints available operators.

Parameters

<i>operators</i>	Array containing every operator-representing char
<i>op_num</i>	Number of operators

2.2 calculator_front.h File Reference

Macros

- `#define OPERATOR_NOT_FOUND -1`

Functions

- `int parse_input (double *op1, double *op2, char *operator)`
Reads two operands and an operator from stdin.
- `void print_calculator_tips (void)`
Prints tips for calculator use.
- `void print_operators (char operators[], int op_num)`
Prints available operators.
- `void print_operation_result (double op1, double op2, char operation, double result)`
Prints the expression to resolve and the result.
- `void clear_screen (void)`
Clears terminal in an OS-specific basis.
- `void flush_stdin (char limit)`
Flushes the stdin buffer. Takes characters from stdin until limit is found.

2.2.1 Macro Definition Documentation

2.2.1.1 OPERATOR_NOT_FOUND

```
#define OPERATOR_NOT_FOUND -1
```

2.2.2 Function Documentation

2.2.2.1 clear_screen()

```
void clear_screen (
    void )
```

Clears terminal in an OS-specific basis.

2.2.2.2 flush_stdin()

```
void flush_stdin (
    char limit )
```

Flushes the stdin buffer. Takes characters from stdin until limit is found.

Parameters

<i>limit</i>	Character that indicates where to stop flushing
--------------	---

2.2.2.3 parse_input()

```
int parse_input (
    double * op1,
    double * op2,
    char * operator )
```

Reads two operands and an operator from stdin.

Parameters

<i>op1</i>	Pointer to the variable that stores the first operand
<i>op2</i>	Pointer to the variable that stores the second operand
<i>operator</i>	Pointer to the variable that stores the operator char representation

Returns

0 if there was an error, 1 otherwise.

2.2.2.4 print_calculator_tips()

```
void print_calculator_tips (
    void )
```

Prints tips for calculator use.

2.2.2.5 print_operation_result()

```
void print_operation_result (
    double op1,
    double op2,
    char operation,
    double result )
```

Prints the expression to resolve and the result.

Parameters

<i>op1</i>	Left operand
<i>op2</i>	Right operand
<i>operation</i>	Char representing operation
<i>result</i>	The operation result

2.2.2.6 print_operators()

```
void print_operators (
    char operators[],
    int op_num )
```

Prints available operators.

Parameters

<i>operators</i>	Array containing every operator-representing char
<i>op_num</i>	Number of operators

2.3 main.c File Reference

```
#include <stdio.h>
#include "operations.h"
#include "calculator_front.h"
```

Macros

- `#define MAX_OPERATORS 8`
Number of operators.
- `#define EXIT_CHAR 'q'`
- `#define HELP_CHAR '?'`
- `#define SUM_CHAR '+'`
- `#define SUBS_CHAR '-'`
- `#define DIV_CHAR '/'`
- `#define PROD_CHAR '*'`
- `#define POW_CHAR '^'`
- `#define FACT_CHAR '!'`
- `#define SIN_CHAR 's'`
- `#define COS_CHAR 'c'`
- `#define TAN_CHAR 't'`
unused
- `#define ASIN_CHAR 'a'`
unused
- `#define ACOS_CHAR 'b'`
unused
- `#define ATAN_CHAR 'd'`
unused

Functions

- int [find_operator](#) (char c)
Finds the index of the operator represented by a char.
- int [add_operation](#) (char operator_char, double(*callback)(double, double))
Adds an operator to the array.
- double [integer_power_wrapper](#) (double a, double b)
Wraps power function due to formatting.
- double [factorial_wrapper](#) (double a, double b)
Wraps factorial function due to formatting.
- double [sin_wrapper](#) (double a, double b)
Wraps sin function due to formatting.
- double [cos_wrapper](#) (double a, double b)
Wraps cos function due to formatting.
- int [main](#) (void)

Variables

- double(* [actions](#) [[MAX_OPERATORS](#)])(double, double)
- char [operators](#) [[MAX_OPERATORS](#)]

2.3.1 Macro Definition Documentation

2.3.1.1 ACOS_CHAR

```
#define ACOS_CHAR 'b'
```

unused

2.3.1.2 ASIN_CHAR

```
#define ASIN_CHAR 'a'
```

unused

2.3.1.3 ATAN_CHAR

```
#define ATAN_CHAR 'd'
```

unused

2.3.1.4 COS_CHAR

```
#define COS_CHAR 'c'
```

2.3.1.5 DIV_CHAR

```
#define DIV_CHAR '/'
```

2.3.1.6 EXIT_CHAR

```
#define EXIT_CHAR 'q'
```

2.3.1.7 FACT_CHAR

```
#define FACT_CHAR '!'
```

2.3.1.8 HELP_CHAR

```
#define HELP_CHAR '?'
```

2.3.1.9 MAX_OPERATORS

```
#define MAX_OPERATORS 8
```

Number of operators.

2.3.1.10 POW_CHAR

```
#define POW_CHAR '^'
```

2.3.1.11 PROD_CHAR

```
#define PROD_CHAR '*'
```

2.3.1.12 SIN_CHAR

```
#define SIN_CHAR 's'
```

2.3.1.13 SUBS_CHAR

```
#define SUBS_CHAR '-'
```

2.3.1.14 SUM_CHAR

```
#define SUM_CHAR '+'
```

2.3.1.15 TAN_CHAR

```
#define TAN_CHAR 't'
```

unused

2.3.2 Function Documentation

2.3.2.1 add_operation()

```
int add_operation (  
    char operator_char,  
    double(*) (double, double) callback )
```

Adds an operator to the array.

Parameters

<i>operator_char</i>	The char represeting the operation
<i>callback</i>	The function to associate to the operator

Returns

1 if succesful, 0 otherwise

2.3.2.2 cos_wrapper()

```
double cos_wrapper (
    double a,
    double b )
```

Wraps cos function due to formatting.

Parameters

<i>a</i>	Left operand (to calc cos of)
<i>b</i>	Right operand (nothing)

Returns

Result of cos(a)

2.3.2.3 factorial_wrapper()

```
double factorial_wrapper (
    double a,
    double b )
```

Wraps factorial function due to formatting.

Parameters

<i>a</i>	Left operand (to take the factorial of)
<i>b</i>	Right operand (nothing)

Returns

Result of a!

2.3.2.4 find_operator()

```
int find_operator (
    char c )
```

Finds the index of the operator represented by a char.

Parameters

<i>c</i>	The identifier of the operator to search for
----------	--

Returns

The operation index

2.3.2.5 integer_power_wrapper()

```
double integer_power_wrapper (
    double a,
    double b )
```

Wraps power function due to formatting.

Parameters

<i>a</i>	Left operand (base)
<i>b</i>	Right operand (exponent)

Returns

Result of a^b

2.3.2.6 main()

```
int main (
    void )
```

2.3.2.7 sin_wrapper()

```
double sin_wrapper (
    double a,
    double b )
```

Wraps sin function due to formatting.

Parameters

<i>a</i>	Left operand (to calc sin of)
<i>b</i>	Right operand (nothing)

Returns

Result of $\sin(a)$

2.3.3 Variable Documentation

2.3.3.1 actions

```
double(* actions[MAX_OPERATORS])(double, double) (  
    double ,  
    double )
```

2.3.3.2 operators

```
char operators[MAX_OPERATORS]
```

2.4 operations.c File Reference

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

Macros

- `#define TAYLOR_TERMS 30`
The number of terms to approximate to when using Taylor expansion.

Functions

- double `sum` (double a, double b)
Adds two numbers together.
- double `subtraction` (double a, double b)
Subtracts two numbers.
- double `division` (double a, double b)
Divides two numbers.
- double `product` (double a, double b)
Calculates the product of two numbers.
- double `integer_power` (double base, int exp)
Calculates base to the integer power exp.
- double `sin` (double a)
Calculates the sine of a.
- double `cos` (double a)
Calculates the cosine of a.
- double `factorial` (unsigned int a)
Calculates the factorial of a number.

2.4.1 Macro Definition Documentation

2.4.1.1 TAYLOR_TERMS

```
#define TAYLOR_TERMS 30
```

The number of terms to approximate to when using Taylor expansion.

2.4.2 Function Documentation

2.4.2.1 cos()

```
double cos (  
    double a )
```

Calculates the cosine of a.

Parameters

<i>b</i>	Number to take the cosine of.
----------	-------------------------------

Returns

cos(a).

2.4.2.2 division()

```
double division (  
    double a,  
    double b )
```

Divides two numbers.

Parameters

<i>a</i>	number to be divided.
<i>b</i>	number to divide by.

Returns

the result of a divided by b .

2.4.2.3 factorial()

```
double factorial (
    unsigned int b )
```

Calculates the factorial of a number.

Parameters

<i>a</i>	Number to take the factorial of.
----------	----------------------------------

Returns

$(a)!$

2.4.2.4 integer_power()

```
double integer_power (
    double a,
    int b )
```

Calculates base to the integer power exp .

Parameters

<i>base</i>	Base of the power.
<i>exp</i>	Exponent of the power.

Returns

the result of a to the power b .

2.4.2.5 product()

```
double product (
    double a,
    double b )
```

Calculates the product of two numbers.

Parameters

<i>a</i>	a factor.
<i>b</i>	another factor.

Returns

the result of a times b.

2.4.2.6 sin()

```
double sin (  
    double a )
```

Calculates the sine of a.

Parameters

<i>a</i>	Number to take the sine of.
----------	-----------------------------

Returns

sin(a).

2.4.2.7 subtraction()

```
double subtraction (  
    double a,  
    double b )
```

Subtracts two numbers.

Parameters

<i>a</i>	number to subtract from.
<i>b</i>	number to subtract.

Returns

result of a minus b.

2.4.2.8 sum()

```
double sum (  
    double a,  
    double b )
```

Adds two numbers together.

Parameters

<i>a</i>	first number to be added.
<i>b</i>	second number to be added.

Returns

the sum of both numbers a and b.

2.5 operations.h File Reference

Macros

- #define [PI](#) 3.141592653589793238

Functions

- double [sum](#) (double a, double b)
Adds two numbers together.
- double [subtraction](#) (double a, double b)
Subtracts two numbers.
- double [division](#) (double a, double b)
Divides two numbers.
- double [product](#) (double a, double b)
Calculates the product of two numbers.
- double [integer_power](#) (double a, int b)
Calculates base to the integer power exp.
- double [sin](#) (double a)
Calculates the sine of a.
- double [cos](#) (double a)
Calculates the cosine of a.
- double [factorial](#) (unsigned int b)
Calculates the factorial of a number.

2.5.1 Macro Definition Documentation

2.5.1.1 PI

```
#define PI 3.141592653589793238
```

2.5.2 Function Documentation

2.5.2.1 cos()

```
double cos (  
    double a )
```

Calculates the cosine of a.

Parameters

<i>b</i>	Number to take the cosine of.
----------	-------------------------------

Returns

cos(a).

2.5.2.2 division()

```
double division (  
    double a,  
    double b )
```

Divides two numbers.

Parameters

<i>a</i>	number to be divided.
<i>b</i>	number to divide by.

Returns

the result of a divided by b.

2.5.2.3 factorial()

```
double factorial (  
    unsigned int b )
```

Calculates the factorial of a number.

Parameters

<i>a</i>	Number to take the factorial of.
----------	----------------------------------

Returns

(a)!

2.5.2.4 integer_power()

```
double integer_power (
    double a,
    int b )
```

Calculates base to the integer power exp.

Parameters

<i>base</i>	Base of the power.
<i>exp</i>	Exponent of the power.

Returns

the result of a to the power b.

2.5.2.5 product()

```
double product (
    double a,
    double b )
```

Calculates the product of two numbers.

Parameters

<i>a</i>	a factor.
<i>b</i>	another factor.

Returns

the result of a times b.

2.5.2.6 sin()

```
double sin (  
    double a )
```

Calculates the sine of a.

Parameters

<i>a</i>	Number to take the sine of.
----------	-----------------------------

Returns

sin(a).

2.5.2.7 subtraction()

```
double subtraction (  
    double a,  
    double b )
```

Subtracts two numbers.

Parameters

<i>a</i>	number to subtract from.
<i>b</i>	number to subtract.

Returns

result of a minus b.

2.5.2.8 sum()

```
double sum (  
    double a,  
    double b )
```

Adds two numbers together.

Parameters

<i>a</i>	first number to be added.
<i>b</i>	second number to be added.

Returns

the sum of both numbers a and b.

Index

- ACOS_CHAR
 - main.c, [9](#)
- actions
 - main.c, [14](#)
- add_operation
 - main.c, [11](#)
- ASCII_TO_NUM
 - calculator_front.c, [3](#)
- ASIN_CHAR
 - main.c, [9](#)
- ATAN_CHAR
 - main.c, [9](#)
- calculator_front.c, [3](#)
 - ASCII_TO_NUM, [3](#)
 - clear_screen, [4](#)
 - flush_stdin, [4](#)
 - IS_NUM, [4](#)
 - parse_input, [4](#)
 - print_calculator_tips, [5](#)
 - print_operation_result, [5](#)
 - print_operators, [5](#)
- calculator_front.h, [6](#)
 - clear_screen, [6](#)
 - flush_stdin, [6](#)
 - OPERATOR_NOT_FOUND, [6](#)
 - parse_input, [7](#)
 - print_calculator_tips, [7](#)
 - print_operation_result, [7](#)
 - print_operators, [8](#)
- clear_screen
 - calculator_front.c, [4](#)
 - calculator_front.h, [6](#)
- cos
 - operations.c, [15](#)
 - operations.h, [19](#)
- COS_CHAR
 - main.c, [9](#)
- cos_wrapper
 - main.c, [12](#)
- DIV_CHAR
 - main.c, [10](#)
- division
 - operations.c, [15](#)
 - operations.h, [19](#)
- EXIT_CHAR
 - main.c, [10](#)
- FACT_CHAR
 - main.c, [10](#)
- factorial
 - operations.c, [16](#)
 - operations.h, [19](#)
- factorial_wrapper
 - main.c, [12](#)
- find_operator
 - main.c, [12](#)
- flush_stdin
 - calculator_front.c, [4](#)
 - calculator_front.h, [6](#)
- HELP_CHAR
 - main.c, [10](#)
- integer_power
 - operations.c, [16](#)
 - operations.h, [21](#)
- integer_power_wrapper
 - main.c, [13](#)
- IS_NUM
 - calculator_front.c, [4](#)
- main
 - main.c, [13](#)
- main.c, [8](#)
 - ACOS_CHAR, [9](#)
 - actions, [14](#)
 - add_operation, [11](#)
 - ASIN_CHAR, [9](#)
 - ATAN_CHAR, [9](#)
 - COS_CHAR, [9](#)
 - cos_wrapper, [12](#)
 - DIV_CHAR, [10](#)
 - EXIT_CHAR, [10](#)
 - FACT_CHAR, [10](#)
 - factorial_wrapper, [12](#)
 - find_operator, [12](#)
 - HELP_CHAR, [10](#)
 - integer_power_wrapper, [13](#)
 - main, [13](#)
 - MAX_OPERATORS, [10](#)
 - operators, [14](#)
 - POW_CHAR, [10](#)
 - PROD_CHAR, [10](#)
 - SIN_CHAR, [11](#)
 - sin_wrapper, [13](#)
 - SUBS_CHAR, [11](#)
 - SUM_CHAR, [11](#)
 - TAN_CHAR, [11](#)

- MAX_OPERATORS
 - main.c, [10](#)
- operations.c, [14](#)
 - cos, [15](#)
 - division, [15](#)
 - factorial, [16](#)
 - integer_power, [16](#)
 - product, [16](#)
 - sin, [17](#)
 - subtraction, [17](#)
 - sum, [17](#)
 - TAYLOR_TERMS, [15](#)
- operations.h, [18](#)
 - cos, [19](#)
 - division, [19](#)
 - factorial, [19](#)
 - integer_power, [21](#)
 - PI, [18](#)
 - product, [21](#)
 - sin, [21](#)
 - subtraction, [22](#)
 - sum, [22](#)
- OPERATOR_NOT_FOUND
 - calculator_front.h, [6](#)
- operators
 - main.c, [14](#)
- parse_input
 - calculator_front.c, [4](#)
 - calculator_front.h, [7](#)
- PI
 - operations.h, [18](#)
- POW_CHAR
 - main.c, [10](#)
- print_calculator_tips
 - calculator_front.c, [5](#)
 - calculator_front.h, [7](#)
- print_operation_result
 - calculator_front.c, [5](#)
 - calculator_front.h, [7](#)
- print_operators
 - calculator_front.c, [5](#)
 - calculator_front.h, [8](#)
- PROD_CHAR
 - main.c, [10](#)
- product
 - operations.c, [16](#)
 - operations.h, [21](#)
- sin
 - operations.c, [17](#)
 - operations.h, [21](#)
- SIN_CHAR
 - main.c, [11](#)
- sin_wrapper
 - main.c, [13](#)
- SUBS_CHAR
 - main.c, [11](#)
- subtraction
 - operations.c, [17](#)
 - operations.h, [22](#)
- sum
 - operations.c, [17](#)
 - operations.h, [22](#)
- SUM_CHAR
 - main.c, [11](#)
- TAN_CHAR
 - main.c, [11](#)
- TAYLOR_TERMS
 - operations.c, [15](#)