Calculator

1.0

Generated by Doxygen 1.8.15

Chapter 1

PP Calculator

Pied Piper Calculator was implemented in a small team withtin the ISV course at Brno University of Technology

Figure 1.1 PRO mode

There are two modes of working. Basic mode provides you with basic arithmetic operations, factorial, modulo and power. With PRO mode you can handle more complicated calculations. You will be able to see the progress of calculations, use trigonometric functions and brackets. The calculator has a minimalistic design in blue hues that will be pleasant to the eye. There is a special button that allows you to switch between these two modes.

Figure 1.2 PRO mode

Author

Mihola David
Foltyn Lukas
Sokolovskii Vladislav

Copyright

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

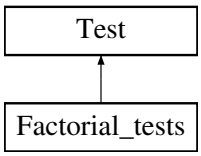
Class Documentation

5.1 Factorial_tests Class Reference

class that tests the factorial calculation

`#include <tests.h>`

Inheritance diagram for Factorial_tests:



Protected Member Functions

- bool fact_test (long unsigned num, long unsigned expected_result, bool fail_expected)
  function that compares calculation of factorial with expected result

5.1.1 Detailed Description

class that tests the factorial calculation

Definition at line 128 of file tests.h.

5.1.2 Member Function Documentation

5.1.2.1 fact_test()

```
bool Factorial_tests::fact_test (
            long unsigned num,
            long unsigned expected_result,
            bool fail_expected )  [protected]
```

function that compares calculation of factorial with expected result

Precondition

expected result has to be accurate to 5 decimal digits

Parameters

num	number from which factorial is calculated
expected_result	number, that is expected to be correct result of calculation
fail_expected	boolean, set to true if fail of calc is expected (too large number), else set to to false

Returns

true on success - expected_result matches the result of calculator or calculation fails while fail_expected is set to true, otherwise false is returned

Definition at line 82 of file test_mock_stub.cpp.

```
83 {
84     long unsigned fact_result;
85     try
86     {
87         fact_result =  factorial(num);
88         if (fact_result == expected_result)
89             return 1;
90         else
91         {
92             fprintf(stderr, "%lu\n", fact_result);
93             return 0;
94         }
95     }
96     catch(const std::exception& e)
97     {
98         return fail_expected;
99     }
100 }
```

The documentation for this class was generated from the following files:

- include/tests.h
- source/test_mock_stub.cpp

5.2 GUI Class Reference

This class includes all the widgets and functions that are needed for initialization of graphics.

```
#include <graphics.h>
```

Static Public Member Functions

- static int init_graphics (int argc, char **argv)

  This functions starts the main loop of the window, connects the graphics to the grap.glade file, handles callbacks and sets the CSS styles for the calculator. Basically the main GUI function. Retrieves all the information about eh widgets from the frab.glade file (builder), checks validity of the widgets and initialized all buttons. Handles the callbacks of buttons and 'window destoyer'. In CSS sections sets fonts and colors for all widgets.

Static Protected Member Functions

- static void on_basic_mode_active ()

  This function is responsible for switching to the basic mode after choosing the option in the menu bar View –> Basic mode. Checks graphics_mode variable and switch the mode if it is needed, otherwise does nothing.
- static void on_pro_mode_active ()

  This function is responsible for switching to the PRO mode after choosing the option in the menu bar View –> Professional mode. Checks graphics_mode variable and switch the mode if it is needed, otherwise does nothing.
- static void on_help_active ()

  This function open the separate window with help message if the user chose option in the menu bar Help. Starts separate gtk_main_loop.
- static void on_button_clicked (GtkButton *button, gpointer data)

  This function reacts to a buttons clicks, in both modes.
- static void on_switch_state_set ()

  This function resizes the window, moves container with the basic buttons and makes the advanced functions visible. When gets a callback from the switcher button switch the mode – moves container with basic buttons and makes advanced menu visible/invisible. Also, switching from basic to PRO mode sets graphics_mode variable to true (necessary for connecting to logic).

Static Protected Attributes

- static GtkWidget * window = NULL
- static GtkWidget * fixed = NULL
- static GtkWidget * main_menu = NULL
- static GtkWidget * advanced_menu = NULL
- static GtkWidget * button [NUM_OF_BUTTONS] = {0,}
- static GtkWidget * label = NULL
- static GtkWidget * label_advanced = NULL
- static GtkWidget * mode = NULL
- static GtkBuilder * builder = NULL
- static GtkWidget * switcher_button = NULL
- static GdkScreen * screen = NULL
- static GtkCssProvider * provider = NULL
- static GdkDisplay * display = NULL
- static GtkWidget * calculator = NULL
- static GtkWidget * view = NULL
- static GtkWidget * quit = NULL
- static GtkWidget * help = NULL
- static GtkWidget * basic_mode = NULL
- static GtkWidget * pro_mode = NULL
- static Logic logic
- static int index_arr [NUM_OF_BUTTONS]
- static const char * button_names [NUM_OF_BUTTONS]
- static char button_text [NUM_OF_BUTTONS]
- static std::string label_text
- static std::string label_text_advanced
- static bool graphics_mode

5.2.1 Detailed Description

This class includes all the widgets and functions that are needed for initialization of graphics.

Definition at line 28 of file graphics.h.

5.2.2 Member Function Documentation

5.2.2.1 init_graphics()

```
int GUI::init_graphics (
            int argc,
            char ** argv )  [static]
```

This functions starts the main loop of the window, connects the graphics to the grap.glade file, handles callbacks and sets the CSS styles for the calculator. Basically the main GUI function. Retrieves all the information about eh widgets from the frab.glade file (builder), checks validity of the widgets and initialized all buttons. Handles the callbacks of buttons and 'window destoyer'. In CSS sections sets fonts and colors for all widgets.

Parameters

in	*argc*	Number of arguments from the comand line.
in	*argv*	Vector of given arguments.

Returns

Returns zero value if work of the window was finihed successfully, otherwise returns non-zero value.

Definition at line 70 of file graphics.cpp.

```
71 {
72     //initiates GTK
73     gtk_init(&argc, &argv);
74 
75     //caling builder, which will toad graphics data from specified file
76     //TODO expetions
77     //checking the location of the file contaning graphic layout
78     if (access("/usr/local/bin/grap.glade", F_OK ) == 0)
79     {
80     //for installed version
81         builder = gtk_builder_new_from_file("/usr/local/bin/grap.glade");
82     }
83     else if (access("grap.glade", F_OK ) == 0)
84     {
85     //for version running prom the source directory
86         builder = gtk_builder_new_from_file("grap.glade");
87     }
88     else
89     {
90     //when the graphic file could not be locate
91         std::cerr « "Graphics initialization failed" « std::endl;
92         std::cerr « "File with graphic layout coul not be located" « std::endl;
93         return 1;
94     }
95 
96     //inicialization of the main window
97     window = GTK_WIDGET(gtk_builder_get_object(builder, "window"));
98 
99     //inicialization of other parts of the graphics
100     pro_mode = GTK_WIDGET(gtk_builder_get_object(builder, "pro_mode"));
101     basic_mode = GTK_WIDGET(gtk_builder_get_object(builder, "basic_mode"));
102     help = GTK_WIDGET(gtk_builder_get_object(builder, "help"));
103     view = GTK_WIDGET(gtk_builder_get_object(builder, "view"));
104     calculator = GTK_WIDGET(gtk_builder_get_object(builder, "calculator"));
105     quit = GTK_WIDGET(gtk_builder_get_object(builder, "quit"));
106     fixed = GTK_WIDGET(gtk_builder_get_object(builder, "fixed"));
107     main_menu = GTK_WIDGET(gtk_builder_get_object(builder, "main_menu"));
108     advanced_menu = GTK_WIDGET(gtk_builder_get_object(builder, "advanced_menu"));
109     label = GTK_WIDGET(gtk_builder_get_object(builder, "label1"));
110     label_advanced = GTK_WIDGET(gtk_builder_get_object(builder, "label3"));
111     mode = GTK_WIDGET(gtk_builder_get_object(builder, "label2"));
112     switcher_button = GTK_WIDGET(gtk_builder_get_object(builder, "switcher_button"));
113 
```

```
114     if (window == NULL || main_menu == NULL || advanced_menu == NULL  || label == NULL || basic_mode ==
        NULL || view == NULL ||
115         label_advanced == NULL || mode == NULL || switcher_button == NULL || pro_mode == NULL ||
        calculator == NULL ||
116         quit == NULL || help == NULL || fixed == NULL || switcher_button == NULL )
117     {
118         std::cerr « "Graphics initialization failed\n" « std::endl;
119         return 1;
120     }
121
122     //add signal to close the app when the X button on the top of the window is pressed
123     g_signal_connect(window, "destroy", G_CALLBACK(gtk_main_quit), NULL);
124     g_signal_connect(quit, "activate", G_CALLBACK(gtk_main_quit), NULL);
125
126     //signal for the switcher when it is turned on/off
127
128     g_signal_connect(GTK_WIDGET (switcher_button), "clicked", G_CALLBACK (on_switch_state_set), NULL);
129
130     g_signal_connect(GTK_WIDGET (help), "activate", G_CALLBACK(on_help_active), NULL);
131
132     g_signal_connect(GTK_WIDGET (pro_mode), "activate", G_CALLBACK(on_pro_mode_active), NULL);
133
134     g_signal_connect(GTK_WIDGET (basic_mode), "activate", G_CALLBACK(on_basic_mode_active), NULL);
135
136     for (int i = 0; i < NUM_OF_BUTTONS; i++)
137     {
138         //inicialization of all the buttons (appart from advanced and basic)
139         button[i] = GTK_WIDGET(gtk_builder_get_object(builder, button_names[i]));
140         if (button[i] == NULL)
141         {
142             std::cerr « "Graphics initialization failed\n" « std::endl;
143             return 1;
144         }
145
146         //signal for each button when pressed
147         //passing data with poineter to the index_arr
148         g_signal_connect(button[i], "clicked", G_CALLBACK(on_button_clicked), &index_arr[i]);
149         //naming all the buttons for css styling
150         gtk_widget_set_name(button[i], button_names[i]);
151     }
152     //advanced menu is hidden by default
153     gtk_widget_hide(advanced_menu);
154
155 //------------------------------------ CSS styles ------------------------------------//
156     display = gdk_display_get_default();
157     if (display == NULL)
158     {
159         std::cerr « "Graphics initialization failed\n" « std::endl;
160         return 1;
161     }
162     screen = gdk_display_get_default_screen (display);
163     provider = gtk_css_provider_new ();
164     if (screen == NULL || provider == NULL)
165     {
166         std::cerr « "Graphics initialization failed\n" « std::endl;
167         return 1;
168     }
169
170     gtk_widget_set_name(view, "view");
171     gtk_widget_set_name(help, "help");
172     gtk_widget_set_name(calculator, "calculator");
173     gtk_widget_set_name(mode, "mode");
174     gtk_widget_set_name(switcher_button, "switcher");
175     //css styling of all the widgets and text
176     gtk_css_provider_load_from_data(GTK_CSS_PROVIDER(provider), "#button0, #button1, #button2, #button3,
        #button4,\
177                                     #button5, #button6, #button7, #button8, #button9, #comma_button{\
178                                     background: #E5E4E2; color:black; font-family: Helvetica Neue;
        font-size: 18px; font-weight: 200;\
179                                     }\
180                                     #equal_button, #add_button, #subb_button, #mul_button,
        #fact_button, #pow_button, #sign_button, \
181                                     #div_button, #mod_button {background: #7494b8; color: white;
        font-family: Helvetica Neue; font-size: 18px; font-weight: 300;\
182                                     }\
183                                     #OB_button, #CB_button, #bck_button, #clear_button, \
184                                     #sin_button, #cos_button, #tan_button{\
185                                     background: #8394A1; color: #ffffff; font-family: Helvetica Neue;
        font-size: 18px; font-weight: 200;\
186                                     }\
187                                     #switcher{\
188                                     font-family: Helvetica Neue;\
189                                     background: #083A4A;\
190                                     color: #ffffff;\
191                                     }\
192                                     #calculator, #view, #help{\
193                                     font-family: Helvetica Neue;\
```

```
194                                     font-weight: 400;\
195                                     }\
196                                     ", -1 , NULL);
197
198     gtk_style_context_add_provider_for_screen (screen, GTK_STYLE_PROVIDER(provider),
        GTK_STYLE_PROVIDER_PRIORITY_USER);
199 //---------------------------------------------------------------------------//
200
201     //windows shows on the screen
202     gtk_widget_show(window);
203     //program is held here waiting for new actions (user input)
204     gtk_main();
205
206     return 0;
207 }
```

5.2.2.2 on_basic_mode_active()

```
void GUI::on_basic_mode_active ( )  [static], [protected]
```

This function is responsible for switching to the basic mode after choosing the option in the menu bar View –> Basic mode. Checks graphics_mode variable and switch the mode if it is needed, otherwise does nothing.

Definition at line 219 of file graphics.cpp.

```
220 {
221     if(!graphics_mode)
222         return;
223
224     on_switch_state_set();
225 }
```

5.2.2.3 on_button_clicked()

```
void GUI::on_button_clicked (
            GtkButton * button,
            gpointer data )  [static], [protected]
```

This function reacts to a buttons clicks, in both modes.

Parameters

button	Pointer to the button that was clicked.
data	Void pointer to a data sent to the function with each button click.

Definition at line 294 of file graphics.cpp.

```
295 {
296     int index = *(int *)data;
297
298     if (graphics_mode) //for the advanced mode
299     {
300         logic.real_time_calculation(button_text[index], label_text, label_text_advanced);
301
302         gtk_label_set_text(GTK_LABEL(label), label_text.c_str());
303         gtk_label_set_text(GTK_LABEL(label_advanced), label_text_advanced.c_str());
304
305     }
306     else // for the basic mode
307     {
308         if (button_text[index] == 'C')
309         {
310             logic.reset_equation_string_control();
```

```
311             label_text.clear();
312
313             label_text.push_back('0');
314             gtk_label_set_text(GTK_LABEL(label), label_text.c_str());
315
316             logic.erase_equation();
317         }
318         else if (button_text[index] != '=')
319         {
320             //adding the character to the string, that is going to be diplayed in the window
321             logic.equation_string_control(label_text, button_text[index]);
322
323             //displaying the changed text on the label
324             gtk_label_set_text(GTK_LABEL(label), label_text.c_str());
325         }
326         else
327         {
328             try
329             {
330                 logic.calculate(label_text, label_text);
331                 gtk_label_set_text(GTK_LABEL(label), label_text.c_str());
332             }
333             catch(const std::runtime_error &e)
334             {
335                 gtk_label_set_text(GTK_LABEL(label), e.what());
336                 logic.erase_equation();
337             }
338
339
340             //prepares variables for next calculation
341             logic.reset_equation_string_control();
342         }
343     }
344
345     (void)button; //dummy for the compiler
346 }
```

5.2.2.4 on_help_active()

```
void GUI::on_help_active ( )  [static], [protected]
```

This function open the separate window with help message if the user chose option in the menu bar Help. Starts separate gtk_main_loop.

Definition at line 228 of file graphics.cpp.

```
229 {
230     GtkWidget *p_window;
231     GtkWidget *p_v_box;
232     GtkWidget *helpmsg;
233
234     p_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
235     gtk_window_set_title(GTK_WINDOW(p_window), "Help");
236     gtk_window_set_default_size(GTK_WINDOW(p_window), 320, 150);
237
238     g_signal_connect(p_window, "destroy", G_CALLBACK(gtk_main_quit), NULL);
239
240     p_v_box = gtk_box_new(GTK_ORIENTATION_VERTICAL, 0);
241     gtk_container_add(GTK_CONTAINER(p_window), p_v_box);
242
243     //help message
244     helpmsg = gtk_label_new("\n\n"
245                             "\t\tPied Piper Calculator can operte in two separate modes.\n"
246                             "\t\tBasic mode, which provides basic arithmetic operation,\n"
247                             "\t\tand PRO mode, which enables you to see the the development of\n"
248                             "\t\ta calculation in real time. It also offers additional
    functionality\t\n"
249                             "\t\tsuch as calculations with brackets and gonimetric functions.\n\n"
250                             "\t\tTo switch between the two modes you can either use the\n"
251                             "\t\tdark-blue button 'PRO/Basic mode' or you can navigate to\n"
252                             "\t\tthe top menu bar and use the section View to chagne the mode\n"
253                             "\t\tthere. Note, that any change in modes will reset the ongoing\n"
254                             "\t\tcalculation.\n\n"
255                             "\t\tTo calculate the result in Basic mode or to get the current
    result\t\t\n"
256                             "\t\tin PRO mode use the '=' button. To reset the ongoing calculation\n"
257                             "\t\tuse the 'C' button. And to delete the last character of the\n"
258                             "\t\tcalculation use the backspace button. The 'x^y' button can be\n"
```

```
259                             "\t\talso used as a root function, when the absolute value of 'y' is\n"
260                             "\t\tbetween 0 and 1. Other buttons have expected functionalites as\t\t\n"
261                             "\t\ttheir labels describes.\n"
262                             "\n\n");
263
264     gtk_box_pack_start(GTK_BOX(p_v_box), helpmsg, TRUE, FALSE, 0);
265     gtk_widget_show_all(p_window);
266     gtk_main();
267 }
```

5.2.2.5 on_pro_mode_active()

```
void GUI::on_pro_mode_active ( )  [static], [protected]
```

This function is responsible for switching to the PRO mode after choosing the option in the menu bar View –> Professional mode. Checks graphics_mode variable and switch the mode if it is needed, otherwise does nothing.

Definition at line 210 of file graphics.cpp.

```
211 {
212     if(graphics_mode)
213         return;
214
215     on_switch_state_set();
216 }
```

5.2.2.6 on_switch_state_set()

```
void GUI::on_switch_state_set ( )  [static], [protected]
```

This function resizes the window, moves container with the basic buttons and makes the advanced functions visible. When gets a callback from the switcher button switch the mode – moves container with basic buttons and makes advanced menu visible/invisible. Also, switching from basic to PRO mode sets graphics_mode variable to true (necessary for connecting to logic).

Definition at line 271 of file graphics.cpp.

```
272 {
273     if (!graphics_mode){
274         gtk_label_set_text(GTK_LABEL(mode), "Basic\nmode");
275         gtk_fixed_move(GTK_FIXED(fixed), main_menu, 0, 175);
276         gtk_widget_show_all(advanced_menu);
277
278         graphics_mode = true;
279     }else{
280         gtk_label_set_text(GTK_LABEL(mode), "PRO\nmode");
281         gtk_fixed_move(GTK_FIXED(fixed), main_menu, 0, 100);
282         gtk_widget_hide(advanced_menu);
283
284         graphics_mode = false;
285     }
286
287     logic.real_time_calculation('C', label_text, label_text_advanced);
288     logic.erase_equation();
289     label_text_advanced.push_back('0');
290     gtk_label_set_text(GTK_LABEL(label), label_text.c_str());
291     gtk_label_set_text(GTK_LABEL(label_advanced), label_text_advanced.c_str());
292 }
```

5.2.3 Member Data Documentation

5.2.3.1 advanced_menu

```
GtkWidget * GUI::advanced_menu = NULL  [static], [protected]
```

Container with advanced functions

Definition at line 35 of file graphics.h.

5.2.3.2 basic_mode

```
GtkWidget * GUI::basic_mode = NULL  [static], [protected]
```

Basic mode option in menu

Definition at line 49 of file graphics.h.

5.2.3.3 builder

```
GtkBuilder * GUI::builder = NULL  [static], [protected]
```

Widget for connecting to the grape.glade file

Definition at line 40 of file graphics.h.

5.2.3.4 button

```
GtkWidget * GUI::button = {0,}  [static], [protected]
```

Definition at line 36 of file graphics.h.

5.2.3.5 button_names

```
char const * GUI::button_names  [static], [protected]
```

Initial value:

```
= {"button0", "button1", "button2", "button3", "button4", "button5",
       "subb_button",
       "clear_button",
       "pow_button",
       "cos_button", "tan_button"}
```

```
"button6", "button7", "button8", "button9", "add_button",
"mul_button", "div_button", "mod_button", "sign_button",
"comma_button", "equal_button", "fact_button",
"OB_button", "CB_button", "bck_button", "sin_button",
```

IDs of all buttons declared in the grap.glade file

Definition at line 54 of file graphics.h.

5.2.3.6 button_text

```
char GUI::button_text  [static], [protected]
```

Initial value:

```
= {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '-', '*', '/',
                                                '%', 'N', 'C', '.', '=', '!', '^', '(', ')', 'D', 's', 'c',
      't'}
```

One char values of every button

Definition at line 55 of file graphics.h.

5.2.3.7 calculator

```
GtkWidget * GUI::calculator = NULL  [static], [protected]
```

Calculator menu button

Definition at line 45 of file graphics.h.

5.2.3.8 display

```
GdkDisplay * GUI::display = NULL  [static], [protected]
```

Widget for setting the CSS styles

Definition at line 44 of file graphics.h.

5.2.3.9 fixed

```
GtkWidget * GUI::fixed = NULL  [static], [protected]
```

Main grid of the app

Definition at line 33 of file graphics.h.

5.2.3.10 graphics_mode

```
bool GUI::graphics_mode  [static], [protected]
```

true for PRO mode, false for Basic mode

Definition at line 58 of file graphics.h.

5.2.3.11 help

```
GtkWidget * GUI::help = NULL  [static], [protected]
```

Help menu button

Definition at line 48 of file graphics.h.

5.2.3.12 index_arr

```
int GUI::index_arr  [static], [protected]
```

Initial value:

```
= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
                                    15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26}
```

Array the assigned button numbers

Definition at line 53 of file graphics.h.

5.2.3.13 label

```
GtkWidget * GUI::label = NULL  [static], [protected]
```

Main label for displaying the result

Definition at line 37 of file graphics.h.

5.2.3.14 label_advanced

```
GtkWidget * GUI::label_advanced = NULL  [static], [protected]
```

Advanced label for displaying the steps of the calculation process

Definition at line 38 of file graphics.h.

5.2.3.15 label_text

```
std::string GUI::label_text  [static], [protected]
```

text to be leaded to the claculator main dispaly

Definition at line 56 of file graphics.h.

5.2.3.16 label_text_advanced

```
std::string GUI::label_text_advanced  [static], [protected]
```

text to be leaded to the claculator advanced dispaly

Definition at line 57 of file graphics.h.

5.2.3.17 logic

```
Logic GUI::logic  [static], [protected]
```

Object of the calculation

Definition at line 52 of file graphics.h.

5.2.3.18 main_menu

```
GtkWidget * GUI::main_menu = NULL  [static], [protected]
```

Container with basic functions

Definition at line 34 of file graphics.h.

5.2.3.19 mode

```
GtkWidget * GUI::mode = NULL  [static], [protected]
```

Name of the current mode

Definition at line 39 of file graphics.h.

5.2.3.20 pro_mode

```
GtkWidget * GUI::pro_mode = NULL  [static], [protected]
```

PRO mode option in menu

Definition at line 50 of file graphics.h.

5.2.3.21 provider

```
GtkCssProvider * GUI::provider = NULL  [static], [protected]
```

Widget for setting the CSS styles

Definition at line 43 of file graphics.h.

5.2.3.22 quit

```
GtkWidget * GUI::quit = NULL  [static], [protected]
```

Calculator quit option in menu

Definition at line 47 of file graphics.h.

5.2.3.23 screen

```
GdkScreen * GUI::screen = NULL  [static], [protected]
```

Widget for setting the CSS styles

Definition at line 42 of file graphics.h.

5.2.3.24 switcher_button

```
GtkWidget * GUI::switcher_button = NULL  [static], [protected]
```

Button for switching between modes

Definition at line 41 of file graphics.h.

5.2.3.25 view

```
GtkWidget * GUI::view = NULL  [static], [protected]
```

top menu bar

Definition at line 46 of file graphics.h.

5.2.3.26 window

```
GtkWidget * GUI::window = NULL  [static], [protected]
```

Main window widget

Definition at line 32 of file graphics.h.

The documentation for this class was generated from the following files:

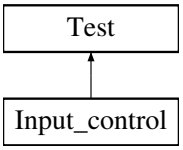
- include/graphics.h
- source/graphics.cpp

5.3 Input_control Class Reference

class that tests the equation_string_control function

```
#include <tests.h>
```

Inheritance diagram for Input_control:



Protected Member Functions

- bool parse_string (std::string input_str, std::string expected_output_str)

  function used to input charcters one by one to the equation_string_control function

Protected Attributes

- Logic test
- std::string output

5.3.1 Detailed Description

class that tests the equation_string_control function

Definition at line 90 of file tests.h.

5.3.2 Member Function Documentation

5.3.2.1 parse_string()

```
bool Input_control::parse_string (
            std::string input_str,
            std::string expected_output_str )  [protected]
```

function used to input charcters one by one to the equation_string_control function

Parameters

input_str	string needed to be parsed
expected_output_str	the expected output of the equation_string_control

Returns

true on success (equation_string_control output matches the expected_output_str), otherwise false

Definition at line 32 of file test_mock_stub.cpp.

```
33 {
34     for (unsigned i = 0; i < input_str.size(); i++)
35     {
36         test.equation_string_control(output, input_str[i]);
37     }
38     return (output == expected_output_str)?(1):(fprintf(stderr, "%s\n", output.c_str()), 0);
39 }
```

5.3.3 Member Data Documentation

5.3.3.1 output

```
std::string Input_control::output  [protected]
```

Definition at line 94 of file tests.h.

5.3.3.2 test

```
Logic Input_control::test  [protected]
```

Definition at line 93 of file tests.h.

The documentation for this class was generated from the following files:

- include/tests.h
- source/test_mock_stub.cpp

5.4 Logic Class Reference

class containing all declaration needed to calculate complex equations, on which either GUI can be added or can be run from the terminal

```
#include <logic.h>
```

Public Member Functions

- Logic ()

  Constructor.
- void equation_string_control (std::string &current, char addition)

  Checks if a character can be added to an ongoing equation, so that the equation would retain mathematical sense, if character cannot be added the equation string remains the same.
- void reset_equation_string_control ()

  Resets the equation string control to the starting state.
- long double get_result ()
- void result_print ()

  Prints the result to stdout.
- void erase_equation ()

  Erases the currently ongoing equation (including the equation string control), sets everything to the starting state, so that new equation can start.
- int real_time_calculation (char input_char, std::string &output_str_result, std::string &output_str_equation)

  Provides new result after any added character, supported characters: '0'-'9' for digits '+', '-', '', '/', '^', '?', '', '!', '=' for operators ('?' represents the nrooth of a number) '.' for decimal point 'N' for negation of a number 'D' for deletion of the last character 'C' for clearing the equation '(', ')' for brackets other characters are ignored.*

Related Functions

(Note that these are not member functions.)

- int calculate (std::string input_str, std::string &output_str)

  Parse the input string to numbers and operators and uses.
- int calculate (std::string input_str, long double &output_result)

  Parses the input string to numbers and operators and uses.

5.4.1 Detailed Description

class containing all declaration needed to calculate complex equations, on which either GUI can be added or can be run from the terminal

Definition at line 123 of file logic.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Logic()

```
Logic::Logic ( )
```

Constructor.

Definition at line 13 of file logic.cpp.

```
14 {
15     //some variables are initilized
16     result = 0.0;
17     currently_entered_num = 0.0;
18     position = INTEGER;
19     op_count_strcnt = 0;
20     result_parts_op.push_back(0);
21     result_parts.push_back(0);
22     decimal_position = 0;
23     result_parts_state.push_back(START_STATE);
24     bracket_count = 0;
25     negate = false;
26 }
```

5.4.3 Member Function Documentation

5.4.3.1 equation_string_control()

```
void Logic::equation_string_control (
            std::string & current,
            char addition )
```

Checks if a character can be added to an ongoing equation, so that the equation would retain mathematical sense, if character cannot be added the equation string remains the same.

Parameters

current	The string holding the text of an ongoing equation
addition	Currently added character

Definition at line 1400 of file logic.cpp.

```
1401 {
1402     static bool active_int_op; //for modulo as it can be used only with integers
1403     static bool active_minus;  //remembers if the current number is negative or not
1404
1405     if (current.size() == 1 && current.back() == '0' && addition != '.')
1406     {
1407         current.pop_back();
1408     }
1409
1410     //when the function is called for the first time or was reseted, it has to find the state of the
      current equation
1411     if (!current.size()) //empty current string
1412     {
1413         position = INTEGER;
1414         active_minus = false;
1415         active_int_op = false;
1416     }
1417     else if (!IS_DIGIT(current.back()) && position == NO_STATE) //this cleares invalid strings
1418     {
1419         current.clear();
1420         position = INTEGER;
1421     }
1422     else if (position == NO_STATE) //there is a number in the current string, but the string control
      was reseted
1423     {
1424         position = INTEGER;          //default sate
1425         for (int i = current.size()-1; i >= 0; i-)
1426         {
1427             if (current[i] == '.') //when it is floating point number, then the state is changed
1428             {
1429                 position = DECIMAL_NUM;
1430                 break;
1431             }
1432         }
1433         if (current[0] == '-') //if the number is negative
1434             active_minus = true;
1435         else
1436             active_minus = false;
1437     }
1438
1439     switch(position) //the state from last call of this function is now used to detemin what has to
      happen it the currnet call
1440     {
1441         case INTEGER: //when digits before decimal point are added (the number is integer)
1442             op_count_strcnt = 0; //the number of operators is reseted
1443             if (IS_DIGIT(addition)) //another digit was added
1444                 position = INTEGER;
1445             else if (addition == '-' && !current.size() && !active_minus) //when the first character
      added to the equation is minus
1446                 active_minus = true;
1447             else if ((active_minus && current.size() == 1) || current.size() == 0) //special operators
      that can be added to the
```

```
1448            {                                                                   //beginnig of an
    equation
1449                if (addition == '(')
1450                {
1451                    position = O_BRACKET; //the state is changed for next call of the function
1452                    current.push_back('(');
1453                    bracket_count++; //number of not closed brackets is increased
1454                }
1455                else if (addition == 's')
1456                {
1457                    current += "sin";
1458                    position = GONIOOMETRY; //the state is changed for next call of the function
1459                }
1460                else if (addition == 't')
1461                {
1462                    current += "tan";
1463                    position = GONIOOMETRY; //the state is changed
1464                }
1465                else if (addition == 'c')
1466                {
1467                    current += "cos";
1468                    position = GONIOOMETRY; //the state is changed
1469                }
1470
1471                    return;
1472            }
1473            else if (IS_OPERATOR(addition) && current.size()) //if there is already at least one digit
    operator can be added
1474            {
1475                if (current.back() == '-') //if the current.size() == 1 and the first character is
    minus, operator cannot be added
1476                    return;
1477
1478                if (current.size() && current.back() == '.') //if the last character is a decimal dot,
    then it is deleted
1479                    current.pop_back();
1480
1481                if (addition == '!' && active_minus) //if the current number is negative, then
    factorial is ignored
1482                {
1483                    return;
1484                }
1485
1486                if (addition == '%') //if the added operator is % then the next number can be only an
    integer
1487                    active_int_op = true;
1488                else
1489                    active_int_op = false;
1490
1491                position = OPERATOR; //the next position is set
1492
1493                if (addition != '!') //the number of currently added oprators is increased, factorial
    is not included
1494                    op_count_strcnt++;
1495
1496                active_minus = false;
1497            }
1498            else if (addition == '.' && !active_int_op) //if the floating point number can be added
1499            {
1500                if (current.size() == 0) //if the string is empty, the number cannot start with decimal
    dot
1501                    return;
1502                position = DECIMAL_NUM; //position is changed
1503            }
1504            else if (addition == 'D' && current.size()) //if the delete button was pressed
1505            {
1506                current.pop_back(); //one character is deleted
1507
1508                if (current.size() == 0) //if the string is empty
1509                {
1510                    active_minus = false;
1511                    return;
1512                }
1513
1514                if (current.back() == 's' || current.back() == 'n') //if the last character of the
    string after deleting is
1515                {                                                   // cos, sin or tan
1516                    position = GONIOOMETRY;
1517                    return;
1518                }
1519
1520                if (current.size() > 1 && IS_OPERATOR(current.back()))
1521                {
1522                    if (current.back() == '-' && (current[current.size() - 2] == '(' ||
    current[current.size() - 2] == 'n' ||
1523                        current[current.size() - 2] == 's')) //if the deleted character was a digit and
    the number was negative,
```

```
1524                    {                                               //then it reamains in this position
1525                        position = INTEGER;
1526                        active_minus = true;
1527                        return;
1528                    }
1529                    //otherwise the position changes
1530                    position = OPERATOR;
1531                    op_count_strcnt = 1; //by default, there is at least one operator
1532                    if (IS_OPERATOR_NF(current[current.size()-2])) // but there can be two
1533                        op_count_strcnt = 2;
1534                }
1535                else if (current.back() == '(') //if the number was inside bracket
1536                {
1537                    position = O_BRACKET;
1538                }
1539
1540                return; //no characters are added
1541
1542            }
1543            else if (addition == ')' && bracket_count > 0) //if closed bracket can be added
1544            {
1545                bracket_count-; // number of open brackets is decreased
1546                position = C_BRACKET;
1547            }
1548            else if (addition == '(' && current.size() == 0) //if the first character added is a open
    bracket
1549            {
1550                bracket_count++;
1551                position = O_BRACKET;
1552            }
1553            else if (addition == 'N') //negate button was pressed
1554            {
1555                for (int i = current.size()-1; i >= 0; i-) //the string is searched from the end
1556                {
1557
1558                    if (IS_OPERATOR(current[i])) //if operator was found, that means that the current
    number ends
1559                    {
1560                        //if the number is not already negative
1561                        if (i > 0 && (IS_DIGIT(current[i-1]) || current[i-1] == ')' || current[i-1] ==
    '!'))
1562                        {
1563                            active_minus = true;
1564                            current.insert(i+1, 1, '-'); //minus is inserted
1565                        }
1566                        //if the number is negative
1567                        else
1568                        {
1569                            active_minus = false;
1570                            current.erase(i, 1); //minus is deleted
1571                        }
1572
1573                        break;
1574                    }
1575                    else if ((current[i] == 'n' || current[i] == 's')) //ensures the negation after
    goniometric functions
1576                    {
1577                        current.insert(i+1, 1, '-');
1578                        break;
1579                    }
1580                    else if (current[i] == '(') // if the number is in bracket, it can also be negated
1581                    {
1582                        active_minus = true;
1583                        current.insert(i+1, 1, '-');
1584                        break;
1585                    }
1586                    else if (i == 0) //or if the number has no operators in front, then it can also be
    negated
1587                    {
1588                        active_minus = true;
1589                        current.insert(0, 1, '-');
1590                    }
1591                }
1592                return; //nothing else is added to the string
1593            }
1594            else //if something else is added
1595                return;
1596
1597            current.push_back(addition); //the character is added to the string
1598            return;
1599
1600        case DECIMAL_NUM: //when digits after decimal point are added
1601            op_count_strcnt = 0;
1602
1603            //these characters cannot be added after a floating point number
1604            if (addition == '.' || addition == '(' || (addition == ')' && current.back() == '.'))
1605                return;
```

```
1606
1607            //modulo and factorial cannot be used after floating point number
1608            if (addition == '+' || addition == '-' || addition == '*' || addition == '/' || addition ==
    '^' ||
1609                addition == '?')
1610            {
1611                if (current.back() == '.') //if the last character is '.', then it is stripped
1612                    current.pop_back();
1613                position = OPERATOR;
1614
1615                    op_count_strcnt++; //the operator count is increased
1616            }
1617            else if (addition == 'D') //deletion is pressed
1618            {
1619                if (current.back() == '.') //the state has to change when the deleted character is '.'
1620                    position = INTEGER;
1621                current.pop_back();
1622                return;
1623            }
1624            else if (addition == '!' || addition == '%') //modulo and factorial cannot be used after
    floating point number
1625                return;
1626            else if (addition == ')' && bracket_count > 0) //if it is possible to add closed bracket
1627            {
1628                bracket_count-;
1629                position = C_BRACKET;
1630            }
1631            else if (addition == 'N')
1632            {
1633                for (int i = current.size()-1; i >=0; i-) //the string is searched from the end
1634                {
1635
1636                    if (IS_OPERATOR(current[i])) //if operator was found, that means that the current
    number ends
1637                    {
1638                        //if the number is not already negative
1639                        if (i > 0 && (IS_DIGIT(current[i-1]) || current[i-1] == ')' || current[i-1] ==
    '!'))
1640                        {
1641                            active_minus = true;
1642                            current.insert(i+1, 1, '-'); //minus is added
1643                        }
1644                        //if it is negative
1645                        else
1646                        {
1647                            active_minus = false;
1648                            current.erase(i, 1); //minus is removed
1649                        }
1650
1651                        break;
1652                    }
1653                    else if ((current[i] == 'n' || current[i] == 's')) //if the number is as an
    argument of a goniometric function
1654                    {
1655                        current.insert(i+1, 1, '-');
1656                        break;
1657                    }
1658                    else if (current[i] == '(') // if the number is in bracket, it can also be negated
1659                    {
1660                        active_minus = true;
1661                        current.insert(i+1, 1, '-');
1662                        break;
1663                    }
1664                    else if (i == 0) //if the equation starts with the negated number
1665                    {
1666                        active_minus = true;
1667                        current.insert(0, 1, '-');
1668                    }
1669                }
1670                return;
1671            }
1672            else if (!IS_DIGIT(addition)) //if something other than digit and the cases above is added,

1673                return;                  //then it is not added to the string
1674
1675            current.push_back(addition); //character is added to the string
1676            return;
1677
1678
1679        case OPERATOR: //when multiple opperators are added
1680            //theses characters cannot be added after operator
1681            if (addition == '.' || (addition == ')' && current.back() != '!') || addition == 'N')
1682                return;
1683
1684            if (IS_DIGIT(addition) && current.back() != '!') //when digit is added and the operator is
    not factorial
1685                position = INTEGER; //position is changed
```

```
1686
1687            else if (IS_OPERATOR(addition)) //if more operators are added
1688            {
1689                if (addition == '!' && !IS_DIGIT(current.back())) //factorail can be added only after
    digit
1690                    return;
1691
1692                op_count_strcnt++;
1693
1694                if ((addition != '-' && op_count_strcnt == 2)) //if the newly added operator is not
    minus, which represents the
1695                {                                                   //the negation of the next number, than
    the operators are changed
1696                    current.pop_back(); //operator is removed
1697                    op_count_strcnt-;
1698                    active_int_op = false;
1699                }
1700                else if (op_count_strcnt == 2) //otherwise minus is activated
1701                    active_minus = true;
1702
1703                if (op_count_strcnt == 3) //if there is already an operator and minus after that
    operator and another operator
1704                {                           //is added, then two has to be deleted first
1705                    current.pop_back(); //operator is removed
1706                    current.pop_back(); //operator is removed
1707                    op_count_strcnt = 1; //number of operators is reseted to 1
1708                    active_minus = false;
1709                    active_int_op = false;
1710                }
1711
1712                if (addition == '%') //if modulo is added, the next number has to be integer
1713                    active_int_op = true;
1714            }
1715            else if (addition == 'D') //when deleting
1716            {
1717                active_minus = false; //if there is a minus, which has the function of negation, it
    will be always deleted first
1718                current.pop_back(); //charcter is removed
1719
1720                if (IS_DIGIT(current.back())) //if the last character of the string is digit
1721                {
1722                    active_int_op = false;  //modulo was surely removed
1723                    position = INTEGER;  //by default
1724                    for (int i = current.size()-2; i >= 0; i-)
1725                    {
1726                        if (current[i] == '.')
1727                        {
1728                            position = DECIMAL_NUM; //if the number was floating point
1729                            break;
1730                        }
1731                        else if (IS_OPERATOR(current[i])) //if the number wasn't floating point
1732                            break;
1733                    }
1734                }
1735                else if (current.back() == ')')
1736                {
1737                    position = C_BRACKET;
1738                }
1739                    op_count_strcnt-; //number od operators is decreased
1740
1741                return;
1742            }
1743            else if (addition == '(' && op_count_strcnt > 0) //open bracket can be added only after
    operator
1744            {
1745                bracket_count++;
1746                op_count_strcnt = 0;
1747                position = O_BRACKET;
1748            }
1749            //goniometric functions
1750            else if (addition == 's' && current.back() != '!')
1751            {
1752                current += "sin"; //3 charcters have to be inserted
1753                position = GONIOOMETRY;
1754                return;
1755            }
1756            else if (addition == 'c' && current.back() != '!')
1757            {
1758                current += "cos"; //3 charcters have to be inserted
1759                position = GONIOOMETRY;
1760                return;
1761            }
1762            else if (addition == 't' && current.back() != '!')
1763            {
1764                current += "tan"; //3 charcters have to be inserted
1765                position = GONIOOMETRY;
1766                return;
```

```
1767            }
1768            else if (addition == ')') //at this point the current.back() == !, so close bracket can be
    added
1769            {
1770                bracket_count-;
1771                position = C_BRACKET;
1772            }
1773            else
1774                return;
1775
1776            current.push_back(addition); //the charcter is inserted to the stirng
1777            return;
1778
1779        case O_BRACKET:
1780            //charcters that cannot be added after open bracket
1781            if ((addition == ')' || IS_OPERATOR(addition) || addition == '.' || addition == 'N') &&
1782                (addition != '-' || current.back() == '-'))
1783                return;
1784
1785            else if (IS_DIGIT(addition)) //when digit is added
1786            {
1787                position = INTEGER;
1788            }
1789            else if (addition == '-') //minus that works as a negation is added
1790            {
1791                active_minus = true;
1792            }
1793            else if (addition == 'D') //when deleting
1794            {
1795                current.pop_back();
1796                bracket_count-; //by default the bracket count is decreased
1797                if (IS_OPERATOR(current.back())) //if the last not deleted character is an operator
1798                {
1799                    if (current.back() == '-' && IS_OPERATOR(current[current.size()-2])) //either there
    are two
1800                    {
1801                        op_count_strcnt = 2;
1802                    }
1803                    else
1804                        op_count_strcnt = 1; //or one
1805                    position = OPERATOR;
1806                }
1807                else if (current.back() == 'n' || current.back() == 's') //if the last not deleted
    charcter represents goniometric function
1808                    position = GONIOOMETRY;
1809
1810                return;
1811            }
1812            else if (addition == '(') //if the addition is another open bracket
1813                bracket_count++;
1814
1815            //when goniometric functions are added directly after open bracket
1816            else if (addition == 's')
1817            {
1818                current += "sin";
1819                position = GONIOOMETRY;
1820                return;
1821            }
1822            else if (addition == 'c')
1823            {
1824                current += "cos";
1825                position = GONIOOMETRY;
1826                return;
1827            }
1828            else if (addition == 't')
1829            {
1830                current += "tan";
1831                position = GONIOOMETRY;
1832                return;
1833            }
1834            else //if something else
1835                return;
1836
1837            current.push_back(addition); //open bracket is added to the string
1838            break;
1839
1840        case C_BRACKET:
1841            //these character cannot be added after a close bracket
1842            if (addition == '(' || addition == '.' || IS_DIGIT(addition))
1843                return;
1844            else if (addition == '!') //factorial is added
1845            {
1846                op_count_strcnt = 0;
1847                position = OPERATOR;
1848            }
1849            else if (IS_OPERATOR(addition)) //operator is added
1850            {
```

```
1851                active_minus = false;
1852                op_count_strcnt = 1;
1853                position = OPERATOR;
1854            }
1855            else if (addition == 'D') //when deleting
1856            {
1857                current.pop_back();
1858                bracket_count++; //number of open brackets have to be increased
1859
1860                if (current.back() == ')') //if there are multiple close brackets
1861                    return;
1862
1863                else if (current.back() == '!') //if the last charcter is factorial, other operators
    cannot occure
1864                {
1865                    op_count_strcnt = 0;
1866                    position = OPERATOR;
1867                }
1868                else //when the last operator is a digit
1869                {
1870                    active_int_op = false;
1871                    position = INTEGER; //by default it is an integer
1872                    for (int i = current.size()-2; i >= 0; i-)
1873                    {
1874                        if (current[i] == '.') //if decimal dot occures, it is a floating point number
1875                        {
1876                            position = DECIMAL_NUM;
1877                            break;
1878                        }
1879                        else if (IS_OPERATOR(current[i]))
1880                            break;
1881                    }
1882                }
1883
1884                return; //no characters are added
1885            }
1886            else if (addition == 'N') //negation
1887            {
1888                for (int i = current.size()-1, j = 0; i >= 0; i-) //the string is searched from the end
1889                {
1890                    if (current[i] == ')') //close brackets are skipped and the counter is increased
1891                        j++;
1892                    else if (j && current[i] == '(') //open brackets are skipped only when closed
    bracket was skipped before
1893                        j-;
1894                    else if (j == 0 && IS_OPERATOR(current[i])) //when operator occures
1895                    {
1896                        //when the number is positive, negation is added
1897                        if (i > 0 && (IS_DIGIT(current[i-1]) || current[i-1] == ')' || current[i-1] ==
    '!'))
1898                        {
1899                            active_minus = true;
1900                            current.insert(i+1, 1, '-');
1901                        }
1902                        //when the number is negative, the negation is deleted
1903                        else
1904                        {
1905                            active_minus = false;
1906                            current.erase(i, 1);
1907                        }
1908
1909                        break;
1910                    }
1911                    else if (j == 0 && (current[i] == 'n' || current[i] == 's')) //neagtion inside
    goniometric function
1912                    {
1913                        current.insert(i+1, 1, '-');
1914                        break;
1915                    }
1916                    else if (j == 0 && current[i] == '(') //negation in front of open bracket
1917                    {
1918                        active_minus = true;
1919                        current.insert(i+1, 1, '-');
1920                        return;
1921                    }
1922
1923                    if (i == 0) //negation when the bracket is the first part of the equation
1924                    {
1925                        active_minus = true;
1926                        current.insert(i, 1, '-');
1927                    }
1928                }
1929
1930                return;
1931            }
1932            else if (bracket_count && addition == ')') //the close bracket is added
1933                bracket_count-;
```

```
1934             else
1935                 return;
1936
1937             current.push_back(addition); //the character is added to the string
1938             break;
1939
1940         case GONIOOMETRY:
1941             if (addition == '-' && current.back() != '-') //to create negative number inside the
    goniometric function
1942             {
1943                 current.push_back('-');
1944                 active_minus = true;
1945             }
1946             if (IS_DIGIT(addition)) //digit is added
1947             {
1948                 current.push_back(addition);
1949                 position = INTEGER;
1950             }
1951             if (addition == '(') //open bracket is added
1952             {
1953                 bracket_count++;
1954                 position = O_BRACKET;
1955                 current.push_back('(');
1956             }
1957             if (addition == 'D')
1958             {
1959                 current.erase(current.end()-3, current.end()); // 3 characters have to be deleted
1960                 if (current.size())
1961                 {
1962                     if (current.size() == 1 && current.back() == '-') //when there is only one
    character and it is minus
1963                     {                                                 //then the position is going to
    be a negative integer
1964                         position = INTEGER;
1965                         active_minus = true;
1966                         return;
1967                     }
1968                     if (IS_OPERATOR_NF(current.back())) //if the currently last character is an
    operator,
1969                     {                                   //then the number of oprators is loaded
1970                         op_count_strcnt = 1;
1971                         if (IS_OPERATOR_NF(current[current.size() - 2]))
1972                             op_count_strcnt = 2;
1973                         position = OPERATOR;
1974                         return;
1975                     }
1976                 }
1977             }
1978
1979             break;
1980
1981         case NO_STATE: //should not happen
1982             return;
1983     }
1984 }
```

5.4.3.2 erase_equation()

```
void Logic::erase_equation ( )
```

Erases the currently ongoing equation (including the equation string control), sets everything to the starting state, so that new equation can start.

Definition at line 1986 of file logic.cpp.

```
1987 {
1988     //erases the variables that are used for the current ongoing equation and sets them to the default
    values
1989     result = 0;
1990     result_parts.clear();
1991     result_parts_op.clear();
1992     result_parts_state.clear();
1993     result_parts_op.push_back(0);
1994     result_parts.push_back(0);
1995     result_parts_state.push_back(START_STATE);
1996     reset_equation_string_control(); //the equation control also has to be reseted
1997 }
```

5.4.3.3 get_result()

```
long double Logic::get_result ( )
```

Returns

Returns the result

Definition at line 28 of file logic.cpp.

```
29 {
30     return result;
31 }
```

5.4.3.4 real_time_calculation()

```
int Logic::real_time_calculation (
            char input_char,
            std::string & output_str_result,
            std::string & output_str_equation )
```

Provides new result after any added character, supported characters: '0'-'9' for digits '+', '-', '*', '/', '^', '?', '', '!', '=' for operators ('?' represents the nrooth of a number) '.' for decimal point 'N' for negation of a number 'D' for deletion of the last character 'C' for clearing the equation '(', ')' for brackets other characters are ignored.

Parameters

input_char	New addition to an ongoing equation or starts a new equation.
output_str_result	String where the new result, or the error message will be stored
output_str_equation	String where the ongoing equation will be stored in the text form

Returns

0 on success (includes wrong character), otherwise 1, when mathematical error occures

Definition at line 1111 of file logic.cpp.

```
1112 {
1113     //setting the locale, so that the decimal dot is used rather than comma
1114     std::setlocale(LC_ALL, "C");
1115     //static variables that holds data about the ongoing equation
1116     static bool add;
1117     static unsigned add_position;
1118     static char last_input;
1119     static unsigned digit_count;
1120
1121     //when the equation is cleared or the last character is deleted, then the equation is reseted to
       the starting state
1122     if (input_char == 'C' || (input_char == 'D' && output_str_equation.size() == 1))
1123     {
1124         reset_equation_string_control();
1125         output_str_equation.clear();
1126         output_str_result.clear();
1127         //output_str_result.push_back('0');
1128         add_to_calculation('C', last_input);
1129         add = false;
1130         add_position = 0;
1131     }
1132     else if (input_char == 'D') //when character is deleted
1133     {
1134         last_input = output_str_equation.back();
```

```
1135        //the request is sent to the equation string cotrol to change the string accordingly
1136        equation_string_control(output_str_equation, input_char);
1137
1138        if (last_input == 'n' || last_input == 's') //if goniometric function was deleted
1139        {
1140            add_position -= 2; //the position of currentlu added character to a equation is reduced
    accordingly
1141        }
1142        if (add_position == output_str_equation.size() + 1) //if the deleted character was already
    added to the equation
1143        {                                                   //f.e. operators aren't added immidiately,
    but with the first following digit3
1144            add = false;
1145            if (last_input == '(' || last_input == ')' || last_input == '!') //the equation has to be
    recalculated
1146            {
1147                recalculate_after_delete(output_str_equation);
1148            }
1149            else if (last_input == '.') //get state has to be informed, that integer was created from a
    floating point number
1150            {
1151                get_state('q', 0);
1152            }
1153            else if (output_str_equation.back() == '(') //if the current last character in the equation
    string is bracket,
1154            {                                           //then the equation has to be recalculated, to
    avoid
1155                currently_entered_num = 0;              //calculation with 0
1156                decimal_position = 0;
1157                recalculate_after_delete(output_str_equation);
1158            }
1159            else if (IS_OPERATOR_NF(output_str_equation.back())) //if the current last character is an
    operator, then
1160            {                                                    //the equation has to be recalculated,
    but without the operators,
1161                currently_entered_num = 0;                       //so that the operator can be changed
1162                decimal_position = 0;
1163                add = true;
1164                add_position-;
1165                char hold1 = output_str_equation.back(); //the first operator is separated
1166                output_str_equation.pop_back();
1167                if (IS_OPERATOR_NF(output_str_equation.back())) //if there is +-, *-, /-, ... two
    operators have to be separated
1168                {
1169                    add_position-;
1170                    char hold2 = output_str_equation.back(); //the first operator is separated
1171                    output_str_equation.pop_back();
1172                    recalculate_after_delete(output_str_equation);
1173                    output_str_equation.push_back(hold2);   //operator is added back
1174                    output_str_equation.push_back(hold1);   //operator is added back
1175                }
1176                else
1177                {
1178                    recalculate_after_delete(output_str_equation);
1179                    output_str_equation.push_back(hold1); //operator is added back
1180                }
1181            }
1182            //if the deleted operator was a gonimetric function, the result has to be recalculated
1183        else if (output_str_equation.back() == 'n' || output_str_equation.back() == 's')
1184            {
1185                currently_entered_num = 0;              //calculation with 0
1186                decimal_position = 0;
1187                recalculate_after_delete(output_str_equation);
1188            }
1189            else if (IS_DIGIT(last_input)) //the currently entered number has to change
1190            {
1191                if (decimal_position > 2) //if it is a floating point number
1192                {
1193                    if (negate)
1194                        currently_entered_num += (last_input - '0')/decimal_position;
1195                    else
1196                        currently_entered_num -= (last_input - '0')/decimal_position;
1197                }
1198                else //if it is an integer
1199                {
1200                    if (negate)
1201                        currently_entered_num = (currently_entered_num + (last_input - '0')) / 10;
1202                    else
1203                        currently_entered_num = (currently_entered_num - (last_input - '0')) / 10;
1204                }
1205
1206                decimal_position /= 10; // the decimal position is decresed
1207
1208                //the new number has to be included to the calculation
1209                if (result_parts_op.back() == '/' && currently_entered_num > -EPS &&
    currently_entered_num < EPS)
1210                {
```

```
1211                    //if the last operation is division and the currentlu entered number is 0, division
    by 0 is avoided
1212                    currently_entered_num = 1;
1213                    try
1214                    {
1215                        math_calculation();
1216
1217                    }
1218                    catch(const std::exception &e)
1219                    {
1220                        output_str_result.clear();
1221                        output_str_result.assign(e.what());
1222                        currently_entered_num = 0;
1223                        if (add_position > 0)
1224                            add_position-;
1225                        return 1;
1226                    }
1227                    currently_entered_num = 0;
1228                }
1229                else //otherwise the reusult is calculated
1230                {
1231                    try
1232                    {
1233                        math_calculation();
1234                    }
1235                    catch(const std::exception &e)
1236                    {
1237                        output_str_result.clear();
1238                        output_str_result.assign(e.what());
1239                        if (add_position > 0)
1240                            add_position-;
1241                        return 1;
1242                    }
1243                }
1244
1245            }
1246
1247            if (add_position > 0)
1248                add_position-; //the add position is decreased
1249        }
1250
1251        //if the current last character is a digit and operator or closed bracket was deleted, then the
    current number and data
1252        //about the number have to be reloaded (negative/positive, number of decimal places)
1253        if (IS_DIGIT(output_str_equation.back()) && (IS_OPERATOR(last_input) || last_input == ')'))
1254        {
1255            decimal_position = 1;
1256            for (int i = output_str_equation.size()-1; i >= 0; i-)
1257            {
1258                if (output_str_equation[i] == '.')
1259                {
1260                    get_state('Q', 0);
1261                    break;
1262                }
1263                else if (IS_OPERATOR(output_str_equation[i]) || i == 0)
1264                {
1265                    get_state('q', 0);
1266                    decimal_position = 1;
1267                    break;
1268                }
1269                decimal_position *= 10;
1270            }
1271            if (currently_entered_num < 0)
1272                negate = true;
1273            else
1274                negate = false;
1275
1276        }
1277    }
1278    else if (input_char != '=') //if digit, operators or brackets are added to the equation
1279    {
1280        //adding the character to the string, that is going to be diplayed in the window
1281        //the string is changed only when the input character is valid
1282        equation_string_control(output_str_equation, input_char);
1283
1284        if (IS_OPERATOR_NF(input_char) && !add && output_str_equation.size() > 1) //if operator is
    added
1285        {
1286            add_position = output_str_equation.size() - 1; //the position is rembered, and the next
    addition to the equation,
1287            add = true;                                    //will be performed after a first digit is
    added
1288        }
1289        else if (input_char == 'N' && add_position != output_str_equation.size())//&&
    (output_str_equation.size() && !IS_OPERATOR(output_str_equation.back())))
1290        //when the nagation button was pressed and the equation string changed (either minus was added,
    or subtracted)
```

```
1291        {
1292            try
1293            {
1294                add_to_calculation('N', last_input); //the current number is negated
1295                math_calculation();
1296            }
1297            catch(const std::exception& e)
1298            {
1299                output_str_result.clear();
1300                output_str_result.assign(e.what());
1301                return 1;
1302            }
1303
1304            add_position = output_str_equation.size();
1305            add = false;
1306        }
1307        else if (!IS_OPERATOR_NF(input_char) && add_position < output_str_equation.size() && input_char
    != 'N' &&
1308                 output_str_equation.back() == input_char) //if the added character is not operator and
    there are characters to be
1309        //added (add_position < str.size()), then the remaining characters are added
1310        {
1311            for (unsigned i = add_position; i < output_str_equation.size(); i++)
1312            {
1313                digit_count++;
1314                if (digit_count >= 20 && IS_DIGIT(output_str_equation[i])) //limitation to the size of
    a number
1315                {                                                          //solves the double overflow
1316                    output_str_equation.pop_back();
1317                    continue;
1318                }
1319                else if (digit_count == 0xFFFFFFFF)
1320                    digit_count = 20;
1321                else if (!IS_DIGIT(output_str_equation[i]))
1322                    digit_count = 0;
1323
1324                try
1325                {
1326                    add_to_calculation(output_str_equation[i], last_input); //charcter is added to the
    calculation
1327                    if (output_str_equation[i] == 't' || output_str_equation[i] == 'c' ||
    output_str_equation[i] == 's')
1328                        i += 2; //when sin, cos, tan are 3 character, two of them have to be skipped
1329
1330                    if (output_str_result[0] == 'C' || output_str_result[0] == 'D' ||
    output_str_result[0] == 'D')
1331                        break;
1332                }
1333                catch(const std::exception& e) //when something fails (division by 0, factoria of
    negative number..)
1334                {
1335                    output_str_result.clear();
1336                    output_str_result.assign(e.what());
1337                    add_position = output_str_equation.size();
1338                    return 1;
1339                }
1340                last_input = output_str_equation[i]; //the current input is stored to the last input,
    which is going to be used
1341            }                                        //in the next call of this function
1342            add_position = output_str_equation.size();
1343            add = false;
1344        }
1345    }
1346    else if (input_char == '=') //equal button was pressed
1347    {
1348        try
1349        {
1350            //add to calculation calculates the result, and then is reseted
1351            add_to_calculation('=', last_input);
1352        }
1353        catch(const std::exception& e)
1354        {
1355            output_str_result.clear();
1356            output_str_result.assign(e.what());
1357            return 1;
1358        }
1359        output_str_equation.clear();
1360        output_str_equation.assign(output_str_result);
1361
1362        //resets the string control
1363        reset_equation_string_control();
1364        add_position = output_str_result.size(); //the add position has to be reseted
1365        add = false;
1366    }
1367
1368    if (IS_DIGIT(input_char) || input_char == '!' || input_char == '=' || input_char == 'D' ||
    input_char == 'N')
```

```
1369     // when some of these buttons were pressed, the calculated result has to be diplayed on the
       calculator
1370     {
1371         //output_str_result.clear();
1372         output_str_result.assign(std::to_string(result)); //the result is converted to string
1373         output_str_result.erase(output_str_result.find_last_not_of('0') + 1, std::string::npos);
       //trailing 0 are striped
1374         if (output_str_result.back() == '.') //id the last character is '.', then it is also striped
1375             output_str_result.pop_back();
1376
1377     }
1378
1379     if (!output_str_result.size()) //if the strig is empty, then 0 is displayed
1380        {
1381         output_str_result.push_back('0');
1382        }
1383
1384     if (output_str_equation.size() && add_position > 0)
1385         last_input = output_str_equation[add_position-1]; //the last input is stored, if it is possible
1386     else
1387         last_input = 0; //if not, it does not really matter, what the last input will be
1388
1389     return 0;
1390 }
```

5.4.3.5 reset_equation_string_control()

```
void Logic::reset_equation_string_control ( )
```

Resets the equation string control to the starting state.

Definition at line 1392 of file logic.cpp.

```
1393 {
1394     //variables, that holds data about the ongoing equation are reseted
1395     bracket_count = 0;
1396     op_count_strcnt = 0;
1397     position = NO_STATE;
1398 }
```

5.4.3.6 result_print()

```
void Logic::result_print ( )
```

Prints the result to stdout.

Definition at line 33 of file logic.cpp.

```
34 {
35     //result is printed to stdin
36     std::cout « result « std::endl;
37 }
```

5.4.4 Friends And Related Function Documentation

5.4.4.1 calculate() [1/2]

```
int Logic::calculate (
             std::string input_str,
             std::string & output_str )  [related]
```

Parse the input string to numbers and operators and uses.

to calculate the final result

Parameters

input_str	String with the equation that needs to be calculated, this string has to be parsed, so it has mathematical sence, otherwise the behaviour is undefined
output_str	Variable where the final result converted from long double, or an error message will be stored.

Returns

Returs 0 on success otherwise 1

Definition at line 269 of file logic.cpp.

```
270 {
271     //adding R resets the current calculation
272     add_to_calculation('R', ' ');
273     output_str.clear();
274     std::setlocale(LC_ALL, "C");
275     std::size_t increase;
276     for (std::size_t i = 0; i < input_str.size(); i++)
277     {
278         //the test can be converted to a number if this conditions is true
279         if ((input_str[i] >= '0' && input_str[i] <= '9') || (i == 0 && input_str.size() > 1 &&
280             input_str[i] == '-' && IS_DIGIT(input_str[1])) || ( input_str[i] == '-' &&
281           (input_str[i-1] == '+' || input_str[i-1] == '-' || input_str[i-1] == '*' || input_str[i-1] ==
    '/' ||
282           input_str[i-1] == '^' || input_str[i-1] == '?' ||  input_str[i-1] == '%' || input_str[i-1] ==
    '(')
283           && i != input_str.size() - 1 && (i + 1 < input_str.size() && input_str[i+1] != '(')))
284         {
285             currently_entered_num = std::stold(&input_str[i], &increase);
286             i += increase; //the iterator is increased for the lenght of the converted number
287         }
288
289         if (i < input_str.size())
290         {
291             try
292             {
293                 //the first operator is added
294                 add_to_calculation(input_str[i], (i > 0)?input_str[i-1]:0);
295
296                 //if there are multiple operators, they are also added
297                 while (i + 2 < input_str.size() && !IS_DIGIT(input_str[i + 1]) && !IS_DIGIT(input_str[i
    + 2]))
298                 {
299                     i++;
300                     if (input_str[i] == '-' && (IS_OPERATOR_NF(input_str[i-1])))
301                     {
302                         add_to_calculation('M', ' ');
303                     }
304                     else
305                     {
306                         add_to_calculation(input_str[i], input_str[i - 1]);
307                     }
308
309                     if(input_str[i] == 's' || input_str[i] == 'c' || input_str[i] == 't')
310                         i+=2;
311                 }
312             }
313             catch(const std::exception& e)
314             {
315                 //when the calculation fails f.e. division by 0
316                 output_str.clear();
317                 output_str.assign(e.what());
318                 return 1;
319             }
320         }
321
322     }
323
324     try
325     {
326         //the final result is calculated by simulatinf the press of '='
327         add_to_calculation('=', ' ');
328     }
329     catch(const std::exception &e)
330     {
331         output_str.assign(e.what());
332         return 1;
333     }
334
```

```
335     //the double result is converted to text, which is displayed on the calculator screen
336     output_str.assign(std::to_string(result));
337     //the trailing 0 are stripped
338     output_str.erase(output_str.find_last_not_of('0') + 1, std::string::npos);
339     //if the number wasn't floating point number the decimal dot is stripped
340     if (output_str.back() == '.')
341         output_str.pop_back();
342 
343     return 0;
344 }
```

5.4.4.2 calculate() [2/2]

```
int Logic::calculate (
              std::string input_str,
              long double & output_result )  [related]
```

Parses the input string to numbers and operators and uses.

to calculate the final result

Parameters

input_str	String with the equation that needs to be calculated, this string has to be parsed, so it has mathematical sence, otherwise the behaviour is undefined
output_result	variable where the final result will be stored

Returns

Returns 0 on success otherwise 1 and std::runtime_exception is thrown

Definition at line 346 of file logic.cpp.

```
347 {
348     //adding R resets the current calculation
349     add_to_calculation('R', ' ');
350     std::size_t increase;
351     for (std::size_t i = 0; i < input_str.size(); i++)
352     {
353         //the test can be converted to a number if this conditions is true
354         if ((input_str[i] >= '0' && input_str[i] <= '9') || (i == 0 && input_str.size() > 1 &&
355             input_str[i] == '-' && IS_DIGIT(input_str[1])) || ( input_str[i] == '-' &&
356           (input_str[i-1] == '+' || input_str[i-1] == '-' || input_str[i-1] == '*' || input_str[i-1] ==
    '/' ||
357            input_str[i-1] == '^' || input_str[i-1] == '?' ||  input_str[i-1] == '%' || input_str[i-1] ==
    '(')
358           && i != input_str.size() - 1 && (i + 1 < input_str.size() && input_str[i+1] != '(')))
359         {
360             currently_entered_num = std::stold(&input_str[i], &increase);
361             i += increase; //the iterator is increased for the lenght of the converted number
362         }
363         if (i < input_str.size())
364         {
365             try
366             {
367                 //the first operator is added
368                 add_to_calculation(input_str[i], (i > 0)?input_str[i-1]:0);
369 
370                 //if there are multiple operators, they are also added
371                 while (i + 2 < input_str.size() && !IS_DIGIT(input_str[i + 1]) && !IS_DIGIT(input_str[i
    + 2]))
372                 {
373                     if (input_str[++i] == '-' && (IS_OPERATOR_NF(input_str[i-1])))
374                         add_to_calculation('M', ' ');
375                     else
376                         add_to_calculation(input_str[i], input_str[i - 1]);
377 
378                     if(input_str[i] == 's' || input_str[i] == 'c' || input_str[i] == 't')
```

```
379                         i+=2;
380                     }
381                 }
382                 catch(const std::exception& e)
383                 {
384                     // if the calculation fails, the result is set to NAN
385                     output_result = NAN;
386                     return 1;
387                 }
388             }
389 
390         }
391         try
392         {
393             //simulates pressing equal button
394             add_to_calculation('=', ' ');
395         }
396         catch(const std::exception &e)
397         {
398             output_result = NAN;
399             return 1;
400         }
401 
402         //the result is loaded to the output_result
403         output_result = result;
404 
405         return 0;
406 }
```

The documentation for this class was generated from the following files:

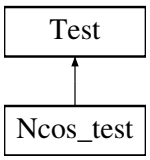
- include/logic.h
- source/logic.cpp

5.5 Ncos_test Class Reference

class that tests cosx calculation

`#include <tests.h>`

Inheritance diagram for Ncos_test:



Protected Member Functions

- bool cosx_test (long double val, long double expected_result, bool fail_expected)

  function that compares calculation of cosx with expected result given user

5.5.1 Detailed Description

class that tests cosx calculation

Definition at line 207 of file tests.h.

5.5.2 Member Function Documentation

5.5.2.1 cosx_test()

```
bool Ncos_test::cosx_test (
            long double val,
            long double expected_result,
            bool fail_expected )  [protected]
```

function that compares calculation of cosx with expected result given user

Precondition

expected result has to be accurate to 5 decimal digits
cosx expects value in degrees

Parameters

val	number from which the cosx is calculated (given in degrees)
expected_result	number, that is expected to be correct result of calculation
fail_expected	boolean, set to true if fail of calculation is expected (values where function does not exist), else set to to false

Returns

true on success - expected_result matches the result of calculator or calculation fails while fail_expected is set to true, otherwise false is returned

Definition at line 166 of file test_mock_stub.cpp.

```
167 {
168     if (expected_result > 1 || expected_result < -1)
169         return fail_expected;
170 
171     long double cosx_result;
172     try
173     {
174         cosx_result = cosx(val);
175         if (cosx_result >= expected_result - TEST_EPS && cosx_result <= expected_result + TEST_EPS)
176                 return 1;
177         else
178         {
179                 fprintf(stderr, "%Lf\n", cosx_result);
180                 return 0;
181         }
182     }
183     catch(const std::exception& e)
184     {
185         return fail_expected;
186     }
187 }
```

The documentation for this class was generated from the following files:

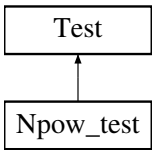
- include/tests.h
- source/test_mock_stub.cpp

5.6 Npow_test Class Reference

class that tests the N-power calculation(N - positive integer)

`#include <tests.h>`

Inheritance diagram for Npow_test:



Protected Member Functions

- bool pow_test (long double num, long unsigned exp, long double expected_result, bool fail_expected)
  function that compares calculation of N-power with expected result

5.6.1 Detailed Description

class that tests the N-power calculation(N - positive integer)

Definition at line 147 of file tests.h.

5.6.2 Member Function Documentation

5.6.2.1 pow_test()

```
bool Npow_test::pow_test (
            long double num,
            long unsigned exp,
            long double expected_result,
            bool fail_expected )  [protected]
```

function that compares calculation of N-power with expected result

Precondition

expected result has to be accurate to 5 decimal digits

Parameters

num	number from which the N-power is calculated
exp	exponent of num variable, must be positive integer
expected_result	number, that is expected to be correct result of calculation
fail_expected	boolean, set to true if fail of calc is expected (too large number), else set to to false

Returns

true on success - expected_result matches the result of calculator or calculation fails while fail_expected is set to true, otherwise false is returned

Definition at line 102 of file test_mock_stub.cpp.

```
103 {
104     long double npow_result;
105     try
106     {
107         npow_result = pow(num, exp);
108         if (npow_result >= expected_result - TEST_EPS && npow_result <= expected_result + TEST_EPS)
109             return 1;
110         else
111         {
112             fprintf(stderr, "%Lf\n", npow_result);
113             return 0;
114         }
115     }
116     catch(const std::exception& e)
117     {
118         return fail_expected;
119     }
120 }
```

The documentation for this class was generated from the following files:

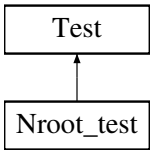
- include/tests.h
- source/test_mock_stub.cpp

5.7 Nroot_test Class Reference

class that tests the Nroot function

`#include <tests.h>`

Inheritance diagram for Nroot_test:



Protected Member Functions

- bool root_test (long double x, unsigned exponent, long double expected_result, bool fail_expected)

  function used in tests to test the nroot function from our_math library

5.7.1 Detailed Description

class that tests the Nroot function

Definition at line 72 of file tests.h.

5.7.2 Member Function Documentation

5.7.2.1 root_test()

```
bool Nroot_test::root_test (
            long double x,
            unsigned exponent,
            long double expected_result,
            bool fail_expected )  [protected]
```

function used in tests to test the nroot function from our_math library

Parameters

x	base, which is rooted @powerd_to the exponent of the root @expected_result result which should be calculated by the nroot function @fail_expected true when the tester expects the nroot function to fail (i.e. negative number is rooted to an even exponent)

Returns

true on success (result matches the expected_result or when fail is expected and function throws an exception), otherwise false

Definition at line 12 of file test_mock_stub.cpp.

```
13 {
14     long double nroot_result;
15     try
16     {
17         nroot_result = nroot(x, exponent);
18         if (nroot_result >= expected_result - TEST_EPS && nroot_result <= expected_result + TEST_EPS)
19             return 1;
20         else
21         {
22             fprintf(stderr, "%Lf\n", nroot_result);
23             return 0;
24         }
25     }
26     catch(const std::exception& e)
27     {
28         return fail_expected;
29     }
30 }
```

The documentation for this class was generated from the following files:

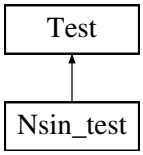
- include/tests.h
- source/test_mock_stub.cpp

5.8 Nsin_test Class Reference

class that tests sinx calculation

```
#include <tests.h>
```

Inheritance diagram for Nsin_test:



Protected Member Functions

- bool sinx_test (double val, long double expected_result, bool fail_expected)

  function that compares calculation of sinx with expected result

5.8.1 Detailed Description

class that tests sinx calculation

Definition at line 187 of file tests.h.

5.8.2 Member Function Documentation

5.8.2.1 sinx_test()

```
bool Nsin_test::sinx_test (
            double val,
            long double expected_result,
            bool fail_expected )  [protected]
```

function that compares calculation of sinx with expected result

Precondition

expected result has to be accurate to 5 decimal digits
sinx expects value in degrees

Parameters

val	number from which the sinx is calculated (given in degrees)
expected_result	number, that is expected to be correct result of calculation
fail_expected	boolean, set to true if fail of calculation is expected (values where function does not exist), else set to to false

Returns

true on success - expected_result matches the result of calculator or calculation fails while fail_expected is set to true, otherwise false is returned

Definition at line 143 of file test_mock_stub.cpp.

```
144 {
145     if (expected_result > 1 || expected_result < -1)
146         return fail_expected;
147 
148     long double sinx_result;
149     try
150     {
151         sinx_result = sinx(val);
152         if (sinx_result >= expected_result - TEST_EPS && sinx_result <= expected_result + TEST_EPS)
153                 return 1;
154         else
155         {
156                 fprintf(stderr, "%Lf\n", sinx_result);
157                 return 0;
158         }
159     }
160     catch(const std::exception& e)
161     {
162         return fail_expected;
163     }
164 }
```

The documentation for this class was generated from the following files:

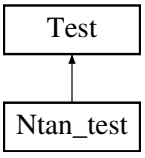
- include/tests.h
- source/test_mock_stub.cpp

5.9 Ntan_test Class Reference

class that tests tanx calculation

`#include <tests.h>`

Inheritance diagram for Ntan_test:



Protected Member Functions

- bool tanx_test (double val, long double expected_result, bool fail_expected)

  function that compares calculation of tanx with expected result given user, uses both sinx and cosx

5.9.1 Detailed Description

class that tests tanx calculation

Definition at line 227 of file tests.h.

5.9.2 Member Function Documentation

5.9.2.1 tanx_test()

```
bool Ntan_test::tanx_test (
            double val,
            long double expected_result,
            bool fail_expected )  [protected]
```

function that compares calculation of tanx with expected result given user, uses both sinx and cosx

Precondition

expected result has to be accurate to 5 decimal digits
tanx expects value in degrees

Parameters

val	number from which the tanx is calculated (given in degrees)
expected_result	number, that is expected to be correct result of calculation
fail_expected	boolean, set to true if fail of calculation is expected (values where function does not exist), else set to to false

Returns

true on success - expected_result matches the result of calculator or calculation fails while fail_expected is set to true, otherwise false is returned

Definition at line 189 of file test_mock_stub.cpp.

```
190 {
191     if (cos(val) == 0)
192         return fail_expected;
193
194     long double tanx_result;
195     try
196     {
197         tanx_result = tanx(val);
198         if (tanx_result >= expected_result - TEST_EPS && tanx_result <= expected_result + TEST_EPS)
199                 return 1;
200         else
201         {
202                 fprintf(stderr, "%Lf\n", tanx_result);
203                 return 0;
204         }
205     }
206     catch(const std::exception& e)
207     {
208         return fail_expected;
209     }
210 }
```

The documentation for this class was generated from the following files:

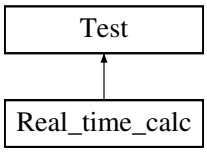
- include/tests.h
- source/test_mock_stub.cpp

5.10 Real_time_calc Class Reference

class that tests the calculations of the real time calculator

`#include <tests.h>`

Inheritance diagram for Real_time_calc:



Protected Member Functions

- void test_real_time_calculation (std::string calculation_string, std::vector< unsigned > comparison_offsets, std::vector< long double > intercount_results)

  function gets an equation as an input and then tests the correctness of results at certain positions

Protected Attributes

- Logic test
- std::string result

5.10.1 Detailed Description

class that tests the calculations of the real time calculator

Definition at line 108 of file tests.h.

5.10.2 Member Function Documentation

5.10.2.1 test_real_time_calculation()

```
void Real_time_calc::test_real_time_calculation (
            std::string calculation_string,
            std::vector< unsigned > comparison_offsets,
            std::vector< long double > intercount_results )  [protected]
```

function gets an equation as an input and then tests the correctness of results at certain positions

function that adds to the real-time calculation characters from string by one and checks at given ofsets, if the intercount result is correct

Parameters

calculation_string	equation that is being calculated
comparison_offsets	positions in equation, where we want to know the result, indexing starts from zero
intercount_results	results, that are expected at positions given by comparison_offsets

Returns

true on success - expected results matche the results of calculator, otherwise false is returned

Parameters

calculation_string	an equation that will be parsed, so it can contain random characters
comparison_offsets	vector with the possitions where, the intercout result is going to be checked (the index of the string where you want to check the intercount result)
intercount_results	vector with results to be checked

Definition at line 50 of file test_mock_stub.cpp.

```
52 {
53     std::string dummy;
54     if (calculation_string.size() == 0 || comparison_offsets.size() == 0 || intercount_results.size() ==
       0)
55     {
56         fprintf(stderr, "Invalid parameters\n");
57         return;
58     }
59
60     test.real_time_calculation('C', result, dummy);
61
62     for (unsigned i = 0, j = 0, k = comparison_offsets[j]; i < calculation_string.size(); i++)
63     {
64         test.real_time_calculation(calculation_string[i], result, dummy);
65         if (intercount_results.size() > j && k- == 0)
66         {
67             if (!(std::stold(result) >= (intercount_results[j] - TEST_EPS) &&
68                         std::stold(result) <= (intercount_results[j] + TEST_EPS)))
69             {
70                 fprintf(stderr, "False result at position: %u. Value is: %s, expected: %Lf\n",
       comparison_offsets[j],
71                         result.c_str(), intercount_results[j]);
72                 EXPECT_TRUE(false);
73             }
74             if (comparison_offsets.size() > j + 1)
75                 k = comparison_offsets[j+1] - comparison_offsets[j] - 1;
76
77             j++;
78         }
79     }
80 }
```

5.10.3 Member Data Documentation

5.10.3.1 result

```
std::string Real_time_calc::result  [protected]
```

Definition at line 112 of file tests.h.

5.10.3.2 test

```
Logic Real_time_calc::test  [protected]
```

Definition at line 111 of file tests.h.

The documentation for this class was generated from the following files:

- include/tests.h
- source/test_mock_stub.cpp

5.11 Stddev_function Class Reference

Class containing all the functions necessary for running the stddev program.

```
#include <logic.h>
```

Public Member Functions

- long double * get_data_pointer ()

  Gets the pointer to an array.
- long double calculate_stddev ()

  Calculates sample standard deviation from the given data.
- Stddev_function ()

  Constructor.
- ~Stddev_function ()

  Deconstructor.
- int load_function ()

  Loads all the data from a file to the data_array.

5.11.1 Detailed Description

Class containing all the functions necessary for running the stddev program.

Definition at line 300 of file logic.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Stddev_function()

```
Stddev_function::Stddev_function ( )
```

Constructor.

Tries to allocate space for 20 long double values, then sets total_addition and loaded_nums to zero

Definition at line 1999 of file logic.cpp.

```
2000 {
2001     data_array = (long double*)malloc(BLOCK_SIZE*sizeof(long double));;
2002     total_addition = 0;
2003     loaded_nums = 0;
2004 }
```

5.11.2.2 ~Stddev_function()

```
Stddev_function::~Stddev_function ( )
```

Deconstructor.

Frees all the allocated memory and set the data_array pointer to NULL

Definition at line 2006 of file logic.cpp.

```
2007 {
2008     delete(data_array);
2009     data_array = NULL;
2010 }
```

5.11.3 Member Function Documentation

5.11.3.1 calculate_stddev()

```
long double Stddev_function::calculate_stddev ( )
```

Calculates sample standard deviation from the given data.

Returns

Sample standard deviation is returned

Definition at line 2060 of file logic.cpp.

```
2061 {
2062     return nroot(get_summation()/(loaded_nums-1),2);
2063 }
```

5.11.3.2 get_data_pointer()

```
long double * Stddev_function::get_data_pointer ( )
```

Gets the pointer to an array.

Returns

NULL is returned if nothing was allocated, else pointer to dynamic array is returned

Definition at line 2064 of file logic.cpp.

```
2065 {
2066     return data_array;
2067 }
```

5.11.3.3 load_function()

```
int Stddev_function::load_function ( )
```

Loads all the data from a file to the data_array.

Returns

-1 is returned if loading fails, 1 is returned when functions ends successfully

Definition at line 2012 of file logic.cpp.

```
2013 {
2014     short int success_load = 0;
2015     unsigned block_count = 1;
2016     while((success_load = scanf("%Lf", &data_array[loaded_nums])) != EOF && success_load != 0)
       //loading data til EOF or error
2017     {
2018         if(loaded_nums == block_count*BLOCK_SIZE-1)
2019         {
2020             block_count++;
2021             long double *resize =(long double*) realloc(data_array,block_count*BLOCK_SIZE*sizeof(long
       double)); //increasing the size of allocated memory
2022             if(resize != NULL)
2023                 data_array = resize;
2024             else
2025                 return -1;
2026
2027         }
2028         total_addition += data_array[loaded_nums];
2029         loaded_nums++;
2030     }
2031
2032     if (loaded_nums < 2)
2033     {
2034         std::cerr « "stddev: Enter at least two numbers." « std::endl;
2035         return -1;
2036     }
2037
2038     if(!success_load)
2039         return -1;
2040     else
2041         return 1;
2042 }
```

The documentation for this class was generated from the following files:

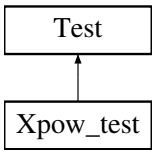
- include/logic.h
- source/logic.cpp

5.12 Xpow_test Class Reference

class that tests the X-power calculation(X - long double)

`#include <tests.h>`

Inheritance diagram for Xpow_test:



Protected Member Functions

- bool pow_test (long double num, long double exp, long double expected_result, bool fail_expected)
  function that compares calculation of X-power with expected result

5.12.1 Detailed Description

class that tests the X-power calculation(X - long double)

Definition at line 167 of file tests.h.

5.12.2 Member Function Documentation

5.12.2.1 pow_test()

```
bool Xpow_test::pow_test (
            long double num,
            long double exp,
            long double expected_result,
            bool fail_expected )  [protected]
```

function that compares calculation of X-power with expected result

Precondition

expected result has to be accurate to 5 decimal digits

Parameters

num	number from which the X-power is calculated
exp	exponent of num variable
expected_result	number, that is expected to be correct result of calculation
fail_expected	boolean, set to true if fail of calculation is expected (too large number), else set to to false

Returns

true on success - expected_result matches the result of calculator or calculation fails while fail_expected is set to true, otherwise false is returned

Definition at line 122 of file test_mock_stub.cpp.

```
123 {
124     long double npow_result;
125     try
126     {
127         npow_result = xpow(num, exp);
128         if (npow_result >= expected_result - TEST_EPS && npow_result <= expected_result + TEST_EPS)
129             return 1;
130         else
131         {
132             fprintf(stderr, "%Lf\n", npow_result);
133             return 0;
134         }
135     }
136     catch(const std::exception& e)
137     {
138         return fail_expected;
139     }
140 }
```

The documentation for this class was generated from the following files:

- include/tests.h
- source/test_mock_stub.cpp

Chapter 6

File Documentation

6.1 include/graphics.h File Reference

Header file consisting declarations of the graphics user interface functions and variables.

```
#include <gtk/gtk.h>
#include <unistd.h>
#include "logic.h"
```

Classes

- class GUI

  This class includes all the widgets and functions that are needed for initialization of graphics.

Macros

- #define NUM_OF_BUTTONS 27

  the number of used buttons on the calculator (appart from the state switch button)

6.1.1 Detailed Description

Header file consisting declarations of the graphics user interface functions and variables.

This header file contains declaration of main graphics class and all the widgets (buttons, labels)

Author

Mihola David
Sokolovskii Vladislav

Date

27.02.2019

Copyright

6.1.2 Macro Definition Documentation

6.1.2.1 NUM_OF_BUTTONS

```
#define NUM_OF_BUTTONS 27
```

the number of used buttons on the calculator (appart from the state switch button)

Number of all the buttons

Definition at line 22 of file graphics.h.

6.2 include/logic.h File Reference

Header file consisting declarations of functions and variables that creates the 'brain' of the calculator.

```
#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <functional>
#include <clocale>
#include <cmath>
#include <locale>
#include "our_math.h"
```

Classes

- class Logic

  class containing all declaration needed to calculate complex equations, on which either GUI can be added or can be run from the terminal
- class Stddev_function

  Class containing all the functions necessary for running the stddev program.

Macros

- #define BLOCK_SIZE 20

  the size of one block of allocated data
- #define EPS 1e-10

  the deviation of a correct result
- #define IS_OPERATOR(char_value)

  determines wheather the input character is an operator or not @char_value the character to be checked
- #define IS_OPERATOR_NF(char_value)

  determines wheather the input character is an operator or not, '!' is not considered as an operator @char_value the character to be checked
- #define IS_DIGIT(char_value) (char_value >= '0' && char_value <= '9')

  determines wheather the input character is a digit or not @char_value the character to be checked

Enumerations

- enum character_input_states {
  UNDEFINED = 0, L1, L2, L3,
  L1L2, L1L3, L2L3, L1L2L3,
  DIGIT, L1_GONIO, L2_GONIO, L3_GONIO,
  GONIO_NEGATE, L1L2_GONIO, L1L3_GONIO, L2L3_GONIO,
  L1L2L3_GONIO, GONIO, REDUCTION_BY3, GONIO_ADDITION,
  RESET, DECIMAL, PROCCEDE, START_STATE,
  REDUCTION_BY1, REDUCTION_BY2, ADDITION, NEGATE,
  NEGATION, OPENED_BRACKET, CLOSED_BRACKET, EQUAL,
  FACTORIAL, DECIMAL_DOT, CLEAR, GONIO_PROCCEDE }
- enum string_control {
  INTEGER, DECIMAL_NUM, OPERATOR, NO_STATE,
  O_BRACKET, C_BRACKET, GONIOOMETRY }

  enumeration used to determin different states in which the equation string can appear

6.2.1 Detailed Description

Header file consisting declarations of functions and variables that creates the 'brain' of the calculator.

Author

Mihola David
Sokolovskii Vladislav
Foltyn Lukas

Date

27.02.2019

Copyright

©Pied Piper

6.2.2 Macro Definition Documentation

6.2.2.1 BLOCK_SIZE

```
#define BLOCK_SIZE 20
```

the size of one block of allocated data

Definition at line 29 of file logic.h.

6.2.2.2 EPS

```
#define EPS 1e-10
```

the deviation of a correct result

Definition at line 35 of file logic.h.

6.2.2.3 IS_DIGIT

```
#define IS_DIGIT(
              char_value ) (char_value >= '0' && char_value <= '9')
```

determines wheather the input character is a digit or not @char_value the character to be checked

Definition at line 57 of file logic.h.

6.2.2.4 IS_OPERATOR

```
#define IS_OPERATOR(
              char_value )
```

Value:

```
(char_value == '+' || char_value == '-' || char_value == '*' || char_value == '/' ||\
                              char_value == '^' || char_value == '?' || char_value == '!' || char_value
      == '%')
```

determines wheather the input character is an operator or not @char_value the character to be checked

Definition at line 42 of file logic.h.

6.2.2.5 IS_OPERATOR_NF

```
#define IS_OPERATOR_NF(
              char_value )
```

Value:

```
(char_value == '+' || char_value == '-' || char_value == '*' || char_value == '/' ||\
                              char_value == '^' || char_value == '?' || char_value == '%')
```

determines wheather the input character is an operator or not, '!' is not considered as an operator @char_value the character to be checked

Definition at line 50 of file logic.h.

6.2.3 Enumeration Type Documentation

6.2.3.1 character_input_states

```
enum character_input_states
```

Enumerator

UNDEFINED	
L1	Level 1 operator
L2	Level 2 operator
L3	Level 3 operator
L1L2	Combination of level 1 operator followed by level 2 operator
L1L3	Combination of level 1 operator followed by level 3 operator
L2L3	Combination of level 2 operator followed by level 3 operator
L1L2L3	Combination of level 1, level 2 and level 3 operators followed respectivly
DIGIT	Digit was entered (i.e. '0', ..., '9' characters)
L1_GONIO	Combination of level 1 operator followed by goniometric operator
L2_GONIO	Combination of level 2 operator followed by goniometric operator
L3_GONIO	Combination of level 3 operator followed by goniometric operator
GONIO_NEGATE	Goniometric operator with a negation in front
L1L2_GONIO	Combination of level 1, level 2 and goniometric operators followed respectivly
L1L3_GONIO	Combination of level 1, level 3 and goniometric operators followed respectivly
L2L3_GONIO	Combination of level 2, level 3 and goniometric operators followed respectivly
L1L2L3_GONIO	Combination of level 1, level 2, level 3 and goniometric operators followed respectivly
GONIO	Goniometric operator on its own
REDUCTION_BY3	Reduction of 3 operators in the operator stack
GONIO_ADDITION	When goniometric function is added
RESET	For reseting the equation
DECIMAL	Digit was entered to a floating point number
PROCCEDE	Operator that does not change the operator arrangement, calculation is procceded
START_STATE	State, in which is the calculation, when it starts or a new bracket is open
REDUCTION_BY1	Reduction of 1 operator in the operator stack
REDUCTION_BY2	Reduction of 2 operators in the operator stack
ADDITION	New operator is added to the operator stack
NEGATE	Input number negation is engaged (i.e. the number is negative)
NEGATION	'N' was added negation of a currently entered number is performed
OPENED_BRACKET	'(' character was added, new bracket is opened
CLOSED_BRACKET	')' character was added, currently opened bracket is closed
EQUAL	'=' character was added, result is calculated
FACTORIAL	'!' character was added to the equation, which invokes the calculation of factorial
DECIMAL_DOT	'.' charcter was added to the equation, which invokes, that floating point nuber is added
CLEAR	'C' character was added, equation cleared and reseted
GONIO_PROCCEDE	Operator that does not change the operator arrangement added after the goniometric operator, calculation is procceded

Definition at line 67 of file logic.h.

```
67                          {
68                              UNDEFINED = 0 ,
69                              L1 ,
70                      L2 , L3 ,
71                      L1L2 ,
72                      L1L3 ,
73                      L2L3 ,
74                      L1L2L3 ,
75                      DIGIT ,
76                      L1_GONIO ,
77                      L2_GONIO ,
78                      L3_GONIO ,
```

```
79                  GONIO_NEGATE ,
80                              L1L2_GONIO ,
81                  L1L3_GONIO ,
82                  L2L3_GONIO ,
83                  L1L2L3_GONIO ,
84                  GONIO ,
85                  REDUCTION_BY3 ,
86                  GONIO_ADDITION ,
87                  RESET ,
88                              DECIMAL ,
89                  PROCCEDE ,
90                  START_STATE ,
91                  REDUCTION_BY1 ,
92                  REDUCTION_BY2 ,
93                  ADDITION ,
94                  NEGATE ,
95                  NEGATION ,
96                              OPENED_BRACKET ,
97                  CLOSED_BRACKET ,
98                  EQUAL ,
99                  FACTORIAL ,
100                  DECIMAL_DOT ,
101                  CLEAR ,
102                  GONIO_PROCCEDE
103 };
```

6.2.3.2 string_control

```
enum string_control
```

enumeration used to determin different states in which the equation string can appear

Enumerator

INTEGER	Digit was entered (i.e. '0', ..., '9' characters)
DECIMAL_NUM	Digit was added (i.e. '0', ..., '9' characters) to a floating point
OPERATOR	Operator was added ('+', '-', '*', '/', '', '^', '?')
NO_STATE	First enter to the function
O_BRACKET	'(' character was entered
C_BRACKET	')' character was enterd
GONIOOMETRY	Goniometric operator was entered ('s' - for sinus, 'c' - for cosinus, 't' - for tangens)

Definition at line 109 of file logic.h.

```
109                  {
110              INTEGER ,
111              DECIMAL_NUM ,
112              OPERATOR ,
113              NO_STATE ,
114              O_BRACKET ,
115              C_BRACKET ,
116              GONIOOMETRY
117 };
```

6.3 include/our_math.h File Reference

Header file containing definition of inline functions for basic arithmetics like addition, subtraction, division, multiplication as well as more advanced functions for calculating factorial, n-power, n-root, cosx, sinx, tanx. Also contains a declaration of xpow function.

```
#include <cmath>
#include <math.h>
```

```
#include <functional>
#include <stdexcept>
```

Macros

- #define XPOW_EPS 1e-4

  deviation for x-power functions to calculate decimal part as well as fraction part
- #define GONIO_EPS 1e-5

  deviation for goniometric functions
- #define ACCURACY_SIN 26

  limit for loop in sinx fucntion, where cosx is accurate in 5 decimal places at minimum
- #define ACCURACY_COS 25

  limit for loop in cosx function, where cosx is accurate in 5 decimal places at minimum
- #define PI 3.14159265358979323846

  constant for calculatin sinx, cosx and tanx
- #define EPS 1e-10

  deviation for npower, xpower, nroot to get quite accurate result

Functions

- long double add (long double x, long double y)

  calculates addition of given numbers
- long double sub (long double x, long double y)

  calculates subtraction of given numbers
- long double mul (long double x, long double y)

  claculates multiplication of given numbers
- long double div_ (long double x, long double y)

  calculates division of given numbers
- long double factorial (int n)

  calculates a factorial of input number
- long double npow (long double x, long unsigned n)

  calculates the power of input number, the number can be powerd only to a natural number
- long double nroot (long double x, long unsigned n)

  calculates the nth-root of input number, the number can be rooted only to a natural number
- long double xpow (long double base, long double exponent)

  calculates calculates power of a floating point number powered by floating point number (also works as nroot)
- long double sinx (double x)

  calculates sinus value of input number given in degrees
- long double cosx (double x)

  calculates cosinus value of input number given in degrees
- long double tanx (double x)

  calculates tangens value of input number given in degrees

6.3.1 Detailed Description

Header file containing definition of inline functions for basic arithmetics like addition, subtraction, division, multiplication as well as more advanced functions for calculating factorial, n-power, n-root, cosx, sinx, tanx. Also contains a declaration of xpow function.

Author

Mihola David
Foltyn Lukas

Date

27.02.2019

Copyright

6.3.2 Macro Definition Documentation

6.3.2.1 ACCURACY_COS

```
#define ACCURACY_COS 25
```

limit for loop in cosx function, where cosx is accurate in 5 decimal places at minimum

Definition at line 42 of file our_math.h.

6.3.2.2 ACCURACY_SIN

```
#define ACCURACY_SIN 26
```

limit for loop in sinx fucntion, where cosx is accurate in 5 decimal places at minimum

Definition at line 35 of file our_math.h.

6.3.2.3 EPS

```
#define EPS 1e-10
```

deviation for npower, xpower, nroot to get quite accurate result

Definition at line 54 of file our_math.h.

6.3.2.4 GONIO_EPS

```
#define GONIO_EPS 1e-5
```

deviation for goniometric functions

Definition at line 28 of file our_math.h.

6.3.2.5 PI

```
#define PI 3.14159265358979323846
```

constant for calculatin sinx, cosx and tanx

Definition at line 48 of file our_math.h.

6.3.2.6 XPOW_EPS

```
#define XPOW_EPS 1e-4
```

deviation for x-power functions to calculate decimal part as well as fraction part

Definition at line 22 of file our_math.h.

6.3.3 Function Documentation

6.3.3.1 add()

```
long double add (
            long double x,
            long double y )  [inline]
```

calculates addition of given numbers

Parameters

x	first number for addition
y	second number for addition

Returns

addition of given numbers x,y

Definition at line 62 of file our_math.h.

```
63 {
64     return x + y;
65 }
```

6.3.3.2 cosx()

```
long double cosx (
            double x )  [inline]
```

calculates cosinus value of input number given in degrees

Parameters

x	angle in degrees

Returns

the numerical value of angle

Definition at line 250 of file our_math.h.

```
251 {
252     while(x>=360)
253         x-=360;
254     while(x<=-360)
255         x+=360;
256
257     long test = (long)x;
258     if (test == 0)
259         return 1;
260     else if (test == 180)
261         return -1;
262     else if (test == 90 || test == 270)
263         return 0;
264
265     double radians=x*PI/180;
266     long double result,previous_result,temp;
267     result=temp=1;
268     for(unsigned i = 1; i < ACCURACY_COS; i += 2)
269     {
270         temp=temp*(-1)*radians*radians/(i*(i+1));
271         previous_result=result;
272         result+=temp;
273         if(fabs(result-previous_result)<GONIO_EPS)
274             break;
275     }
276     return result;
277 }
```

6.3.3.3 div_()

```
long double div_ (
            long double x,
            long double y )  [inline]
```

calculates division of given numbers

Parameters

x	number that is divided
y	number that divides

Returns

divison of given numbers x,y

Definition at line 93 of file our_math.h.

```
94 {
95     if (y == 0)
96         throw std::runtime_error("Division by 0");
97     return x / y;
98 }
```

6.3.3.4 factorial()

```
long double factorial (
             int n )  [inline]
```

calculates a factorial of input number

Parameters

n	number to calculate factorial from

Returns

factorial of the input number

Definition at line 105 of file our_math.h.

```
106 {
107     if (n < 0)
108         throw std::runtime_error("Factorial of negative number.");
109
110     if (n == 0 || n == 1 )
111         return 1;
112
113     //if (n > 20)
114       //  throw std::runtime_error("Too large number for factorial.");
115
116     long double x = (long double)n;
117     for (unsigned i = n; i > 1; --i, x*= i);
118
119     return x;
120 }
```

6.3.3.5 mul()

```
long double mul (
             long double x,
             long double y )  [inline]
```

claculates multiplication of given numbers

Parameters

x	first number for multiplication
y	second number for multiplication

Returns

multiplication of given numbers x,y

Definition at line 83 of file our_math.h.

```
84 {
85     return x * y;
86 }
```

6.3.3.6 npow()

```
long double npow (
            long double x,
            long unsigned n )  [inline]
```

calculates the power of input number, the number can be powerd only to a natural number

Parameters

x	number to be powered
n	the exponent

Returns

number powered by the exponent

Definition at line 128 of file our_math.h.

```
129 {
130     if (n == 0)
131         return 1;
132     long double hold = x;
133     if (x >= 1 - EPS && x <= 1 + EPS)
134         return x;
135     while (-n) x *= hold;
136 
137     return x;
138 }
```

6.3.3.7 nroot()

```
long double nroot (
            long double x,
            long unsigned n )  [inline]
```

calculates the nth-root of input number, the number can be rooted only to a natural number

Parameters

x	number to be rooted
n	the exponent

Returns

the nth-root of a number

Definition at line 147 of file our_math.h.

```
148 {
149     if (n == 0)
150         return 1;
151     if (n == 1)
152         return x;
153     if (x >= 1 - EPS && x <= 1 + EPS)
154         return 1;
155 
156     if (x < 0 && !(n&1))
157     {
158         throw std::runtime_error("Cannot root this.");
159         return -1;
160     }
161 
162     if (x >= -EPS && x <= EPS)
163         return 0;
164 
165     //aproximation of the result
166     long double delta = x;
167     long unsigned i = 1;
168 
169     while ((delta /= 10) > 1 && ++i);
170 
171     long double guess = 1;
172     if (i / n)
173     {
174         for (long unsigned j = 0; j < i/n; j++)
175             guess *= 10;
176 
177         guess *= i % n + 1;
178     }
179     //end of approximation
180 
181     delta = 0;
182 
183     do
184     {
185         guess += delta;
186         delta = (((x/npow(guess, (n-1)))-guess)/n);
187     }
188     while(delta > EPS || delta < -EPS);
189     return guess+delta;
190 }
```

6.3.3.8 sinx()

```
long double sinx (
             double x )  [inline]
```

calculates sinus value of input number given in degrees

Parameters

x	angle in degrees

Returns

the numerical value of angle

Definition at line 205 of file our_math.h.

```
206 {
207     while(x>=360)
208         x-=360;
209     while(x<=-360)
210         x+=360;
211 
212     //values for 90, 180, 270, 360 degrees
213     long test = (long)x;
214     if (test == 90)
215         return 1;
216     else if (test == 270)
217         return -1;
218     else if (test == 0 || test == 180)
219         return 0;
220 
221     double radians=x*PI/180;
222     long double result,previous_result,temp;
223     result=temp=radians;
224     for(unsigned i=2;i<ACCURACY_SIN;i+=2)
225     {
226         temp=temp*(-1)*radians*radians/(i*(i+1));
227         previous_result=result;
228         result+=temp;
229         if(fabs(result-previous_result)<GONIO_EPS)
230             break;
231     }
232     return result;
233 }
```

6.3.3.9 sub()

```
long double sub (
              long double x,
              long double y )  [inline]
```

calculates subtraction of given numbers

Parameters

x	number being subtracted from
y	number that is subtracted

Returns

subtraction of given numbers x,y

Definition at line 73 of file our_math.h.

```
74 {
75     return x - y;
76 }
```

6.3.3.10 tanx()

```
long double tanx (
              double x )  [inline]
```

calculates tangens value of input number given in degrees

Parameters

x	angle in degrees

Returns

the numerical value of angle

Definition at line 294 of file our_math.h.

```
295 {
296     if(!(fmod(x,90)) && (fmod(x,180)))
297         return 0.0;
298     return sinx(x)/cosx(x);
299 }
```

6.3.3.11 xpow()

```
long double xpow (
            long double base,
            long double exponent )
```

calculates calculates power of a floating point number powered by floating point number (also works as nroot)

Parameters

base	the number to be powered
exponent	the exponent on which the number is powerd

Returns

result of power the base to it's exponent

Definition at line 31 of file our_math.cpp.

```
32 {
33     if (base < EPS && base > -EPS)
34         return 0;
35
36     if (exponent < EPS && exponent > -EPS)
37         return 1;
38
39     if (exponent >= 1 - EPS && exponent <= 1 + EPS)
40         return base;
41
42     //switching the values when exponent is negative
43     if (exponent < 0)
44     {
45         base = 1 / base;
46         exponent = -exponent;
47     }
48
49     //separating the number to it's natural part and decimal part
50     long natural = (long)exponent;
51     long double rest = exponent - natural;
52     long double tmp = base;
53
54     //calculating the npower of natural part
55     if (natural)
56         while(-natural)
57             base *= tmp;
58     else
59         base = 1;
```

```
60
61
62      //calculating the power of deciaml part, if the decimal part is large enough
63      if (rest > XPOW_EPS - 0.00001)
64      {
65          // number^0.xxxx.. can be caulculated as number^(top/bot) == pow(nroot(number, bot), top), where
        top/bot = 0.xxxx..,
66          long double top = 1;
67          long double bot = 2;
68          long double delta = rest - 0.5;
69
70          // approximating the fraction, f.e. 0.65 = 13/20
71          while (delta > XPOW_EPS - 0.00001 || delta < -XPOW_EPS + 0.00001)
72          {
73              if (delta > 0)
74                  top += 1;
75              else
76                  bot += 1;
77
78              delta = rest - (top / bot);
79          }
80          //calculatig the nroot
81
82          if (tmp < 0 && !((long unsigned)bot & 1))
83          {
84              throw std::runtime_error("Cannot calculate this.");
85              return -1;
86          }
87
88          long double n_res = tmp = nroot(tmp, bot);
89          //calculating the power of the rooted number
90          while (-top)
91              n_res *= tmp;
92
93          //multiplying the natural part by the decial part
94          return base * n_res;
95      }
96      else
97          return base;
98
99      return 0;
100 }
```

6.4 include/tests.h File Reference

Header file containing test classes with declarations of their test functions for logic.cpp and our_math.cpp.

```
#include "gtest/gtest.h"
#include "logic.h"
```

Classes

- class Nroot_test

  class that tests the Nroot function
- class Input_control

  class that tests the equation_string_control function
- class Real_time_calc

  class that tests the calculations of the real time calculator
- class Factorial_tests

  class that tests the factorial calculation
- class Npow_test

  class that tests the N-power calculation(N - positive integer)
- class Xpow_test

  class that tests the X-power calculation(X - long double)
- class Nsin_test

class that tests sinx calculation

- class Ncos_test

  class that tests cosx calculation

- class Ntan_test

  class that tests tanx calculation

Macros

- #define TEST_EPS 1e-3

  deviation for testing calculation from string to long double

- #define ADD_EQUATION_WITH_RESULT_STS(EQUATION_STR, RESULT_STR)

  macro that takes "string" equation which is calculated and compared with its expected "string" result given as second parameter and return true if matched, otherwise false is returned

- #define ADD_EQUATION_WITH_RESULT_STD(EQUATION_STR, RESULT_LD)

  macro that takes "string" equation which is calculated and compared with its expected long double result given as second parameter and return true if matched, otherwise false is returned

- #define NUMS (std::vector<double>)

  subsitution for vector of number(data type - double)

- #define OPS (std::vector<char>)

  substitution for vector of chars

- #define YES true

  substitution for true in test functions

- #define NO false

  substitution for false in test functions

6.4.1 Detailed Description

Header file containing test classes with declarations of their test functions for logic.cpp and our_math.cpp.

Author

Mihola David
Foltyn Lukas
Sokolovskii Vladislav

Date

27.02.2019

Copyright

6.4.2 Macro Definition Documentation

6.4.2.1 ADD_EQUATION_WITH_RESULT_STD

```
#define ADD_EQUATION_WITH_RESULT_STD(
                EQUATION_STR,
                RESULT_LD )
```

Value:

```
Logic test_logic;\
    long double result_hold;\
    test_logic.calculate(((std::string)EQUATION_STR), result_hold);\
    EXPECT_TRUE((RESULT_LD) >= result_hold - TEST_EPS && (RESULT_LD) <= result_hold + TEST_EPS);\
```

macro that takes "string" equation which is calculated and compared with its expected long double result given as second parameter and return true if matched, otherwise false is returned

Definition at line 38 of file tests.h.

6.4.2.2 ADD_EQUATION_WITH_RESULT_STS

```
#define ADD_EQUATION_WITH_RESULT_STS(
                EQUATION_STR,
                RESULT_STR )
```

Value:

```
Logic test_logic;\
    std::string result_hold;\
    test_logic.calculate(((std::string)EQUATION_STR), result_hold);\
    EXPECT_EQ(result_hold, ((std::string)RESULT_STR));
```

macro that takes "string" equation which is calculated and compared with its expected "string" result given as second parameter and return true if matched, otherwise false is returned

Definition at line 26 of file tests.h.

6.4.2.3 NO

```
#define NO false
```

substitution for false in test functions

Definition at line 66 of file tests.h.

6.4.2.4 NUMS

```
#define NUMS (std::vector<double>)
```

subsitution for vector of number(data type - double)

Definition at line 48 of file tests.h.

6.4.2.5 OPS

```
#define OPS (std::vector<char>)
```

substitution for vector of chars

Definition at line 54 of file tests.h.

6.4.2.6 TEST_EPS

```
#define TEST_EPS 1e-3
```

deviation for testing calculation from string to long double

Definition at line 18 of file tests.h.

6.4.2.7 YES

```
#define YES true
```

substitution for true in test functions

Definition at line 60 of file tests.h.

6.5 source/CMakeLists.txt File Reference

Functions

- set (CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib) set(CMAKE_LIBRARY_O↩ UTPUT_DIRECTORY $
- lib set (CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin) find_package(PkgConfig REQUIRED) include_directories($
- link_directories (${GTK3_LIBRARY_DIRS}) add_definitions($
- set (INCLUDE_DIR ../include) add_library(our_math STATIC $
- our_math h our_math cpp add_library (graphics STATIC ${INCLUDE_DIR}/graphics.h graphics.cpp ${INC↩ LUDE_DIR}/our_math.h our_math.cpp ${INCLUDE_DIR}/logic.h logic.cpp) add_library(logic STATIC $

6.5.1 Function Documentation

6.5.1.1 add_library()

```
our_math h our_math cpp add_library (
             graphics STATIC ${INCLUDE_DIR}/graphics.h graphics.cpp ${INCLUDE_DIR}/our_math.h
our_math.cpp ${INCLUDE_DIR}/logic.h logic.  cpp )
```

Definition at line 23 of file CMakeLists.txt.

```
24              {INCLUDE_DIR}/graphics.h graphics.cpp
25             ${INCLUDE_DIR}/our_math.h our_math.cpp
26             ${INCLUDE_DIR}/logic.h logic.cpp)
27 # logic.a
28 add_library(logic STATIC
29             ${INCLUDE_DIR}/our_math.h our_math.cpp
```

6.5.1.2 link_directories()

```
link_directories (
              ${GTK3_LIBRARY_DIRS} )
```

Definition at line 13 of file CMakeLists.txt.

```
13                  {GTK3_LIBRARY_DIRS})
14 add_definitions(${GTK3_CFLAGS_OTHER})
```

6.5.1.3 set() [1/3]

```
set (
             CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/ lib )
```

Definition at line 3 of file CMakeLists.txt.

```
3                                  {CMAKE_BINARY_DIR}/lib)
4 set(CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
```

6.5.1.4 set() [2/3]

```
lib set (
             CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/ bin )
```

Definition at line 5 of file CMakeLists.txt.

```
5                                  {CMAKE_BINARY_DIR}/bin)
6
7 #finding the gtk packages
8 find_package(PkgConfig REQUIRED)
9 PKG_CHECK_MODULES(GTK3 REQUIRED gtk+-3.0)
10
11 #linking with GTK3
12 include_directories(${GTK3_INCLUDE_DIRS})
```

6.5.1.5 set() [3/3]

```
set (
              INCLUDE_DIR ../ include )
```

Definition at line 16 of file CMakeLists.txt.

```
21              {INCLUDE_DIR}/our_math.h our_math.cpp)
```

6.6 source/graphics.cpp File Reference

Initializes graphics and its API, connects it to the grap.glade file and sets the styles for the calculator.

```
#include "../include/graphics.h"
```

6.6.1 Detailed Description

Initializes graphics and its API, connects it to the grap.glade file and sets the styles for the calculator.

This source file contains initialization of GUI of the calculator (based on the GTK tool kit), function for switching between two modes and function for sending the on-click signals. Also, there is a CSS section where the styles and fonts of the calculator are set.

Author

Sokolovskii Vladislav
Mihola David

Copyright

©Pied Piper

6.7 source/logic.cpp File Reference

Source file containing all the the definitions of functions declared in logic.h, serves as a "brain" of the calculator.It is connected to all math functions in the library and put them to use for solving advanced mathematical equations.

```
#include "../include/logic.h"
```

6.7.1 Detailed Description

Source file containing all the the definitions of functions declared in logic.h, serves as a "brain" of the calculator.It is connected to all math functions in the library and put them to use for solving advanced mathematical equations.

Author

Mihola David
Foltyn Lukas

Copyright

©Pied Piper

6.8 source/main.cpp File Reference

Main function of the calculator, which just runs the graphics library.

```
#include "../include/graphics.h"
#include <gtk/gtk.h>
```

Functions

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

  the main function of the calculator

6.8.1 Detailed Description

Main function of the calculator, which just runs the graphics library.

Author

Mihola David
Sokolovskii Vladislav

Date

27.02.2019

Copyright

©Pied Piper

6.8.2 Function Documentation

6.8.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

the main function of the calculator

Definition at line 15 of file main.cpp.

```
16 {
17
18     GUI::init_graphics(argc, argv);
19
20     return 0;
21 }
```

6.9 source/our_math.cpp File Reference

This source file contains extern declarations of inline functions and definition of xpow function from the math library.

```
#include "../include/our_math.h"
```

Functions

- long double add (long double x, long double y)
  calculates addition of given numbers
- long double sub (long double x, long double y)
  calculates subtraction of given numbers
- long double mul (long double x, long double y)
  claculates multiplication of given numbers
- long double div_ (long double x, long double y)
  calculates division of given numbers
- long double cosx (double x)
  calculates cosinus value of input number given in degrees
- long double sinx (double x)
  calculates sinus value of input number given in degrees
- long double tanx (double x)
  calculates tangens value of input number given in degrees
- long double npow (long double x, long unsigned n)
  calculates the power of input number, the number can be powerd only to a natural number
- long double nroot (long double x, long unsigned n)
  calculates the nth-root of input number, the number can be rooted only to a natural number
- long double factorial (int n)
  calculates a factorial of input number
- long double xpow (long double base, long double exponent)
  calculates calculates power of a floating point number powered by floating point number (also works as nroot)

6.9.1 Detailed Description

This source file contains extern declarations of inline functions and definition of xpow function from the math library.

Author

Mihola David
Foltyn Lukas

Copyright

©Pied Piper

6.9.2 Function Documentation

6.9.2.1 add()

```
long double add (
            long double x,
            long double y )  [inline]
```

calculates addition of given numbers

Parameters

x	first number for addition
y	second number for addition

Returns

addition of given numbers x,y

Definition at line 62 of file our_math.h.

```
63 {
64     return x + y;
65 }
```

6.9.2.2 cosx()

```
long double cosx (
             double x )  [inline]
```

calculates cosinus value of input number given in degrees

Parameters

x	angle in degrees

Returns

the numerical value of angle

Definition at line 250 of file our_math.h.

```
251 {
252     while(x>=360)
253         x-=360;
254     while(x<=-360)
255         x+=360;
256
257     long test = (long)x;
258     if (test == 0)
259         return 1;
260     else if (test == 180)
261         return -1;
262     else if (test == 90 || test == 270)
263         return 0;
264
265     double radians=x*PI/180;
266     long double result,previous_result,temp;
267     result=temp=1;
268     for(unsigned i = 1; i < ACCURACY_COS; i += 2)
269     {
270         temp=temp*(-1)*radians*radians/(i*(i+1));
271         previous_result=result;
272         result+=temp;
273         if(fabs(result-previous_result)<GONIO_EPS)
274             break;
275     }
276     return result;
277 }
```

6.9.2.3 div_()

```
long double div_ (
            long double x,
            long double y )  [inline]
```

calculates division of given numbers

Parameters

x	number that is divided
y	number that divides

Returns

divison of given numbers x,y

Definition at line 93 of file our_math.h.

```
94 {
95     if (y == 0)
96         throw std::runtime_error("Division by 0");
97     return x / y;
98 }
```

6.9.2.4 factorial()

```
long double factorial (
            int n )  [inline]
```

calculates a factorial of input number

Parameters

n	number to calculate factorial from

Returns

factorial of the input number

Definition at line 105 of file our_math.h.

```
106 {
107     if (n < 0)
108         throw std::runtime_error("Factorial of negative number.");
109
110     if (n == 0 || n == 1 )
111         return 1;
112
113     //if (n > 20)
114       //  throw std::runtime_error("Too large number for factorial.");
115
116     long double x = (long double)n;
117     for (unsigned i = n; i > 1; --i, x*= i);
118
119     return x;
120 }
```

6.9.2.5 mul()

```
long double mul (
              long double x,
              long double y )  [inline]
```

claculates multiplication of given numbers

Parameters

x	first number for multiplication
y	second number for multiplication

Returns

multiplication of given numbers x,y

Definition at line 83 of file our_math.h.

```
84 {
85     return x * y;
86 }
```

6.9.2.6 npow()

```
long double npow (
              long double x,
              long unsigned n )  [inline]
```

calculates the power of input number, the number can be powerd only to a natural number

Parameters

x	number to be powered
n	the exponent

Returns

number powered by the exponent

Definition at line 128 of file our_math.h.

```
129 {
130     if (n == 0)
131         return 1;
132     long double hold = x;
133     if (x >= 1 - EPS && x <= 1 + EPS)
134         return x;
135     while (-n) x *= hold;
136 
137     return x;
138 }
```

6.9.2.7 nroot()

```
long double nroot (
            long double x,
            long unsigned n )  [inline]
```

calculates the nth-root of input number, the number can be rooted only to a natural number

Parameters

x	number to be rooted
n	the exponent

Returns

the nth-root of a number

Definition at line 147 of file our_math.h.

```
148 {
149     if (n == 0)
150         return 1;
151     if (n == 1)
152         return x;
153     if (x >= 1 - EPS && x <= 1 + EPS)
154         return 1;
155 
156     if (x < 0 && !(n&1))
157     {
158         throw std::runtime_error("Cannot root this.");
159         return -1;
160     }
161 
162     if (x >= -EPS && x <= EPS)
163         return 0;
164 
165     //aproximation of the result
166     long double delta = x;
167     long unsigned i = 1;
168 
169     while ((delta /= 10) > 1 && ++i);
170 
171     long double guess = 1;
172     if (i / n)
173     {
174         for (long unsigned j = 0; j < i/n; j++)
175             guess *= 10;
176 
177         guess *= i % n + 1;
178     }
179     //end of approximation
180 
181     delta = 0;
182 
183     do
184     {
185         guess += delta;
186         delta = (((x/npow(guess, (n-1)))-guess)/n);
187     }
188     while(delta > EPS || delta < -EPS);
189     return guess+delta;
190 }
```

6.9.2.8 sinx()

```
long double sinx (
            double x )  [inline]
```

calculates sinus value of input number given in degrees

Parameters

x	angle in degrees

Returns

the numerical value of angle

Definition at line 205 of file our_math.h.

```
206 {
207     while(x>=360)
208         x-=360;
209     while(x<=-360)
210         x+=360;
211
212     //values for 90, 180, 270, 360 degrees
213     long test = (long)x;
214     if (test == 90)
215         return 1;
216     else if (test == 270)
217         return -1;
218     else if (test == 0 || test == 180)
219         return 0;
220
221     double radians=x*PI/180;
222     long double result,previous_result,temp;
223     result=temp=radians;
224     for(unsigned i=2;i<ACCURACY_SIN;i+=2)
225     {
226         temp=temp*(-1)*radians*radians/(i*(i+1));
227         previous_result=result;
228         result+=temp;
229         if(fabs(result-previous_result)<GONIO_EPS)
230             break;
231     }
232     return result;
233 }
```

6.9.2.9 sub()

```
long double sub (
              long double x,
              long double y )  [inline]
```

calculates subtraction of given numbers

Parameters

x	number being subtracted from
y	number that is subtracted

Returns

subtraction of given numbers x,y

Definition at line 73 of file our_math.h.

```
74 {
75     return x - y;
76 }
```

6.9.2.10 tanx()

```
long double tanx (
            double x )  [inline]
```

calculates tangens value of input number given in degrees

Parameters

x	angle in degrees

Returns

the numerical value of angle

Definition at line 294 of file our_math.h.

```
295 {
296     if(!(fmod(x,90)) && (fmod(x,180)))
297         return 0.0;
298     return sinx(x)/cosx(x);
299 }
```

6.9.2.11 xpow()

```
long double xpow (
             long double base,
             long double exponent )
```

calculates calculates power of a floating point number powered by floating point number (also works as nroot)

Parameters

base	the number to be powered
exponent	the exponent on which the number is powerd

Returns

result of power the base to it's exponent

Definition at line 31 of file our_math.cpp.

```
32 {
33     if (base < EPS && base > -EPS)
34         return 0;
35
36     if (exponent < EPS && exponent > -EPS)
37         return 1;
38
39     if (exponent >= 1 - EPS && exponent <= 1 + EPS)
40         return base;
41
42     //switching the values when exponent is negative
43     if (exponent < 0)
44     {
45         base = 1 / base;
46         exponent = -exponent;
47     }
```

```
48
49      //separating the number to it's natural part and decimal part
50      long natural = (long)exponent;
51      long double rest = exponent - natural;
52      long double tmp = base;
53
54      //calculating the npower of natural part
55      if (natural)
56          while(-natural)
57              base *= tmp;
58      else
59          base = 1;
60
61
62      //calculating the power of deciaml part, if the decimal part is large enough
63      if (rest > XPOW_EPS - 0.00001)
64      {
65          // number^0.xxxx.. can be caulculated as number^(top/bot) == pow(nroot(number, bot), top), where
        top/bot = 0.xxxx..,
66          long double top = 1;
67          long double bot = 2;
68          long double delta = rest - 0.5;
69
70          // approximating the fraction, f.e. 0.65 = 13/20
71          while (delta > XPOW_EPS - 0.00001 || delta < -XPOW_EPS + 0.00001)
72          {
73              if (delta > 0)
74                  top += 1;
75              else
76                  bot += 1;
77
78              delta = rest - (top / bot);
79          }
80          //calculatig the nroot
81
82          if (tmp < 0 && !((long unsigned)bot & 1))
83          {
84              throw std::runtime_error("Cannot calculate this.");
85              return -1;
86          }
87
88          long double n_res = tmp = nroot(tmp, bot);
89          //calculating the power of the rooted number
90          while (-top)
91              n_res *= tmp;
92
93          //multiplying the natural part by the decial part
94          return base * n_res;
95      }
96      else
97          return base;
98
99      return 0;
100 }
```

6.10 source/stddev.cpp File Reference

Source file, which serves for calculating sample standard deviation with help of functions from math library.

```
#include "../include/logic.h"
```

Functions

- int main ()

6.10.1 Detailed Description

Source file, which serves for calculating sample standard deviation with help of functions from math library.

Author

Foltyn Lukas
Mihola David

Copyright

©Pied Piper

6.10.2 Function Documentation

6.10.2.1 main()

```
int main ( )
```

Definition at line 11 of file stddev.cpp.

```
12 {
13     Stddev_function sample_std_deviation; // initialization with constructor
14     if(sample_std_deviation.get_data_pointer() != NULL) // testing if allocation in constructor did not
       fail
15     {
16         if(sample_std_deviation.load_function() != -1) // loading data from user given file and testing
       for error
17             std::cout « sample_std_deviation.calculate_stddev() « std::endl;
18         else
19             std::cerr « "stddev: Loading of data failed." « std::endl;
20     }
21     else
22         std::cerr « "stddev: Allocation of initial memory failed." « std::endl;
23
24     return 0;
25 }
```

6.11 source/test_mock_stub.cpp File Reference

Source file containing the definitions of test functions, that test the functioanlity of math library.

```
#include "../include/tests.h"
```

6.11.1 Detailed Description

Source file containing the definitions of test functions, that test the functioanlity of math library.

Author

Mihola David
Sokolovskii Vladislav
Foltyn Lukas

Copyright

©Pied Piper

6.12 source/tests.cpp File Reference

Source file containing tests for each part of the math library.

```
#include "../include/tests.h"
```

Functions

- TEST (Logic_test2, Calculation_str_to_str1)
- TEST (Logic_test2, Calculation_str_to_str2)
- TEST (Logic_test2, Calculation_str_to_str3)
- TEST (Logic_test2, Calculation_str_to_str4)
- TEST (Logic_test2, Calculation_str_to_str6)
- TEST (Logic_test2, Calculation_str_to_str7)
- TEST (Logic_test2, Calculation_str_to_str8)
- TEST (Logic_test2, Calculation_str_to_str9)
- TEST (Logic_test2, Calculation_str_to_str10)
- TEST (Logic_test2, Calculation_str_to_str11)
- TEST (Logic_test2, Calculation_str_to_str12)
- TEST (Logic_test2, Calculation_str_to_str13)
- TEST (Logic_test2, Calculation_str_to_str14)
- TEST (Logic_test2, Calculation_str_to_str15)
- TEST (Logic_test2, Calculation_str_to_str16)
- TEST (Logic_test2, Calculation_str_to_str17)
- TEST (Logic_test2, Calculation_str_to_str18)
- TEST (Logic_test2, Calculation_str_to_str19)
- TEST (Logic_test2, Calculation_str_to_str20)
- TEST (Logic_test2, Calculation_str_to_str21)
- TEST (Logic_test2, Calculation_str_to_str22)
- TEST (Logic_test2, Calculation_str_to_str23)
- TEST (Logic_test2, Calculation_str_to_str24)
- TEST (Logic_test2, Calculation_str_to_str25)
- TEST (Logic_test2, Calculation_str_to_str26)
- TEST (Logic_test2, Calculation_str_to_str27)
- TEST (Logic_test3, Calculation_str_to_ld1)
- TEST (Logic_test3, Calculation_str_to_ld2)
- TEST (Logic_test3, Calculation_str_to_ld3)
- TEST (Logic_test3, Calculation_str_to_ld4)
- TEST (Logic_test3, Calculation_str_to_ld5)
- TEST (Logic_test3, Calculation_str_to_ld6)
- TEST (Logic_test3, Calculation_str_to_ld7)
- TEST (Logic_test3, Calculation_str_to_ld8)
- TEST (Logic_test3, Calculation_str_to_ld9)
- TEST (Logic_test3, Calculation_str_to_ld10)
- TEST_F (Input_control, Input_parsing1)
- TEST_F (Input_control, Input_parsing2)
- TEST_F (Input_control, Input_parsing3)
- TEST_F (Input_control, Input_parsing4)
- TEST_F (Input_control, Input_parsing5)
- TEST_F (Nroot_test, Nroot_test1)
- TEST_F (Nroot_test, Nroot_test2)
- TEST_F (Nroot_test, Nroot_test3)

- TEST_F (Nroot_test, Nroot_test4)
- TEST_F (Nroot_test, Nroot_test5)
- TEST_F (Nroot_test, Nroot_test6)
- TEST_F (Nroot_test, Nroot_test7)
- TEST_F (Nroot_test, Nroot_test8)
- TEST_F (Nroot_test, Nroot_test9)
- TEST_F (Nroot_test, Nroot_test10)
- TEST_F (Nroot_test, Nroot_test11)
- TEST_F (Nroot_test, Nroot_test12)
- TEST_F (Real_time_calc, Real_time_calc_test1)
- TEST_F (Real_time_calc, Real_time_calc_test2)
- TEST_F (Real_time_calc, Real_time_calc_test3)
- TEST_F (Real_time_calc, Real_time_calc_test4)
- TEST_F (Real_time_calc, Real_time_calc_test5)
- TEST_F (Real_time_calc, Real_time_calc_test6)
- TEST_F (Real_time_calc, Real_time_calc_test7)
- TEST_F (Real_time_calc, Real_time_calc_test8)
- TEST_F (Real_time_calc, Real_time_calc_test9)
- TEST_F (Real_time_calc, Real_time_calc_test10)
- TEST_F (Real_time_calc, Real_time_calc_test11)
- TEST_F (Real_time_calc, Real_time_calc_test12)
- TEST_F (Real_time_calc, Real_time_calc_test13)
- TEST_F (Real_time_calc, Real_time_calc_test14)
- TEST_F (Real_time_calc, Real_time_calc_test15)
- TEST_F (Real_time_calc, Real_time_calc_test16)
- TEST_F (Real_time_calc, Real_time_calc_test17)
- TEST_F (Real_time_calc, Real_time_calc_test18)
- TEST_F (Real_time_calc, Real_time_calc_test19)
- TEST_F (Real_time_calc, Real_time_calc_test20)
- TEST_F (Real_time_calc, Real_time_calc_test21)
- TEST_F (Real_time_calc, Real_time_calc_test22)
- TEST_F (Real_time_calc, Real_time_calc_test23)
- TEST_F (Real_time_calc, Real_time_calc_test24)
- TEST_F (Real_time_calc, Real_time_calc_test25)
- TEST_F (Real_time_calc, Real_time_calc_test26)
- TEST_F (Real_time_calc, Real_time_calc_test27)
- TEST_F (Real_time_calc, Real_time_calc_test28)
- TEST_F (Real_time_calc, Real_time_calc_test29)
- TEST_F (Real_time_calc, Real_time_calc_test30)
- TEST_F (Real_time_calc, Real_time_calc_test31)
- TEST_F (Real_time_calc, Real_time_calc_test32)
- TEST_F (Real_time_calc, Real_time_calc_test33)
- TEST_F (Real_time_calc, Real_time_calc_test34)
- TEST_F (Real_time_calc, Real_time_calc_test35)
- TEST_F (Real_time_calc, Real_time_calc_test36)
- TEST_F (Real_time_calc, Real_time_calc_test37)
- TEST_F (Real_time_calc, Real_time_calc_test38)
- TEST_F (Real_time_calc, Real_time_calc_test39)
- TEST_F (Real_time_calc, Real_time_calc_test40)
- TEST_F (Real_time_calc, Real_time_calc_test41)
- TEST_F (Real_time_calc, Real_time_calc_test42)
- TEST_F (Real_time_calc, Real_time_calc_test43)
- TEST_F (Real_time_calc, Real_time_calc_test44)
- TEST_F (Real_time_calc, Real_time_calc_test45)
- TEST_F (Real_time_calc, Real_time_calc_test46)

- TEST_F (Real_time_calc, Real_time_calc_test47)
- TEST_F (Real_time_calc, Real_time_calc_test48)
- TEST_F (Real_time_calc, Real_time_calc_test49)
- TEST_F (Real_time_calc, Real_time_calc_test50)
- TEST_F (Real_time_calc, Real_time_calc_test51)
- TEST_F (Real_time_calc, Real_time_calc_test52)
- TEST_F (Real_time_calc, Real_time_calc_test53)
- TEST_F (Real_time_calc, Real_time_calc_test54)
- TEST_F (Real_time_calc, Real_time_calc_test55)
- TEST_F (Real_time_calc, Real_time_calc_test56)
- TEST_F (Real_time_calc, Real_time_calc_test57)
- TEST_F (Real_time_calc, Real_time_calc_test58)
- TEST_F (Real_time_calc, Real_time_calc_test59)
- TEST_F (Real_time_calc, Real_time_calc_test60)
- TEST_F (Real_time_calc, Real_time_calc_test61)
- TEST_F (Real_time_calc, Real_time_calc_test62)
- TEST_F (Real_time_calc, Real_time_calc_test63)
- TEST_F (Real_time_calc, Real_time_calc_test64)
- TEST_F (Real_time_calc, Real_time_calc_test65)
- TEST_F (Real_time_calc, Real_time_calc_test66)
- TEST_F (Real_time_calc, Real_time_calc_test67)
- TEST_F (Factorial_tests, Factorial_tests1)
- TEST_F (Factorial_tests, Factorial_tests2)
- TEST_F (Factorial_tests, Factorial_tests3)
- TEST_F (Factorial_tests, Factorial_tests4)
- TEST_F (Factorial_tests, Factorial_tests5)
- TEST_F (Factorial_tests, Factorial_tests6)
- TEST_F (Factorial_tests, Factorial_tests7)
- TEST_F (Npow_test, Npow_test1)
- TEST_F (Npow_test, Npow_test2)
- TEST_F (Npow_test, Npow_test3)
- TEST_F (Npow_test, Npow_test4)
- TEST_F (Npow_test, Npow_test5)
- TEST_F (Npow_test, Npow_test6)
- TEST_F (Xpow_test, Xpow_test1)
- TEST_F (Xpow_test, Xpow_test2)
- TEST_F (Xpow_test, Xpow_test3)
- TEST_F (Xpow_test, Xpow_test4)
- TEST_F (Xpow_test, Xpow_test5)
- TEST_F (Xpow_test, Xpow_test6)
- TEST_F (Xpow_test, Xpow_test7)
- TEST_F (Xpow_test, Xpow_test8)
- TEST_F (Xpow_test, Xpow_test9)
- TEST_F (Xpow_test, Xpow_test10)
- TEST_F (Xpow_test, Xpow_test11)
- TEST_F (Xpow_test, Xpow_test12)
- TEST_F (Nsin_test, Sinx_test1)
- TEST_F (Nsin_test, Sinx_test2)
- TEST_F (Nsin_test, Sinx_test3)
- TEST_F (Nsin_test, Sinx_test4)
- TEST_F (Ncos_test, Cosx_test1)
- TEST_F (Ncos_test, Cosx_test2)
- TEST_F (Ncos_test, Cosx_test3)
- TEST_F (Ncos_test, Cosx_test4)
- TEST_F (Ntan_test, Tanx_test1)

- TEST_F (Ntan_test, Tanx_test2)
- TEST (Add_test, Add_test1)
- TEST (Add_test, Add_test2)
- TEST (Add_test, Add_test3)
- TEST (Add_test, Add_test4)
- TEST (Add_test, Add_test5)
- TEST (Sub_test, Sub_test1)
- TEST (Sub_test, Sub_test2)
- TEST (Sub_test, Sub_test3)
- TEST (Sub_test, Sub_test4)
- TEST (Sub_test, Sub_test5)
- TEST (Mul_test, Mul_test1)
- TEST (Mul_test, Mul_test2)
- TEST (Mul_test, Mul_test3)
- TEST (Mul_test, Mul_test4)
- TEST (Mul_test, Mul_test5)
- TEST (Div_test, Div_test1)
- TEST (Div_test, Div_test2)
- TEST (Div_test, Div_test3)
- TEST (Div_test, Div_test4)
- TEST (Div_test, Div_test5)

6.12.1 Detailed Description

Source file containing tests for each part of the math library.

Author

Mihola David
Sokolovskii Vladislav
Foltyn Lukas

Copyright

6.12.2 Function Documentation

6.12.2.1 TEST() [1/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str1  )
```

Definition at line 15 of file tests.cpp.

```
16 {
17     ADD_EQUATION_WITH_RESULT_STS(("2+3*-6/2+5"), ("-2"));
18 }
```

6.12.2.2 TEST() [2/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str2  )
```

Definition at line 20 of file tests.cpp.

```
21 {
22     ADD_EQUATION_WITH_RESULT_STS(("12+3.5^6*1.2-14*1"), ("2203.91875"));
23 }
```

6.12.2.3 TEST() [3/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str3  )
```

Definition at line 25 of file tests.cpp.

```
26 {
27     ADD_EQUATION_WITH_RESULT_STS(("3+8*2/0+5"), ("Division by 0."));
28 }
```

6.12.2.4 TEST() [4/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str4  )
```

Definition at line 30 of file tests.cpp.

```
31 {
32     ADD_EQUATION_WITH_RESULT_STS(("3+-8?4"), ("Cannot root this."));
33 }
```

6.12.2.5 TEST() [5/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str6  )
```

Definition at line 35 of file tests.cpp.

```
36 {
37     ADD_EQUATION_WITH_RESULT_STS(("2+8-15+-18"), ("-23"));
38 }
```

6.12.2.6 TEST() [6/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str7  )
```

Definition at line 40 of file tests.cpp.

```
41 {
42     ADD_EQUATION_WITH_RESULT_STS(("2*-4+5*2/2"), ("-3"));
43 }
```

6.12.2.7 TEST() [7/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str8  )
```

Definition at line 45 of file tests.cpp.

```
46 {
47     ADD_EQUATION_WITH_RESULT_STS(("-4-3-9-8+-8/4"), ("-26"));
48 }
```

6.12.2.8 TEST() [8/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str9  )
```

Definition at line 50 of file tests.cpp.

```
51 {
52     ADD_EQUATION_WITH_RESULT_STS(("15^2+2.5*8-4!"), ("221"));
53 }
```

6.12.2.9 TEST() [9/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str10  )
```

Definition at line 55 of file tests.cpp.

```
56 {
57     ADD_EQUATION_WITH_RESULT_STS(("16?4*3-4.35^1/2!"), ("3.825"));
58 }
```

6.12.2.10 TEST() [10/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str11  )
```

Definition at line 60 of file tests.cpp.

```
61 {
62     ADD_EQUATION_WITH_RESULT_STS(("2!-2!+4!*4!-2*576/2!"), ("0"));
63 }
```

6.12.2.11 TEST() [11/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str12  )
```

Definition at line 65 of file tests.cpp.

```
66 {
67     ADD_EQUATION_WITH_RESULT_STS(("4?2*8?3*16?4*32?5-5!"), ("-104"));
68 }
```

6.12.2.12 TEST() [12/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str13  )
```

Definition at line 70 of file tests.cpp.

```
71 {
72     ADD_EQUATION_WITH_RESULT_STS(("3!^2*3-8/4"), ("106"));
73 }
```

6.12.2.13 TEST() [13/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str14  )
```

Definition at line 75 of file tests.cpp.

```
76 {
77     ADD_EQUATION_WITH_RESULT_STS(("8.45*4-45+2/5/54*5!+-9+-8?8+66*5"), ("Cannot root this."));
78 }
```

6.12.2.14 TEST() [14/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str15  )
```

Definition at line 80 of file tests.cpp.

```
81 {
82     ADD_EQUATION_WITH_RESULT_STS(("6/6+6*6-2*4^2!"), ("5"));
83 }
```

6.12.2.15 TEST() [15/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str16  )
```

Definition at line 85 of file tests.cpp.

```
86 {
87     ADD_EQUATION_WITH_RESULT_STS(("18^2?2+18*18/18^2?2-3!^3"), ("-180"));
88 }
```

6.12.2.16 TEST() [16/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str17  )
```

Definition at line 90 of file tests.cpp.

```
91 {
92     ADD_EQUATION_WITH_RESULT_STS(("0!*125?3+-32?5+2*2!+2*8.5"), ("24"));
93 }
```

6.12.2.17 TEST() [17/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str18  )
```

Definition at line 95 of file tests.cpp.

```
96 {
97     ADD_EQUATION_WITH_RESULT_STS(("8!-45^23*12?2/0+-4!?3"), ("Division by 0."));
98 }
```

6.12.2.18 TEST() [18/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str19  )
```

Definition at line 100 of file tests.cpp.

```
101 {
102     ADD_EQUATION_WITH_RESULT_STS(("8*45*0/45*5!*4?2^12"), ("0"));
103 }
```

6.12.2.19 TEST() [19/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str20  )
```

Definition at line 105 of file tests.cpp.

```
106 {
107     ADD_EQUATION_WITH_RESULT_STS(("5!^3!+10*2^3/8-100-250^5*4+963258741"), ("-919302741149"));
108 }
```

6.12.2.20 TEST() [20/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str21  )
```

Definition at line 110 of file tests.cpp.

```
111 {
112     ADD_EQUATION_WITH_RESULT_STS(("2*(4+5)"), ("18"));
113 }
```

6.12.2.21 TEST() [21/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str22  )
```

Definition at line 115 of file tests.cpp.

```
116 {
117     ADD_EQUATION_WITH_RESULT_STS(("3+2*(4+5)^2"), ("165"));
118 }
```

6.12.2.22 TEST() [22/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str23  )
```

Definition at line 120 of file tests.cpp.

```
121 {
122     ADD_EQUATION_WITH_RESULT_STS(("3+2*(4+5)^2-65"), ("100"));
123 }
```

6.12.2.23 TEST() [23/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str24  )
```

Definition at line 125 of file tests.cpp.

```
126 {
127     //it is necessary to put sin cos and tan i to this equations, the don't go threw the string check
      mechanism
128     ADD_EQUATION_WITH_RESULT_STS(("2*((2.5+2.5)-2*(sin90+(cos(0)))+-1*4)^2+2"), ("20"));
129 }
```

6.12.2.24 TEST() [24/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str25  )
```

Definition at line 131 of file tests.cpp.

```
132 {
133     ADD_EQUATION_WITH_RESULT_STS(("(((5)*2+-15/(1+2^2)+3+-(2*2^2+2))"), ("0"));
134 }
```

6.12.2.25 TEST() [25/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str26  )
```

Definition at line 136 of file tests.cpp.

```
137 {
138     ADD_EQUATION_WITH_RESULT_STS(("2*(2-((5)*2+-15/(1+2^2)+3+-(2*2^2+2)))-3"), ("1"));
139 }
```

6.12.2.26 TEST() [26/56]

```
TEST (
            Logic_test2 ,
            Calculation_str_to_str27  )
```

Definition at line 141 of file tests.cpp.

```
142 {
143     ADD_EQUATION_WITH_RESULT_STS(("10+-(sin90*2-cos0*4+2*(-sin90+6))*2"), ("-6"));
144 }
```

6.12.2.27 TEST() [27/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld1  )
```

Definition at line 151 of file tests.cpp.

```
152 {
153     ADD_EQUATION_WITH_RESULT_STD(("5!^3!+10*2^3/8-100-250^5*4+963258741"), (-919302741149));
154 }
```

6.12.2.28 TEST() [28/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld2  )
```

Definition at line 156 of file tests.cpp.

```
157 {
158     ADD_EQUATION_WITH_RESULT_STD(("3.24+5.86*2.2/3*2^6/16?4+25/5*2"), (150.754666));
159 }
```

6.12.2.29 TEST() [29/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld3  )
```

Definition at line 161 of file tests.cpp.

```
162 {
163     ADD_EQUATION_WITH_RESULT_STD(("1024?2/2/2/2*2^7+512"), (1024));
164 }
```

6.12.2.30 TEST() [30/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld4  )
```

Definition at line 166 of file tests.cpp.

```
167 {
168     ADD_EQUATION_WITH_RESULT_STD(("-4.3*123+10^3*(12/-3.3)+0.7"), (-4164.56363636));
169 }
```

6.12.2.31 TEST() [31/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld5  )
```

Definition at line 171 of file tests.cpp.

```
172 {
173     ADD_EQUATION_WITH_RESULT_STD(("(9!^0.5-14*2*(14488%300)^2)?5"), (-11.6676402745))
174 }
```

6.12.2.32 TEST() [32/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld6  )
```

Definition at line 176 of file tests.cpp.

```
177 {
178     ADD_EQUATION_WITH_RESULT_STD(("-0.13*11-15*(13+69)-1228.57/12"), (-1333.81083333));
179 }
```

6.12.2.33 TEST() [33/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld7  )
```

Definition at line 181 of file tests.cpp.

```
182 {
183     ADD_EQUATION_WITH_RESULT_STD(("(((543-122%2*2+(14+8)-540))"), (25));
184 }
```

6.12.2.34 TEST() [34/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld8  )
```

Definition at line 186 of file tests.cpp.

```
187 {
188     ADD_EQUATION_WITH_RESULT_STD(("(((2)N+13N+20)!-100)?2"), (4.472135955));
189 }
```

6.12.2.35 TEST() [35/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld9  )
```

Definition at line 191 of file tests.cpp.

```
192 {
193     ADD_EQUATION_WITH_RESULT_STD(("((10^0.3-16+22*2.5)/120+1.7)N"), (-2.04162718596));
194 }
```

6.12.2.36 TEST() [36/56]

```
TEST (
            Logic_test3 ,
            Calculation_str_to_ld10  )
```

Definition at line 196 of file tests.cpp.

```
197 {
198     ADD_EQUATION_WITH_RESULT_STD(("(144^(1/2)/3+2)!-333"), (1053));
199 }
```

6.12.2.37 TEST() [37/56]

```
TEST (
            Add_test ,
            Add_test1  )
```

Definition at line 1039 of file tests.cpp.

```
1040 {
1041     EXPECT_DOUBLE_EQ(add(2.25, 1.25), 3.5);
1042 }
```

6.12.2.38 TEST() [38/56]

```
TEST (
              Add_test ,
              Add_test2  )
```

Definition at line 1044 of file tests.cpp.

```
1045 {
1046     EXPECT_DOUBLE_EQ(add(5.3, -1.2), 4.1);
1047 }
```

6.12.2.39 TEST() [39/56]

```
TEST (
              Add_test ,
              Add_test3  )
```

Definition at line 1049 of file tests.cpp.

```
1050 {
1051     EXPECT_DOUBLE_EQ(add(-8.25, -1.75), -10);
1052 }
```

6.12.2.40 TEST() [40/56]

```
TEST (
              Add_test ,
              Add_test4  )
```

Definition at line 1054 of file tests.cpp.

```
1055 {
1056     EXPECT_DOUBLE_EQ(add(-4.8, 1.2), -3.6);
1057 }
```

6.12.2.41 TEST() [41/56]

```
TEST (
              Add_test ,
              Add_test5  )
```

Definition at line 1059 of file tests.cpp.

```
1060 {
1061     EXPECT_DOUBLE_EQ(add(0, 0), 0);
1062 }
```

6.12.2.42 TEST() [42/56]

```
TEST (
              Sub_test ,
              Sub_test1  )
```

Definition at line 1066 of file tests.cpp.

```
1067 {
1068     EXPECT_DOUBLE_EQ(sub(2.25, 1.25), 1);
1069 }
```

6.12.2.43 TEST() [43/56]

```
TEST (
              Sub_test ,
              Sub_test2  )
```

Definition at line 1071 of file tests.cpp.

```
1072 {
1073     EXPECT_DOUBLE_EQ(sub(5.3, -1.2), 6.5);
1074 }
```

6.12.2.44 TEST() [44/56]

```
TEST (
              Sub_test ,
              Sub_test3  )
```

Definition at line 1076 of file tests.cpp.

```
1077 {
1078     EXPECT_DOUBLE_EQ(sub(-8.25, -1.75), -6.5);
1079 }
```

6.12.2.45 TEST() [45/56]

```
TEST (
              Sub_test ,
              Sub_test4  )
```

Definition at line 1081 of file tests.cpp.

```
1082 {
1083     EXPECT_DOUBLE_EQ(sub(-4.8, 1.2), -6);
1084 }
```

6.12.2.46 TEST() [46/56]

```
TEST (
              Sub_test ,
              Sub_test5  )
```

Definition at line 1086 of file tests.cpp.

```
1087 {
1088     EXPECT_DOUBLE_EQ(sub(0, 0), 0);
1089 }
```

6.12.2.47 TEST() [47/56]

```
TEST (
              Mul_test ,
              Mul_test1  )
```

Definition at line 1093 of file tests.cpp.

```
1094 {
1095     EXPECT_DOUBLE_EQ(mul(2.25, 1.25), 2.8125);
1096 }
```

6.12.2.48 TEST() [48/56]

```
TEST (
              Mul_test ,
              Mul_test2  )
```

Definition at line 1098 of file tests.cpp.

```
1099 {
1100     EXPECT_DOUBLE_EQ(mul(5.3, -1.2), -6.36);
1101 }
```

6.12.2.49 TEST() [49/56]

```
TEST (
              Mul_test ,
              Mul_test3  )
```

Definition at line 1103 of file tests.cpp.

```
1104 {
1105     EXPECT_DOUBLE_EQ(mul(-8.25, -1.75), 14.4375);
1106 }
```

6.12.2.50 TEST() [50/56]

```
TEST (
              Mul_test ,
              Mul_test4  )
```

Definition at line 1108 of file tests.cpp.

```
1109 {
1110     EXPECT_DOUBLE_EQ(mul(-4.8, 1.2), -5.76);
1111 }
```

6.12.2.51 TEST() [51/56]

```
TEST (
              Mul_test ,
              Mul_test5  )
```

Definition at line 1113 of file tests.cpp.

```
1114 {
1115     EXPECT_DOUBLE_EQ(mul(0, 0), 0);
1116 }
```

6.12.2.52 TEST() [52/56]

```
TEST (
              Div_test ,
              Div_test1  )
```

Definition at line 1120 of file tests.cpp.

```
1121 {
1122     EXPECT_DOUBLE_EQ(div_(2.25, 1.25), 1.8);
1123 }
```

6.12.2.53 TEST() [53/56]

```
TEST (
              Div_test ,
              Div_test2  )
```

Definition at line 1125 of file tests.cpp.

```
1126 {
1127     EXPECT_DOUBLE_EQ(div_(5.3, -1.2), -4.416666666666666);
1128 }
```

6.12.2.54 TEST() [54/56]

```
TEST (
              Div_test ,
              Div_test3  )
```

Definition at line 1130 of file tests.cpp.

```
1131 {
1132     EXPECT_DOUBLE_EQ(div_(-8.25, -1.75), 4.7142857142857144);
1133 }
```

6.12.2.55 TEST() [55/56]

```
TEST (
              Div_test ,
              Div_test4  )
```

Definition at line 1135 of file tests.cpp.

```
1136 {
1137     EXPECT_DOUBLE_EQ(div_(-4.8, 1.2), -4);
1138 }
```

6.12.2.56 TEST() [56/56]

```
TEST (
              Div_test ,
              Div_test5  )
```

Definition at line 1140 of file tests.cpp.

```
1141 {
1142     EXPECT_THROW(div_(0, 0), std::runtime_error);
1143 }
```

6.12.2.57 TEST_F() [1/119]

```
TEST_F (
              Input_control ,
              Input_parsing1  )
```

Definition at line 204 of file tests.cpp.

```
205 {
206     EXPECT_TRUE(parse_string("2+2-4", "2+2-4"));
207 }
```

6.12.2.58 TEST_F() [2/119]

```
TEST_F (
              Input_control ,
              Input_parsing2  )
```

Definition at line 209 of file tests.cpp.

```
210 {
211     EXPECT_TRUE(parse_string("2+--2.5!+*/0.25*+-6!+5^6.62", "2-2.5/0.25+-6+5^6.62"));
212 }
```

6.12.2.59 TEST_F() [3/119]

```
TEST_F (
              Input_control ,
              Input_parsing3  )
```

Definition at line 214 of file tests.cpp.

```
215 {
216     EXPECT_TRUE(parse_string("2+*!+-*-5.5^-5!!!!", "2*-5.5^-5"));
217 }
```

6.12.2.60 TEST_F() [4/119]

```
TEST_F (
              Input_control ,
              Input_parsing4  )
```

Definition at line 219 of file tests.cpp.

```
220 {
221     EXPECT_TRUE(parse_string("89+5.3%5++++*/-85!!!!^000.25", "89+5.35/-85^000.25"));
222 }
```

6.12.2.61 TEST_F() [5/119]

```
TEST_F (
              Input_control ,
              Input_parsing5  )
```

Definition at line 224 of file tests.cpp.

```
225 {
226     EXPECT_TRUE(parse_string("((-+*/*-85+56N))", "((-85+-56))"));
227 }
```

6.12.2.62 TEST_F() [6/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test1  )
```

Definition at line 233 of file tests.cpp.

```
234 {
235     EXPECT_TRUE(root_test(16, 4 , 2, NO));
236 }
```

6.12.2.63 TEST_F() [7/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test2  )
```

Definition at line 238 of file tests.cpp.

```
239 {
240     EXPECT_TRUE(root_test(-16, 4 , 2, YES));
241 }
```

6.12.2.64 TEST_F() [8/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test3  )
```

Definition at line 243 of file tests.cpp.

```
244 {
245     EXPECT_TRUE(root_test(2, 2, 1.41421, NO));
246 }
```

6.12.2.65 TEST_F() [9/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test4  )
```

Definition at line 248 of file tests.cpp.

```
249 {
250     EXPECT_TRUE(root_test(1, 999123131, 1, NO));
251 }
```

6.12.2.66 TEST_F() [10/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test5  )
```

Definition at line 253 of file tests.cpp.

```
254 {
255     EXPECT_TRUE(root_test(150, 10, 1.65047, NO));
256 }
```

6.12.2.67 TEST_F() [11/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test6  )
```

Definition at line 258 of file tests.cpp.

```
259 {
260     EXPECT_TRUE(root_test(123.2, 0, 1, NO));
261 }
```

6.12.2.68 TEST_F() [12/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test7  )
```

Definition at line 263 of file tests.cpp.

```
264 {
265     EXPECT_TRUE(root_test(-42.2, 1, -42.2, NO));
266 }
```

6.12.2.69 TEST_F() [13/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test8  )
```

Definition at line 268 of file tests.cpp.

```
269 {
270     EXPECT_TRUE(root_test(0, 15, 0, NO));
271 }
```

6.12.2.70 TEST_F() [14/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test9  )
```

Definition at line 273 of file tests.cpp.

```
274 {
275     EXPECT_TRUE(root_test(0.999999999999, 1432, 1, NO));
276 }
```

6.12.2.71 TEST_F() [15/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test10  )
```

Definition at line 278 of file tests.cpp.

```
279 {
280     EXPECT_TRUE(root_test(-52.124, 13, -1.3554385, NO));
281 }
```

6.12.2.72 TEST_F() [16/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test11  )
```

Definition at line 283 of file tests.cpp.

```
284 {
285     EXPECT_TRUE(root_test(-112, 4, 0, YES));
286 }
```

6.12.2.73 TEST_F() [17/119]

```
TEST_F (
              Nroot_test ,
              Nroot_test12  )
```

Definition at line 288 of file tests.cpp.

```
289 {
290     EXPECT_TRUE(root_test(-1, 14, 0, YES));
291 }
```

6.12.2.74 TEST_F() [18/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test1  )
```

Definition at line 297 of file tests.cpp.

```
298 {
299     //checking at possition 1, 4 and 6. Expecting on results 10, 30, 33 at the possitions respectivly

300     test_real_time_calculation("10+20+30", {1, 4, 6}, {10, 30, 33});
301 }
```

6.12.2.75 TEST_F() [19/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test2  )
```

Definition at line 303 of file tests.cpp.

```
304 {
305     test_real_time_calculation("10+20*40", {0, 1, 4, 6, 7}, {1, 10, 30, 90, 810});
306 }
```

6.12.2.76 TEST_F() [20/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test3  )
```

Definition at line 308 of file tests.cpp.

```
309 {
310     test_real_time_calculation("10+20*40-410", {0, 1, 4, 6, 7, 10, 11}, {1, 10, 30, 90, 810, 769, 400});
311 }
```

6.12.2.77 TEST_F() [21/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test4  )
```

Definition at line 313 of file tests.cpp.

```
314 {
315     test_real_time_calculation("10+2*4^2", {3, 5, 7}, {12, 18, 42});
316 }
```

6.12.2.78 TEST_F() [22/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test5  )
```

Definition at line 318 of file tests.cpp.

```
319 {
320     test_real_time_calculation("10+2*4^2-5*8", {3, 5, 7, 9, 11}, {12, 18, 42, 37, 2});
321 }
```

6.12.2.79 TEST_F() [23/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test6  )
```

Definition at line 323 of file tests.cpp.

```
324 {
325     test_real_time_calculation("10+2*4^2-5*8", {3, 5, 7, 9, 11}, {12, 18, 42, 37, 2});
326 }
```

6.12.2.80 TEST_F() [24/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test7  )
```

Definition at line 328 of file tests.cpp.

```
329 {
330     test_real_time_calculation("5*2*4+2^3", {4, 8}, {40, 48});
331 }
```

6.12.2.81 TEST_F() [25/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test8  )
```

Definition at line 333 of file tests.cpp.

```
334 {
335     test_real_time_calculation("5*2*4+2^3-25.5", {4, 8, 11, 13}, {40, 48, 23, 22.5});
336 }
```

6.12.2.82 TEST_F() [26/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test9  )
```

Definition at line 338 of file tests.cpp.

```
339 {
340     test_real_time_calculation("5*2*4+2^3-25.5+7.5*3", {4, 8, 11, 13, 17, 19}, {40, 48, 23, 22.5, 30,
      45});
341 }
```

6.12.2.83 TEST_F() [27/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test10  )
```

Definition at line 343 of file tests.cpp.

```
344 {
345     test_real_time_calculation("1+(2+3)+4", {3, 5, 8}, {3, 6, 10});
346 }
```

6.12.2.84 TEST_F() [28/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test11  )
```

Definition at line 348 of file tests.cpp.

```
349 {
350     test_real_time_calculation("1+((2+3))+4", {4, 6, 10}, {3, 6, 10});
351 }
```

6.12.2.85 TEST_F() [29/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test12  )
```

Definition at line 353 of file tests.cpp.

```
354 {
355     test_real_time_calculation("1+(((2+3)))+4", {5, 7, 12}, {3, 6, 10});
356 }
```

6.12.2.86 TEST_F() [30/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test13  )
```

Definition at line 358 of file tests.cpp.

```
359 {
360     test_real_time_calculation("2*(2+3)", {3, 5}, {4, 10});
361 }
```

6.12.2.87 TEST_F() [31/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test14  )
```

Definition at line 363 of file tests.cpp.

```
364 {
365     test_real_time_calculation("2*(2+3)*2", {3, 5, 8}, {4, 10, 20});
366 }
```

6.12.2.88 TEST_F() [32/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test15  )
```

Definition at line 368 of file tests.cpp.

```
369 {
370     test_real_time_calculation("10+2*(2+3)*2", {6, 8, 11}, {14, 20, 30});
371 }
```

6.12.2.89 TEST_F() [33/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test16  )
```

Definition at line 373 of file tests.cpp.

```
374 {
375     test_real_time_calculation("10+2*(2+3)^2", {6, 8, 11}, {14, 20, 60});
376 }
```

6.12.2.90 TEST_F() [34/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test17  )
```

Definition at line 378 of file tests.cpp.

```
379 {
380     test_real_time_calculation("10+2*(2+3*2)^2", {6, 8, 10, 13}, {14, 20, 26, 138});
381 }
```

6.12.2.91 TEST_F() [35/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test18  )
```

Definition at line 383 of file tests.cpp.

```
384 {
385     test_real_time_calculation("10+2*(4+3*2^2)?2", {6, 8, 10, 12, 15}, {18, 24, 30, 42, 18});
386 }
```

6.12.2.92 TEST_F() [36/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test19  )
```

Definition at line 388 of file tests.cpp.

```
389 {
390     test_real_time_calculation("10+2*(4+3*2^2)?2+2", {6, 8, 10, 12, 15, 17}, {18, 24, 30, 42, 18, 20});
391 }
```

6.12.2.93 TEST_F() [37/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test20  )
```

Definition at line 393 of file tests.cpp.

```
394 {
395     test_real_time_calculation("(2+6)*(3+7)", {3, 7, 9, 10}, {8, 24, 80, 80});
396 }
```

6.12.2.94 TEST_F() [38/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test21  )
```

Definition at line 398 of file tests.cpp.

```
399 {
400     test_real_time_calculation("(2+6)*(3+7)/8", {3, 7, 9, 10, 12}, {8, 24, 80, 80, 10});
401 }
```

6.12.2.95 TEST_F() [39/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test22  )
```

Definition at line 403 of file tests.cpp.

```
404 {
405     test_real_time_calculation("(2+6)*(3+7)/8+5", {3, 7, 9, 10, 12, 14}, {8, 24, 80, 80, 10, 15});
406 }
```

6.12.2.96 TEST_F() [40/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test23  )
```

Definition at line 408 of file tests.cpp.

```
409 {
410     test_real_time_calculation("(2+6)+(2*3)*4^(1+1)*2", {7, 9, 12, 17, 20}, {10, 14, 32, 104, 200});
411 }
```

6.12.2.97 TEST_F() [41/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test24  )
```

Definition at line 413 of file tests.cpp.

```
414 {
415     test_real_time_calculation("5.5+2*(3.5+27?3)^2-10", {3, 5, 10, 13, 15, 18, 21}, {5.5, 7.5, 12.5,
       66.5, 18.5, 90, 80});
416 }
```

6.12.2.98 TEST_F() [42/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test25  )
```

Definition at line 418 of file tests.cpp.

```
419 {
420     test_real_time_calculation("(2*(10/(5+(2.5*2)))+8", {5, 8, 13, 15, 20}, {20, 4, 2.66666, 2, 10});
421 }
```

6.12.2.99 TEST_F() [43/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test26  )
```

Definition at line 423 of file tests.cpp.

```
424 {
425     test_real_time_calculation("2+3*(1+8/(2-2)+3.5*2)-10", {5, 7, 10, 13, 18, 20, 24}, {5, 29, 17, 11,
       21.5, 32, 22});
426 }
```

6.12.2.100 TEST_F() [44/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test27  )
```

Definition at line 428 of file tests.cpp.

```
429 {
430     test_real_time_calculation("10+-(4+5)+5", {5, 7, 10}, {6, 1, 6});
431 }
```

6.12.2.101 TEST_F() [45/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test28  )
```

Definition at line 433 of file tests.cpp.

```
434 {
435     test_real_time_calculation("10+-(4+5)*2", {5, 7, 10}, {6, 1, -8});
436 }
```

6.12.2.102 TEST_F() [46/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test29  )
```

Definition at line 438 of file tests.cpp.

```
439 {
440     test_real_time_calculation("150+2*-(3+2)^3", {4, 8, 10, 13}, {152, 144, 140, -100});
441 }
```

6.12.2.103 TEST_F() [47/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test30  )
```

Definition at line 443 of file tests.cpp.

```
444 {
445     test_real_time_calculation("150+2*-(3+2)^3*2+300", {4, 8, 10, 13, 15, 18, 19}, {152, 144, 140, -100,
      -350, -320, -50});
446 }
```

6.12.2.104 TEST_F() [48/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test31  )
```

Definition at line 448 of file tests.cpp.

```
449 {
450     test_real_time_calculation("150/-(((2+1)*2)+4)*-2", {8, 10, 13, 16, 20}, {-75, -50, -25, -15, 30});
451 }
```

6.12.2.105 TEST_F() [49/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test32  )
```

Definition at line 453 of file tests.cpp.

```
454 {
455     test_real_time_calculation("-250+(2+4)!*2-190", {6, 8, 10, 12, 16}, {-248, -244, 470, 1190, 1000});
456 }
```

6.12.2.106 TEST_F() [50/119]

```
TEST_F (
            Real_time_calc ,
            Real_time_calc_test33  )
```

Definition at line 458 of file tests.cpp.

```
459 {
460     test_real_time_calculation("8*(2+3)-6", {8}, {34});
461 }
```

6.12.2.107 TEST_F() [51/119]

```
TEST_F (
            Real_time_calc ,
            Real_time_calc_test34  )
```

Definition at line 463 of file tests.cpp.

```
464 {
465     test_real_time_calculation("524D*2.5-5^2", {2, 3, 5, 6, 7, 9, 11}, {524, 52, 104, 104, 130, 125,
      105});
466 }
```

6.12.2.108 TEST_F() [52/119]

```
TEST_F (
            Real_time_calc ,
            Real_time_calc_test35  )
```

Definition at line 468 of file tests.cpp.

```
469 {
470     test_real_time_calculation("-2N^(32-28D)%22", {1, 2, 5, 6, 8, 9, 10, 13, 14}, {-2, 2, 8, 4294967296,
      1073741824, 16, 1073741824, 0, 12});
471 }
```

6.12.2.109 TEST_F() [53/119]

```
TEST_F (
            Real_time_calc ,
            Real_time_calc_test36  )
```

Definition at line 473 of file tests.cpp.

```
474 {
475     test_real_time_calculation("(s45)^2+(c45)^2", {3, 6, 11, 14}, {0.7071067811, 0.5, 1.2071067811, 1});
476 }
```

6.12.2.110 TEST_F() [54/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test37  )
```

Definition at line 478 of file tests.cpp.

```
479 {
480     test_real_time_calculation("(s25DD35)^2+(c75DD35)^2", {3, 7, 10, 15, 19, 22}, {0.4226182617,
      0.573576436, 0.32898992,
481                                                          0.5878089734, 1.1481419726, 1});
482 }
```

6.12.2.111 TEST_F() [55/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test38  )
```

Definition at line 484 of file tests.cpp.

```
485 {
486     test_real_time_calculation("2+3*t45NN^6-3/s30NNN", {6, 7, 8, 10, 12, 16, 17, 18, 19}, {5, -1, 5, 5,
      2, -1, 11, -1, 11});
487 }
```

6.12.2.112 TEST_F() [56/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test39  )
```

Definition at line 489 of file tests.cpp.

```
490 {
491     test_real_time_calculation("9-14D*(2D-5NN)^2", {1, 3, 4, 7, 8, 9, 15}, {9,-5, 8, 7, 8, 8,-16});
492 }
```

6.12.2.113 TEST_F() [57/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test40  )
```

Definition at line 494 of file tests.cpp.

```
495 {
496     test_real_time_calculation("((((3+5)-2*2)*4-1)+2)*2", {4, 6, 9, 11, 14, 16, 19, 22}, {3, 8, 6, 4,
      16, 15, 17, 34});
497 }
```

6.12.2.114 TEST_F() [58/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test41  )
```

Definition at line 499 of file tests.cpp.

```
500 {
501     test_real_time_calculation("1*12345DNDDD+3D*2!N", {0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14, 16, 17, 18},
       {1, 12, 123, 1234, 12345, 1234, -1234, -123, -12, -1, -1, -2, -2, -2});
502 }
```

6.12.2.115 TEST_F() [59/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test42  )
```

Definition at line 504 of file tests.cpp.

```
505 {
506     test_real_time_calculation("((s90)-(3*(2-2)+5^(3-1)))+2D", {4, 8, 11, 13, 16, 19, 21, 26, 27, 28},
       {1, -2, -5, 1, -4, -124, -24, -22, -24});
507 }
```

6.12.2.116 TEST_F() [60/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test43  )
```

Definition at line 509 of file tests.cpp.

```
510 {
511     test_real_time_calculation("((2+2)^(2-1))N*(2+48)", {2, 4, 8, 10, 13, 16, 18, 19}, {2, 4, 16, 4, -4,
       -8, -24, -200});
512 }
```

6.12.2.117 TEST_F() [61/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test44  )
```

Definition at line 514 of file tests.cpp.

```
515 {
516     test_real_time_calculation("((s0)N*(c0)N)^54", {15}, {0});
517 }
```

6.12.2.118 TEST_F() [62/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test45  )
```

Definition at line 519 of file tests.cpp.

```
520 {
521     test_real_time_calculation("(8+17D8)*2D10", {1, 3, 4, 5, 6, 9, 10, 11, 12}, {8, 9, 25, 9, 26, 52,
       26, 26, 260});
522 }
```

6.12.2.119 TEST_F() [63/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test46  )
```

Definition at line 524 of file tests.cpp.

```
525 {
526     test_real_time_calculation("((2-3)*(4+2D1))N", {2, 4, 8, 10, 11, 12, 15}, {2, -1, -4, -6, -4, -5,
       5});
527 }
```

6.12.2.120 TEST_F() [64/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test47  )
```

Definition at line 529 of file tests.cpp.

```
530 {
531     test_real_time_calculation("(35DNN-5ND)*2^3", {2, 3, 4, 5, 7, 8, 9, 12, 14}, {35, 3, -3, 3, -2, 8,
       3, 6, 24});
532 }
```

6.12.2.121 TEST_F() [65/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test48  )
```

Definition at line 534 of file tests.cpp.

```
535 {
536     test_real_time_calculation("653DDNN98DN", {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }, {6, 65, 653, 65, 6,
       -6, 6, 69, 698, 69, -69});
537 }
```

6.12.2.122 TEST_F() [66/119]

```
TEST_F (
             Real_time_calc ,
             Real_time_calc_test49  )
```

Definition at line 539 of file tests.cpp.

```
540 {
541     test_real_time_calculation("(8+4+6)N*10D-44DN", {7, 9, 10, 11, 14, 15, 16}, {-18, -18, -180, -18,
      -62, -22, -14});
542 }
```

6.12.2.123 TEST_F() [67/119]

```
TEST_F (
             Real_time_calc ,
             Real_time_calc_test50  )
```

Definition at line 544 of file tests.cpp.

```
545 {
546     test_real_time_calculation("(2^3D2N)+-1^2", {3, 4, 5, 6, 10, 12}, {8, 2, 4, 0.25, -0.75, 1.25});
547 }
```

6.12.2.124 TEST_F() [68/119]

```
TEST_F (
             Real_time_calc ,
             Real_time_calc_test51  )
```

Definition at line 549 of file tests.cpp.

```
550 {
551     test_real_time_calculation("s(c0+89)*2^2.5-6^1.25", {3, 5, 6, 9, 13, 15, 20},
552                                {0.0174524064, 0.156434465, 1, 2, 5.656854249, -0.3431457505,
      -3.7336532309});
553 }
```

6.12.2.125 TEST_F() [69/119]

```
TEST_F (
             Real_time_calc ,
             Real_time_calc_test52  )
```

Definition at line 555 of file tests.cpp.

```
556 {
557     test_real_time_calculation("2!+3!*8!^(1/4!)+s(0!)-5", {1, 4, 7, 12, 13, 19, 22},
558                                              {2, 8, 241922, 87.022012615, 11.333573221,
      11.3510256282, 6.351025628});
559 }
```

6.12.2.126 TEST_F() [70/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test53  )
```

Definition at line 561 of file tests.cpp.

```
562 {
563     test_real_time_calculation("(s45+c45)!+2*3", {3, 7, 9, 11, 13}, {0.70710678, 1.4142135623, 1, 3,
       7});
564 }
```

6.12.2.127 TEST_F() [71/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test54  )
```

Definition at line 566 of file tests.cpp.

```
567 {
568     test_real_time_calculation("2*(c0)+5*(s90+2)^3", {7, 12, 14, 17}, {7, 7, 17, 137});
569 }
```

6.12.2.128 TEST_F() [72/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test55  )
```

Definition at line 571 of file tests.cpp.

```
572 {
573     test_real_time_calculation("5+2*(1+2)^2*3-20*(c0+s90)*2/0.5+s45*2^0.5", {15, 19, 23, 26, 31, 40},
       {39, 39, 19, -21, -101, -100});
574 }
```

6.12.2.129 TEST_F() [73/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test56  )
```

Definition at line 576 of file tests.cpp.

```
577 {
578     test_real_time_calculation("(-2*(s90))N^(2D2N)", {2, 7, 10, 13, 14, 15, 16}, {-2, -2, 2, 4, 2, 4,
       0.25 });
579 }
```

6.12.2.130 TEST_F() [74/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test57  )
```

Definition at line 581 of file tests.cpp.

```
582 {
583     test_real_time_calculation("(s30)^(2N)DD-1)", {3, 7, 8, 11, 13}, {0.5, 0.25, 4, 0.5, 2});
584 }
```

6.12.2.131 TEST_F() [75/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test58  )
```

Definition at line 586 of file tests.cpp.

```
587 {
588     test_real_time_calculation("-(t180)+5!DN*123DN", {5, 8, 9, 10, 11, 15, 16,17}, {0, 5, 120, 5, -5,
      -615, -60, 60});
589 }
```

6.12.2.132 TEST_F() [76/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test59  )
```

Definition at line 591 of file tests.cpp.

```
592 {
593     test_real_time_calculation("(((4-2)+1)-3*2)/(6/2-((5+3)*(1+1)))", {5, 8, 11, 13, 19, 33}, {2, 3, 0,
      -3, -1, 0.23076923});
594 }
```

6.12.2.133 TEST_F() [77/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test60  )
```

Definition at line 596 of file tests.cpp.

```
597 {
598     test_real_time_calculation("((((5-3)*(3-1))N*(3+1))N^2)N", {6, 13, 15, 21, 23, 25, 27}, {2, 4, -4,
      -16, 16, 256, -256});
599 }
```

6.12.2.134 TEST_F() [78/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test61  )
```

Definition at line 601 of file tests.cpp.

```
602 {
603     test_real_time_calculation("((sin90)N)*(2-c0)N*3!DN", {8, 12, 15, 17, 19, 20, 21, 22}, {-1, -2, -1,
      1, 3, 6, 3, -3});
604 }
```

6.12.2.135 TEST_F() [79/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test62  )
```

Definition at line 606 of file tests.cpp.

```
607 {
608     test_real_time_calculation("(83DN-(((c(s0))^1224243DDDD)^(1N)))-24/8", {4, 12, 18, 23, 25, 32, 39},
      {-8, -9, -9, -9, -9, -9, -12});
609 }
```

6.12.2.136 TEST_F() [80/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test63  )
```

Definition at line 611 of file tests.cpp.

```
612 {
613     test_real_time_calculation("((c((360-4+2*2)/2))N+4)!", {6, 7, 18, 19, 21, 23}, {0.8090169943, 1,-1,
      1, 5, 120});
614 }
```

6.12.2.137 TEST_F() [81/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test64  )
```

Definition at line 616 of file tests.cpp.

```
617 {
618     test_real_time_calculation("((s90*(22DN)^(23DN))N)*(-1)N", {4, 8, 9, 10, 15, 16, 17, 20, 27}, {1,
      22, 2, -2,-8388608, 4, 0.25, -0.25, -0.25});
619 }
```

6.12.2.138 TEST_F() [82/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test65  )
```

Definition at line 621 of file tests.cpp.

```
622 {
623     test_real_time_calculation("(-(c45*(s30-t60))N)/3^((2*c0+2*s90)/2)", {17, 37}, {-0.8711914807,
      -0.0967990534});
624 }
```

6.12.2.139 TEST_F() [83/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test66  )
```

Definition at line 626 of file tests.cpp.

```
627 {
628     test_real_time_calculation("((6.5-sin90+-(-cos0+6.5)))N*2+5DDDDDDDDDDDDDDDDDDDDDDDDDDD1",
629                                {26, 29, 31, 32, 34, 39, 40, 41, 43, 44, 45, 47, 50, 51, 54, 56, 57, 59,
      60},
630                                {2, -4, 1, -4, -2, -2.5, -2.5, -8.5, -7.5, -7.5, -7.5, -7.5, -6.5, -6.5,
      -6, 0, 0, -1});
631 }
```

6.12.2.140 TEST_F() [84/119]

```
TEST_F (
              Real_time_calc ,
              Real_time_calc_test67  )
```

Definition at line 633 of file tests.cpp.

```
634 {
635     test_real_time_calculation("(((3)))^2DDDDD1-+-5)*2-50)/2)/4NDDDDDDNDDDDDDNDDDDDD",
636                          {8, 10, 12, 14, 18, 21, 24, 27, 30, 31, 32, 36, 38, 40, 43, 45, 49, 51},
637                          {9, 3, 3, 31, 26, 52, 2, 1, 0.25, -0.25, 1, 2, -2, -47, -26, 26, 31, 0});
638 }
```

6.12.2.141 TEST_F() [85/119]

```
TEST_F (
              Factorial_tests ,
              Factorial_tests1  )
```

Definition at line 810 of file tests.cpp.

```
811 {
812     EXPECT_TRUE(fact_test(5, 120, NO));
813 }
```

6.12.2.142 TEST_F() [86/119]

```
TEST_F (
              Factorial_tests ,
              Factorial_tests2  )
```

Definition at line 815 of file tests.cpp.

```
816 {
817     EXPECT_TRUE(fact_test(-4, 24, YES));
818 }
```

6.12.2.143 TEST_F() [87/119]

```
TEST_F (
              Factorial_tests ,
              Factorial_tests3  )
```

Definition at line 820 of file tests.cpp.

```
821 {
822     EXPECT_TRUE(fact_test(0, 1, NO));
823 }
```

6.12.2.144 TEST_F() [88/119]

```
TEST_F (
              Factorial_tests ,
              Factorial_tests4  )
```

Definition at line 825 of file tests.cpp.

```
826 {
827     //EXPECT_TRUE(fact_test(21, -1, YES));
828     EXPECT_EQ(factorial(21), (double)5109094217170944e4);
829 }
```

6.12.2.145 TEST_F() [89/119]

```
TEST_F (
              Factorial_tests ,
              Factorial_tests5  )
```

Definition at line 831 of file tests.cpp.

```
832 {
833     EXPECT_TRUE(fact_test(-1, -1, YES));
834 }
```

6.12.2.146 TEST_F() [90/119]

```
TEST_F (
             Factorial_tests ,
             Factorial_tests6  )
```

Definition at line 836 of file tests.cpp.

```
837 {
838     EXPECT_TRUE(fact_test(-124, -1, YES));
839 }
```

6.12.2.147 TEST_F() [91/119]

```
TEST_F (
             Factorial_tests ,
             Factorial_tests7  )
```

Definition at line 841 of file tests.cpp.

```
842 {
843     EXPECT_DOUBLE_EQ(factorial(124), (double)1.506141741511140879795014161 99e207);
844 }
```

6.12.2.148 TEST_F() [92/119]

```
TEST_F (
             Npow_test ,
             Npow_test1  )
```

Definition at line 849 of file tests.cpp.

```
850 {
851     EXPECT_TRUE(pow_test(3.3, 2, 10.89, NO));
852 }
```

6.12.2.149 TEST_F() [93/119]

```
TEST_F (
             Npow_test ,
             Npow_test2  )
```

Definition at line 854 of file tests.cpp.

```
855 {
856     EXPECT_TRUE(pow_test(123.32442, 0, 1, NO));
857 }
```

6.12.2.150 TEST_F() [94/119]

```
TEST_F (
              Npow_test ,
              Npow_test3  )
```

Definition at line 859 of file tests.cpp.

```
860 {
861     EXPECT_TRUE(pow_test(-14.2, 0, 1, NO));
862 }
```

6.12.2.151 TEST_F() [95/119]

```
TEST_F (
              Npow_test ,
              Npow_test4  )
```

Definition at line 864 of file tests.cpp.

```
865 {
866     EXPECT_TRUE(pow_test(0, 0, 1, NO));
867 }
```

6.12.2.152 TEST_F() [96/119]

```
TEST_F (
              Npow_test ,
              Npow_test5  )
```

Definition at line 869 of file tests.cpp.

```
870 {
871     EXPECT_TRUE(pow_test(0.9999999999999, 2, 1, NO));
872 }
```

6.12.2.153 TEST_F() [97/119]

```
TEST_F (
              Npow_test ,
              Npow_test6  )
```

Definition at line 874 of file tests.cpp.

```
875 {
876     EXPECT_TRUE(pow_test(0.99, 4, 0.96059601, NO));
877 }
```

6.12.2.154 TEST_F() [98/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test1  )
```

Definition at line 882 of file tests.cpp.

```
883 {
884     EXPECT_TRUE(pow_test(5, 2, 25, NO));
885 }
```

6.12.2.155 TEST_F() [99/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test2  )
```

Definition at line 887 of file tests.cpp.

```
888 {
889     EXPECT_TRUE(pow_test(36.1556, 0, 1, NO));
890 }
```

6.12.2.156 TEST_F() [100/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test3  )
```

Definition at line 892 of file tests.cpp.

```
893 {
894     EXPECT_TRUE(pow_test(-14.2, 0.5, 1, YES));
895 }
```

6.12.2.157 TEST_F() [101/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test4  )
```

Definition at line 897 of file tests.cpp.

```
898 {
899     EXPECT_TRUE(pow_test(16, 0.25, 2, NO));
900 }
```

6.12.2.158 TEST_F() [102/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test5  )
```

Definition at line 902 of file tests.cpp.

```
903 {
904     EXPECT_TRUE(pow_test(4, 1.5, 8, NO));
905 }
```

6.12.2.159 TEST_F() [103/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test6  )
```

Definition at line 907 of file tests.cpp.

```
908 {
909     EXPECT_TRUE(pow_test(-5, 0.2, -1.37972966, NO));
910 }
```

6.12.2.160 TEST_F() [104/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test7  )
```

Definition at line 912 of file tests.cpp.

```
913 {
914     EXPECT_TRUE(pow_test(0.5, 3, 0.125, NO));
915 }
```

6.12.2.161 TEST_F() [105/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test8  )
```

Definition at line 917 of file tests.cpp.

```
918 {
919     EXPECT_TRUE(pow_test(-8, 2.5, 0, YES));
920 }
```

6.12.2.162 TEST_F() [106/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test9  )
```

Definition at line 922 of file tests.cpp.

```
923 {
924     EXPECT_TRUE(pow_test(4, -3, 0.015625, YES));
925 }
```

6.12.2.163 TEST_F() [107/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test10  )
```

Definition at line 927 of file tests.cpp.

```
928 {
929     EXPECT_TRUE(pow_test(0.2, -5, 3125, YES));
930 }
```

6.12.2.164 TEST_F() [108/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test11  )
```

Definition at line 932 of file tests.cpp.

```
933 {
934     EXPECT_TRUE(pow_test(0.5, -3.4, 10.5560632, YES));
935 }
```

6.12.2.165 TEST_F() [109/119]

```
TEST_F (
              Xpow_test ,
              Xpow_test12  )
```

Definition at line 937 of file tests.cpp.

```
938 {
939     EXPECT_TRUE(pow_test(60, -3.2, 2.0413430699e-6, YES));
940 }
```

6.12.2.166 TEST_F() [110/119]

```
TEST_F (
              Nsin_test ,
              Sinx_test1  )
```

Definition at line 945 of file tests.cpp.

```
946 {
947     EXPECT_TRUE(sinx_test(0, 0, NO));
948     EXPECT_TRUE(sinx_test(30, 0.5, NO));
949     EXPECT_TRUE(sinx_test(90, 1, NO));
950     EXPECT_TRUE(sinx_test(150, 0.5, NO));
951     EXPECT_TRUE(sinx_test(180, 0, NO));
952     EXPECT_TRUE(sinx_test(210, -0.5, NO));
953     EXPECT_TRUE(sinx_test(270, -1, NO));
954     EXPECT_TRUE(sinx_test(360, 0, NO));
955 }
```

6.12.2.167 TEST_F() [111/119]

```
TEST_F (
              Nsin_test ,
              Sinx_test2  )
```

Definition at line 957 of file tests.cpp.

```
958 {
959     EXPECT_TRUE(sinx_test(45, 0.7071067811, NO));
960     EXPECT_TRUE(sinx_test(60, 0.8660254037, NO));
961     EXPECT_TRUE(sinx_test(120, 0.8660254037, NO));
962     EXPECT_TRUE(sinx_test(135, 0.7071067811, NO));
963 }
```

6.12.2.168 TEST_F() [112/119]

```
TEST_F (
              Nsin_test ,
              Sinx_test3  )
```

Definition at line 965 of file tests.cpp.

```
966 {
967     EXPECT_TRUE(sinx_test(15, 0.2588190451, NO));
968     EXPECT_TRUE(sinx_test(-42, -0.66913060, NO));
969     EXPECT_TRUE(sinx_test(-34.5, -0.566406235, NO));
970     EXPECT_TRUE(sinx_test(42.2, 0.6717205893, NO));
971 }
```

6.12.2.169 TEST_F() [113/119]

```
TEST_F (
              Nsin_test ,
              Sinx_test4  )
```

Definition at line 972 of file tests.cpp.

```
973 {
974     EXPECT_TRUE(sinx_test(90, 1.01, YES));
975     EXPECT_TRUE(sinx_test(270, -1.0001, YES));
976 }
```

6.12.2.170 TEST_F() [114/119]

```
TEST_F (
              Ncos_test ,
              Cosx_test1  )
```

Definition at line 982 of file tests.cpp.

```
983 {
984     EXPECT_TRUE(cosx_test(0, 1, NO));
985     EXPECT_TRUE(cosx_test(60, 0.5, NO));
986     EXPECT_TRUE(cosx_test(90, 0, NO));
987     EXPECT_TRUE(cosx_test(120, -0.5, NO));
988     EXPECT_TRUE(cosx_test(180, -1, NO));
989     EXPECT_TRUE(cosx_test(240, -0.5, NO));
990     EXPECT_TRUE(cosx_test(270, 0, NO));
991     EXPECT_TRUE(cosx_test(360, 1, NO));
992 }
```

6.12.2.171 TEST_F() [115/119]

```
TEST_F (
              Ncos_test ,
              Cosx_test2  )
```

Definition at line 994 of file tests.cpp.

```
995 {
996     EXPECT_TRUE(cosx_test(45, 0.7071067811, NO));
997     EXPECT_TRUE(cosx_test(30, 0.8660254037, NO));
998     EXPECT_TRUE(cosx_test(150, -0.8660254037, NO));
999     EXPECT_TRUE(cosx_test(135, -0.7071067811, NO));
1000     EXPECT_TRUE(cosx_test(210, -0.8660254037, NO));
1001 }
```

6.12.2.172 TEST_F() [116/119]

```
TEST_F (
              Ncos_test ,
              Cosx_test3  )
```

Definition at line 1003 of file tests.cpp.

```
1004 {
1005     EXPECT_TRUE(cosx_test(1337, -0.224951054, NO));
1006     EXPECT_TRUE(cosx_test(49, 0.6560590289, NO));
1007     EXPECT_TRUE(cosx_test(0.0001, 1, NO));
1008     EXPECT_TRUE(cosx_test(-3.14, 0.99849867, NO));
1009 }
```

6.12.2.173 TEST_F() [117/119]

```
TEST_F (
              Ncos_test ,
              Cosx_test4  )
```

Definition at line 1010 of file tests.cpp.

```
1011 {
1012     EXPECT_TRUE(cosx_test(0.000001, 1.001, YES));
1013     EXPECT_TRUE(cosx_test(179.9, -1.1, YES));
1014     EXPECT_TRUE(cosx_test(359.9, 1, YES));
1015 }
```

6.12.2.174 TEST_F() [118/119]

```
TEST_F (
              Ntan_test ,
              Tanx_test1  )
```

Definition at line 1020 of file tests.cpp.

```
1021 {
1022     EXPECT_TRUE(tanx_test(0, 0, NO));
1023     EXPECT_TRUE(tanx_test(30, 0.5773502691, NO));
1024     EXPECT_TRUE(tanx_test(45, 1, NO));
1025     EXPECT_TRUE(tanx_test(60, 1.7320508075, NO));
1026     EXPECT_TRUE(tanx_test(90, 0, YES));
1027     EXPECT_TRUE(tanx_test(270, 0, YES));
1028 }
```

6.12.2.175 TEST_F() [119/119]

```
TEST_F (
              Ntan_test ,
              Tanx_test2  )
```

Definition at line 1029 of file tests.cpp.

```
1030 {
1031     EXPECT_TRUE(tanx_test(32, 0.6248693519, NO));
1032     EXPECT_TRUE(tanx_test(-1180, 5.671281819, NO));
1033     EXPECT_TRUE(tanx_test(99, -6.3137515146, NO));
1034     EXPECT_TRUE(tanx_test(630, 0, YES));
1035 }
```

Index