IVS Calculator

1.0

Generated by Doxygen 1.9.1

Chapter 1

IVS Calculator

Developed in C++. User interface generated via Qt Creator.

1.1 Environment

Ubuntu 64bit

1.2 Authors

EEEEEEEEEEEEEEEEEEEE

- xmikoja00 Jakub Miko
- xjordan00 Nikola Jordanov
- xcontop00 Patrik Čontofalský
- xziklaa00 Alexander Žikla

1.3 Manual Instalation

Manual installation is done from the src folder. In order to manually install the program from the source code you need to run:

```
sudo make install
```

To uninstall it run:

```
sudo make uninstall
```

1.4 Installer

For the installer you need to visit https://github.com/Jakub-Miko/IVS_Kalkulacka.git. From the releases section download debpackage for the desired version. For the Calculator use the package ending with E-Calculator. For the stddev utility use the package ending with Profiler. Once the debpackage is downloaded run:

```
sudo apt install [DEBPACKAGE_NAME]
```

To uninstall the application run:

```
sudo apt remove ivs-calculator-e-calculator (for the calculator)
```

or sudo apt remove ivs-calculator-profiler (for the profiler)

1.5 License

LICENSE

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

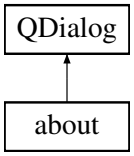
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 about Class Reference

Inheritance diagram for about:



Public Member Functions

- **about** (QWidget *parent=nullptr)

Private Slots

- void **on_pushButton_clicked** ()
- void **on_pushButton_doc_clicked** ()

Private Attributes

- Ui::about * **ui**

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

- about.h
- about.cpp

5.2 MathEngine::Context Struct Reference

The current status of the operation and result of a paranthesis.

`#include <math_engine.h>`

Public Attributes

- long double accumulator

  result value of the current context
- Operation last_op

  last operation initiated on the context

5.2.1 Detailed Description

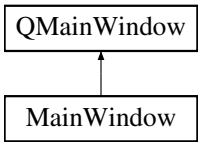
The current status of the operation and result of a paranthesis.

The documentation for this struct was generated from the following file:

- math_engine.h

5.3 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- **MainWindow** (QWidget *parent=nullptr)
- void PlaySound ()

  Play sound effect.

Private Slots

- void number_clicked ()

  Print a number on the display when button is clicked.
- void keyPressEvent (QKeyEvent *)

  Slot called when a key is pressed.
- void on_pushButton_backspace_clicked ()

  Remove the last character from the display.
- void on_pushButton_clearfull_clicked ()

  Reset the calculator (display, math engine).
- void on_pushButton_cleardisp_clicked ()

  Clear main display.
- void on_actionAbout_triggered ()

  Open new "About" window.
- void on_pushButton_mode_clicked ()

  Change mods of calculator.

- void on_pushButton_plus_clicked ()

  Addition operation.
- void on_pushButton_minus_clicked ()

  Substract operation.
- void on_pushButton_mul_clicked ()

  Multiplication operation.
- void on_pushButton_div_clicked ()

  Division operation.
- void on_pushButton_equals_clicked ()

  Equals operation.
- void on_pushButton_comma_clicked ()

  Print a comma on the display.
- void on_pushButton_root_clicked ()

  Root operation.
- void on_pushButton_power_clicked ()

  Power operation.
- void on_pushButton_abs_clicked ()

  Absolute value of number.
- void on_pushButton_pi_clicked ()

  Print a Pi constant on the display.
- void on_pushButton_c_clicked ()

  Print a c constant on the display.
- void on_pushButton_e_clicked ()

  Print a e constant on the display.
- void on_pushButton_factorial_clicked ()

  Calculate the factorial of the number.
- void on_pushButton_log_clicked ()

  Calculate the natural logarithm of the number.
- void on_pushButton_chngval_clicked ()

  Change the value of the number in the main display.
- void on_pushButton_sine_clicked ()

  Calculate the sine of the number.
- void on_pushButton_cosine_clicked ()

  Calculate the cosine of the number.
- void on_pushButton_tangent_clicked ()

  Calculate the tangent of the number.
- void on_pushButton_open_clicked ()

  Print an open paranthesis on the display when it is possible.
- void on_pushButton_close_clicked ()

  Print a close paranthesis on the display when it is possible.

Private Attributes

- QMediaPlayer * player

  Pointer to the sound player.
- QAudioOutput * output

  Pointer to the audio output.
- Ui::MainWindow * ui

  Pointer to the graphical interface.
- about * about_window

Pointer to the "About" window.

- bool mode_choice

  Variable indicating the mode choice.

- MathEngine math

  Instance of the math engine.

5.3.1 Member Function Documentation

5.3.1.1 keyPressEvent

```
void MainWindow::keyPressEvent (
            QKeyEvent * event )  [private], [slot]
```

Slot called when a key is pressed.

Parameters

event	Pointer to the key event.

```
154 {   //SWITCH CASE FOR NUMBERS
155     switch(event->key()) {
156     case Qt::Key_0:
157         AddNumber(ui, "0");
158         break;
159     case Qt::Key_1:
160         AddNumber(ui, "1");
161         break;
162     case Qt::Key_2:
163         AddNumber(ui, "2");
164         break;
165     case Qt::Key_3:
166         AddNumber(ui, "3");
167         break;
168     case Qt::Key_4:
169         AddNumber(ui, "4");
170         break;
171     case Qt::Key_5:
172         AddNumber(ui, "5");
173         break;
174     case Qt::Key_6:
175         AddNumber(ui, "6");
176         break;
177     case Qt::Key_7:
178         AddNumber(ui, "7");
179         break;
180     case Qt::Key_8:
181         AddNumber(ui, "8");
182         break;
183     case Qt::Key_9:
184         AddNumber(ui, "9");
185         break;
186     case Qt::Key_ParenLeft:
187         on_pushButton_open_clicked();
188         break;
189     case Qt::Key_ParenRight:
190         on_pushButton_close_clicked();
191         break;
192     case Qt::Key_Comma:
193     case Qt::Key_Period:
194         on_pushButton_comma_clicked();
195         break;
196     case Qt::Key_Backspace:
197     {
198         QString DisplayedWithoutLast = ui->display->text();
199         DisplayedWithoutLast.chop(1);
200         ui->display->setText(DisplayedWithoutLast);
201         break;
202     }
```

```
203     case Qt::Key_Delete:
204         ui->display->setText("");
205         break;
206     case Qt::Key_R:
207         on_pushButton_clearfull_clicked();
208         break;
209     }
210 
211     // SWITCH CASES FOR OPERATIONS
212     switch(event->key()) {
213     case Qt::Key_Plus:
214         on_pushButton_plus_clicked();
215         break;
216     case Qt::Key_Minus:
217         on_pushButton_minus_clicked();
218         break;
219     case Qt::Key_Asterisk:
220         on_pushButton_mul_clicked();
221         break;
222     case Qt::Key_Slash:
223         on_pushButton_div_clicked();
224         break;
225     case Qt::Key_Exclam:
226         on_pushButton_factorial_clicked();
227         break;
228     case Qt::Key_AsciiCircum:
229         on_pushButton_power_clicked();
230         break;
231     case Qt::Key_L:
232         on_pushButton_log_clicked();
233         break;
234     case Qt::Key_A:
235         on_pushButton_abs_clicked();
236         break;
237     case Qt::Key_Equal:
238     case Qt::Key_Enter:
239     case Qt::Key_Return:
240         on_pushButton_equals_clicked();
241         break;
242     }
243 }
```

5.3.1.2 on_pushButton_abs_clicked

```
void MainWindow::on_pushButton_abs_clicked ( )  [private], [slot]
```

Absolute value of number.

Operation based on the current state of the calculator and updates the display accordingly.

```
373 {
374     try {
375         SendNumberToEngine(ui, math);
376         math.SendAbsVal();
377         ShowResult(ui, math);
378     } catch(const std::runtime_error& err) {
379         QMessageBox::information(this, "Warning", err.what());
380         setWindowModality(Qt::ApplicationModal);
381     }
382 }
```

5.3.1.3 on_pushButton_cosine_clicked

```
void MainWindow::on_pushButton_cosine_clicked ( )  [private], [slot]
```

Calculate the cosine of the number.

Unary operation based on the current state of the calculator and updates the display accordingly.

```
463 {
```

```
464     try {
465         SendNumberToEngine(ui, math);
466         math.SendCosine();
467         ShowResult(ui, math);
468     } catch(const std::runtime_error& err) {
469         QMessageBox::information(this, "Warning", err.what());
470         setWindowModality(Qt::ApplicationModal);
471     }
472 }
```

5.3.1.4 on_pushButton_div_clicked

```
void MainWindow::on_pushButton_div_clicked ( )  [private], [slot]
```

Division operation.

Binary operation based on the current state of the calculator and updates the display accordingly.

```
303 {
304     try {
305         if(math.GetContextStack().back().last_op == MathEngine::Operation::DEFAULT) {
306             math.SendNumber(1);
307         }
308         SendNumberToEngine(ui, math);
309         math.SendDivide();
310         ShowResult(ui, math);
311     } catch(const std::runtime_error& err) {
312         QMessageBox::information(this, "Warning", err.what());
313         setWindowModality(Qt::ApplicationModal);
314     }
315 }
```

5.3.1.5 on_pushButton_equals_clicked

```
void MainWindow::on_pushButton_equals_clicked ( )  [private], [slot]
```

Equals operation.

Performs the current arithmetic operation based on the operands stored in the calculator's memory and displays the result.

```
319 {
320     try {
321         SendNumberToEngine(ui, math);
322         math.SendEquals();
323         ShowResult(ui, math);
324     } catch(const std::runtime_error& err) {
325         QMessageBox::information(this, "Warning", err.what());
326             setWindowModality(Qt::ApplicationModal);
327     }
328 }
```

5.3.1.6 on_pushButton_factorial_clicked

```
void MainWindow::on_pushButton_factorial_clicked ( )  [private], [slot]
```

Calculate the factorial of the number.

Unary operation based on the current state of the calculator and updates the display accordingly.

```
386 {
387     try {
388         SendNumberToEngine(ui, math);
389         const char* msg = nullptr;
390         if((msg = math.SendFactorial().msg)) {
391             QMessageBox::information(nullptr, "Warning", msg);
392         }
393         ShowResult(ui, math);
394     } catch(const std::runtime_error& err) {
395         QMessageBox::information(this, "Warning", err.what());
396         setWindowModality(Qt::ApplicationModal);
397     }
398 }
```

5.3.1.7 on_pushButton_log_clicked

```
void MainWindow::on_pushButton_log_clicked ( )  [private], [slot]
```

Calculate the natural logarithm of the number.

Unary operation based on the current state of the calculator and updates the display accordingly.

```
402 {
403     try {
404         SendNumberToEngine(ui, math);
405         math.Sendln();
406         ShowResult(ui, math);
407     } catch(const std::runtime_error& err) {
408         QMessageBox::information(this, "Warning", err.what());
409         setWindowModality(Qt::ApplicationModal);
410     }
411 }
```

5.3.1.8 on_pushButton_minus_clicked

```
void MainWindow::on_pushButton_minus_clicked ( )  [private], [slot]
```

Substract operation.

Binary operation based on the current state of the calculator and updates the display accordingly.

```
274 {
275     try {
276         SendNumberToEngine(ui, math);
277         math.SendSubtract();
278         ShowResult(ui, math);
279     } catch(const std::runtime_error& err) {
280         QMessageBox::information(this, "Warning", err.what());
281         setWindowModality(Qt::ApplicationModal);
282     }
283 }
```

5.3.1.9 on_pushButton_mode_clicked

```
void MainWindow::on_pushButton_mode_clicked ( )  [private], [slot]
```

Change mods of calculator.

Changes between functions mode and constant mode.

```
253 {
254     ui->MOD_container->setCurrentIndex(mode_choice);
255     if(mode_choice) mode_choice = false;
256     else mode_choice = true;
257 }
```

5.3.1.10 on_pushButton_mul_clicked

```
void MainWindow::on_pushButton_mul_clicked ( )  [private], [slot]
```

Multiplication operation.

Binary operation based on the current state of the calculator and updates the display accordingly.

```
287 {
288     try {
289         if(math.GetContextStack().back().last_op == MathEngine::Operation::DEFAULT) {
290             math.SendNumber(1);
291         }
292         SendNumberToEngine(ui, math);
293         math.SendMultiply();
294         ShowResult(ui, math);
295     } catch(const std::runtime_error& err) {
296         QMessageBox::information(this, "Warning", err.what());
297         setWindowModality(Qt::ApplicationModal);
298     }
299 }
```

5.3.1.11 on_pushButton_plus_clicked

```
void MainWindow::on_pushButton_plus_clicked ( )  [private], [slot]
```

Addition operation.

Binary operation based on the current state of the calculator and updates the display accordingly.

```
261 {
262     try {
263         SendNumberToEngine(ui, math);
264         math.SendAdd();
265         ShowResult(ui, math);
266     } catch(const std::runtime_error& err) {
267         QMessageBox::information(this, "Warning", err.what());
268         setWindowModality(Qt::ApplicationModal);
269     }
270 }
```

5.3.1.12 on_pushButton_power_clicked

```
void MainWindow::on_pushButton_power_clicked ( )  [private], [slot]
```

Power operation.

Binary operation based on the current state of the calculator and updates the display accordingly.

```
359 {
360     try {
361         SendNumberToEngine(ui, math);
362         math.SendPower();
363         ShowResult(ui, math);
364     } catch(const std::runtime_error& err) {
365         QMessageBox::information(this, "Warning", err.what());
366         setWindowModality(Qt::ApplicationModal);
367     }
368
369 }
```

5.3.1.13 on_pushButton_root_clicked

```
void MainWindow::on_pushButton_root_clicked ( )  [private], [slot]
```

Root operation.

Binary operation based on the current state of the calculator and updates the display accordingly.

```
346 {
347     try {
348         SendNumberToEngine(ui, math);
349         math.SendRoot();
350         ShowResult(ui, math);
351     } catch(const std::runtime_error& err) {
352         QMessageBox::information(this, "Warning", err.what());
353         setWindowModality(Qt::ApplicationModal);
354     }
355 }
```

5.3.1.14 on_pushButton_sine_clicked

```
void MainWindow::on_pushButton_sine_clicked ( )  [private], [slot]
```

Calculate the sine of the number.

Unary operation based on the current state of the calculator and updates the display accordingly.

```
450 {
451     try {
452         SendNumberToEngine(ui, math);
453         math.SendSine();
454         ShowResult(ui, math);
455     } catch(const std::runtime_error& err) {
456         QMessageBox::information(this, "Warning", err.what());
457         setWindowModality(Qt::ApplicationModal);
458     }
459 }
```

5.3.1.15 on_pushButton_tangent_clicked

```
void MainWindow::on_pushButton_tangent_clicked ( )  [private], [slot]
```

Calculate the tangent of the number.

Unary operation based on the current state of the calculator and updates the display accordingly.

```
476 {
477     try {
478         SendNumberToEngine(ui, math);
479         math.SendTangent();
480         ShowResult(ui, math);
481     } catch(const std::runtime_error& err) {
482         QMessageBox::information(this, "Warning", err.what());
483         setWindowModality(Qt::ApplicationModal);
484     }
485 }
```

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

- mainwindow.h
- mainwindow.cpp

5.4 MathEngine Class Reference

Abstraction of UI math logic.

`#include <math_engine.h>`

Classes

- struct Context

  The current status of the operation and result of a paranthesis.
- struct ReturnCode

Public Types

- enum class Operation {
  DEFAULT , **RESULT** , **ADD** , **SUBTRACT** ,
  MULTIPLY , **DIVIDE** , **FACTORIAL** , **LN** ,
  ABSVAL , **POWER** , **ROOT** }

  Type defining last active mathematical operation.
- enum class **Status** { **OK** , **ROUNDING** }

Public Member Functions

- **MathEngine** (const MathEngine &ref)=delete
- **MathEngine** (MathEngine &&ref)=delete
- MathEngine & **operator=** (const MathEngine &ref)=delete
- MathEngine & **operator=** (MathEngine &&ref)=delete
- ReturnCode SendNumber (long double number)

  Submit number entered by the user.
- ReturnCode SendEquals ()

  After user clicked the equals sign, calculate the result of the whole expression.
- void SendAdd ()

  User clicked the add button.
- void SendSubtract ()

  User clicked the subtract button.
- void SendMultiply ()

  User clicked the multiply button.
- void SendPower ()

  User clicked the Power button Be Aware the exponent can only be an argument so rounding may occur (can be checked by the return value of send number)
- void SendRoot ()

  User clicked the Root button Be Aware the exponent can only be an argument so rounding may occur (can be checked by the return value of send number)
- void SendDivide ()

  User clicked the divide button.
- ReturnCode SendFactorial ()

  User clicked the factorial button. Be Aware Factorial only takes integer arguments so rounding may occur.
- void SendLn ()

  User clicked the ln button.
- void SendAbsVal ()

User clicked the ABS button.

- void SendSine ()

  User clicked the SIN button.

- void SendCosine ()

  User clicked the COS button.

- void SendTangent ()

  User clicked the TAN button.

- long double GetAccumulator () const

  Get the current value of the accumulator (result from the current paranthesis)

- bool IsAccumulatorResult () const

  The value of the accumulator is the result, and no other operation is currently in progress.

- bool IsResultAvailable () const

  All stacks have been calculated and the accumulator contains the final result.

- void ClearAccumulator ()

  Clear the current accumulator and reset the operation to default.

- void ResetAllContexts ()

  Clear all the contexts (paranthesis)

- void StartContext ()

  Start a paranthesis.

- ReturnCode EndContext ()

  End a paranthesis.

- const std::vector< Context > & GetContextStack () const

  Get the context stack, needed to display the equation.

- std::string GetDisplay () const

  Gets the result display containg all pending(unclosed) contexts(paranthesis)

Private Attributes

- std::vector< Context > context_stack

  Stack of paranthesis (Context), enabling nested paranthesis.

Static Private Attributes

- static const char * **op_symbols** [ ]

5.4.1 Detailed Description

Abstraction of UI math logic.

5.4.2 Member Function Documentation

5.4.2.1 EndContext()

```
MathEngine::ReturnCode MathEngine::EndContext ( )
```

End a paranthesis.

Returns

```
237 {
238     // After the paranthesis ends, we take the result of the paranthesis, and restore the original
      operation
239     // before the paranthesis, supplying the paranthesis result as an input
240     if(context_stack.size() <= 1) {
241         throw std::runtime_error("Can't pop default context");
242     }
243     long double accumulator_temp = context_stack.back().accumulator;
244     context_stack.pop_back();
245     ReturnCode code = SendNumber(accumulator_temp);
246     context_stack.back().last_op = Operation::RESULT;
247     return code;
248 }
```

5.4.2.2 GetAccumulator()

```
long double MathEngine::GetAccumulator ( ) const
```

Get the current value of the accumulator (result from the current paranthesis)

Returns

accumulator value

```
192 {
193     return context_stack.back().accumulator;
194 }
```

5.4.2.3 GetContextStack()

```
const std::vector< MathEngine::Context > & MathEngine::GetContextStack ( ) const
```

Get the context stack, needed to display the equation.

Returns

context stack Context

```
251 {
252     return context_stack;
253 }
```

5.4.2.4 IsAccumulatorResult()

```
bool MathEngine::IsAccumulatorResult ( ) const
```

The value of the accumulator is the result, and no other operation is currently in progress.

Returns

Whether no operation is in progress on the current accumulator

```
197 {
198     return context_stack.back().last_op == Operation::RESULT;
199 }
```

5.4.2.5 IsResultAvailable()

```
bool MathEngine::IsResultAvailable ( ) const
```

All stacks have been calculated and the accumulator contains the final result.

Returns

Whether the accumulator contains the final result (combination of all the paranthesis)

```
202 {
203     // For a complete result, the accumulator result needs to be the only one (all paranthesis need to
      be closed)
204     return (context_stack.size() == 1) && (context_stack.back().last_op == Operation::RESULT);
205 }
```

5.4.2.6 SendFactorial()

```
MathEngine::ReturnCode MathEngine::SendFactorial ( )
```

User clicked the factorial button. Be Aware Factorial only takes integer arguments so rounding may occur.

Returns

Status indicating whether rounding happened withing this operation

```
131 {
132     if(context_stack.back().accumulator < 0) throw std::runtime_error("Factorial of negative numbers not
      allowed");
133     // Unary operations only need the accumulator, so their calculation doesnt need to be deffered
134     ReturnCode code;
135     if(!CheckConversion(context_stack.back().accumulator)) {
136         code.status = Status::ROUNDING;
137         code.msg = "Rounding occured in the Factorial operation, because non integer value was
      provided";
138     }
139     context_stack.back().accumulator = Factorial(context_stack.back().accumulator);
140     // After the calculation is complete, since no other operand is awaited
141     // we can consider the value to be a complete result
142     context_stack.back().last_op = Operation::RESULT;
143     return code;
144 }
```

5.4.2.7 SendNumber()

```
MathEngine::ReturnCode MathEngine::SendNumber (
            long double number )
```

Submit number entered by the user.

Parameters

number	the number submitted by the user

Returns

Status indicating whether rounding happened withing the last operation

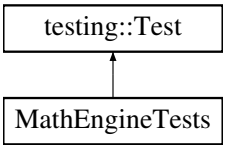
```
36 {
37     // based on the last submitted operation, perform a calculation with the accumulator and the input
       number
38     // and then save the result back into the accumulator
39     long double result = 0;
40     long double accumulator = context_stack.back().accumulator;
41     MathEngine::ReturnCode status;
42     switch (context_stack.back().last_op)
43     {
44     case Operation::ADD:
45         result = Add(accumulator, number);
46         break;
47     case Operation::SUBTRACT:
48         result = Sub(accumulator, number);
49         break;
50     case Operation::MULTIPLY:
51         result = Mult(accumulator, number);
52         break;
53     case Operation::DIVIDE:
54         result = Div(accumulator, number);
55         break;
56     case Operation::POWER:
57         if(number < 0) throw std::runtime_error("Exponent cant be negative");
58         if(!CheckConversion(number)) {
59             status.status = Status::ROUNDING;
60             status.msg = "Rounding occured in the Power operation, because non integer exponent was
       provided";
61         }
62         result = Power(accumulator, number);
63         break;
64     case Operation::ROOT:
65         if(!CheckConversion(accumulator)) {
66             status.status = Status::ROUNDING;
67             status.msg = "Rounding occured in the Root operation, because non integer exponent was
       provided";
68         }
69         result = Root(number, accumulator);
70         break;
71     case Operation::RESULT:
72     case Operation::DEFAULT:
73         result = number;
74         break;
75     default:
76         throw std::runtime_error("Unsupported mathematical operation");
77         break;
78     }
79     context_stack.back().accumulator = result;
80     return status;
81 }
```

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

- math_engine.h
- math_engine.cpp

5.5 MathEngineTests Class Reference

Inheritance diagram for MathEngineTests:



Public Attributes

- MathEngine **engine**

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

- MathEngine_tests.cpp

5.6 MathEngine::ReturnCode Struct Reference

Public Attributes

- Status **status** = Status::OK
- const char * **msg** = nullptr

The documentation for this struct was generated from the following file:

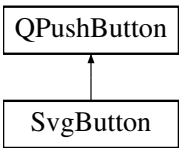
- math_engine.h

5.7 SvgButton Class Reference

Custom PushButton for svg rendering with hover effects.

`#include <svgbutton.h>`

Inheritance diagram for SvgButton:



Signals

- void **pathChanged** ()
- void **color_default_Changed** ()
- void **color_hover_Changed** ()

Public Member Functions

- **SvgButton** (QWidget *parent=nullptr)
- void SetPath (QString path)
- bool **event** (QEvent *event) override
- QString **path** () const
- void **setPath** (const QString &newPath)
- QColor **color_default** () const
- void **setColor_default** (const QColor &newColor_default)
- QColor **color_hover** () const
- void **setColor_hover** (const QColor &newColor_hover)

Protected Member Functions

- virtual void **paintEvent** (QPaintEvent ∗event) override
- virtual void **resizeEvent** (QResizeEvent ∗event) override

Properties

- QString **path**
- QColor **color_default**
- QColor **color_hover**

Private Member Functions

- void **EnterHover** ()
- void **LeaveHover** ()
- void **SetColor** (const QVariant &color)

Private Attributes

- QString m_path

  path to the rendered SVG
- QColor m_color_default

  the idle color of the widget
- QColor m_color_hover

  the hovered color of the widget
- QSvgRenderer ∗ svg_renderer

  SvgRenderer instance for painting svg image to image.
- QColor target_color

  the current color (set by the animation)
- QImage image

  the canvas containing the rendered content to display
- QVariantAnimation animation

  used for handling animations

5.7.1 Detailed Description

Custom PushButton for svg rendering with hover effects.

While Qt does support SvgWidgets, it doesnt allow for manipulating color.

5.7.2 Member Function Documentation

5.7.2.1 SetPath()

```
void SvgButton::SetPath (
            QString path )
```

< margin used for maintaining the aspect ration

```
31 {
32     svg_renderer->load(QString(":/resources/") + this->path()); // Loads the svg from the Resource file
33     int margin = std::abs(width() - height()) / 2;
34     QRect res;
35     if(width() >height()) {
36         res = QRect(margin ,0, height(), height());
37 
38     } else {
39         res = QRect(0 ,margin, width(), width());
40 
41     }
42     QImage image(res.size(),QImage::Format_A2BGR30_Premultiplied);
43     image.fill(Qt::GlobalColor::transparent); // Clear the canvas
44     QPainter painter(&image);
45     painter.setRenderHint(QPainter::Antialiasing , true);
46     painter.setRenderHint(QPainter::SmoothPixmapTransform,true);
47 
48     painter.setCompositionMode(QPainter::CompositionMode_SourceOver); // make sure both alpha and color
       channels are written
49     svg_renderer->setAspectRatioMode(Qt::AspectRatioMode::KeepAspectRatio);
50     svg_renderer->render(&painter,QRectF(0 ,0, res.width(), res.height())); // render the svg
51     this->image = image;
52     painter.end();
53 }
```

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

- svgbutton.h
- svgbutton.cpp

Chapter 6

File Documentation

6.1 about.cpp File Reference

Representation of the "about app" window of the application.

```
#include "about.h"
#include "qdesktopservices.h"
#include "ui_about.h"
#include <QStandardPaths>
```

6.1.1 Detailed Description

Representation of the "about app" window of the application.

This class represents the "about application" window of the application. It contains informations about license, authors, documentation and user manual.

Author

Patrik Contofalsky, Jakub Miko

6.2 about.h File Reference

Representation of the "about app" window of the application.

```
#include <QDialog>
```

Classes

- class about

6.2.1 Detailed Description

Representation of the "about app" window of the application.

This class represents the "about application" window of the application. It contains informations about license, authors, documentation and user manual.

Author

Patrik Contofalsky, Jakub Miko

6.3 main.cpp File Reference

Entry point of the application.

```
#include "mainwindow.h"
#include <iostream>
#include <QApplication>
#include <QFontDatabase>
```

Functions

- int **main** (int argc, char ∗argv[ ])

6.3.1 Detailed Description

Entry point of the application.

Author

Patrik Contofalsky, Jakub Miko

6.4 mainwindow.cpp File Reference

Representation of the main window of the application.

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include <QKeyEvent>
#include <QMediaDevices>
#include <QAudioDevice>
#include <QMessageBox>
#include "mathlibrary.h"
```

Functions

- void AddNumber (Ui::MainWindow *ui, QString NextNumber)

  Add a number to the display.
- QString ReplaceString (QString Text, QString Find, QString Replace)

  Replaces a part of string with another string.
- void SendNumberToEngine (Ui::MainWindow *ui, MathEngine &math)

  Sends the current number on the display to the math engine.
- void ShowResult (Ui::MainWindow *ui, MathEngine &math)

  Shows the result of the calculation in the equation display.

6.4.1 Detailed Description

Representation of the main window of the application.

This class represents the main window of the application. It contains functionalities for using the math_engine and interacting with the user interface.

Author

Patrik Contofalsky, Jakub Miko

6.4.2 Function Documentation

6.4.2.1 AddNumber()

```
void AddNumber (
            Ui::MainWindow * ui,
            QString NextNumber )
```

Add a number to the display.

Adds the clicked number to the display widget.

Parameters

ui	Pointer to the main window's user interface.
NextNumber	QString number to add to the display.

```
80                                                  {
81     QString LabelNumber = (ui->display->text() + NextNumber);
82     ui->display->setText(LabelNumber);
83 }
```

6.4.2.2 ReplaceString()

```
QString ReplaceString (
```

```
            QString Text,
            QString Find,
            QString Replace )
```

Replaces a part of string with another string.

Parameters

Text	QString of the original string.
Find	QString of the substring to find.
Replace	QString of the substring to replace with.

Returns

QString The modified string.

```
93                                                                  {
94     Text.replace(Find, Replace);
95     return Text;
96 }
```

6.4.2.3 SendNumberToEngine()

```
void SendNumberToEngine (
            Ui::MainWindow * ui,
            MathEngine & math )
```

Sends the current number on the display to the math engine.

Sends the current number on the display to the math engine for further processing.

Parameters

ui	Pointer to the main window's user interface.
math	Reference to the math engine instance.

```
106                                                                 {
107     if (!ui->display->text().isEmpty()) {
108         long double ld = std::strtold(ReplaceString(ui->display->text(), ",",
       ".").toLatin1().data(),nullptr);
109         const char* msg = nullptr;
110         if((msg = math.SendNumber(ld).msg)) {
111             QMessageBox::information(nullptr, "Warning", msg);
112         }
113     }
114 }
```

6.4.2.4 ShowResult()

```
void ShowResult (
            Ui::MainWindow * ui,
            MathEngine & math )
```

Shows the result of the calculation in the equation display.

Returns the result of the calculation from the math engine and displays it in the equation display.

Parameters

ui	Pointer to the main window's user interface.
math	Reference to the math engine instance.

```
125                                                 {
126     ui->display->setText("");
127
128     ui->equation->setText(math.GetDisplay().c_str());
129
130 }
```

6.5 mainwindow.h File Reference

Representation of the main window of the application.

```
#include <QMainWindow>
#include "about.h"
#include "qaudiooutput.h"
#include "qmediaplayer.h"
#include "math_engine.h"
```

Classes

- class MainWindow

6.5.1 Detailed Description

Representation of the main window of the application.

This class represents the main window of the application. It contains functionalities for using the math_engine and interacting with the user interface.

Author

Patrik Contofalsky, Jakub Miko

6.6 math_engine.cpp File Reference

Abstraction of UI math logic.

```
#include "math_engine.h"
#include <mathlibrary.h>
#include <stdexcept>
#include <cmath>
#include <sstream>
#include <iomanip>
```

Macros

- #define **ROUNDING_EPSILON** 1.0e-15

Functions

- bool CheckConversion (long double input)

  Checks if conversion from double to int doesnt cause information loss (ignoring floating point error)
- std::string **ConvertDigit** (long double digit)

6.6.1 Detailed Description

Abstraction of UI math logic.

Author

Jakub Miko

6.6.2 Function Documentation

6.6.2.1 CheckConversion()

```
bool CheckConversion (
            long double input )
```

Checks if conversion from double to int doesnt cause information loss (ignoring floating point error)

Returns

false if rounding occur; true if conversion was withing accuracy margin defined by ROUNDING_EPSILON

```
18                                      {
19     std::uint64_t converted = input;
20     long double input_reverted = converted;
21     if(AbsVal(input - input_reverted) > ROUNDING_EPSILON) {
22         return false;
23     }
24     return true;
25 }
```

6.7 math_engine.h File Reference

Abstraction of UI math logic.

```
#include <vector>
#include <string>
```

Classes

- class MathEngine

  Abstraction of UI math logic.
- struct MathEngine::ReturnCode
- struct MathEngine::Context

  The current status of the operation and result of a paranthesis.

Macros

- #define **PRECISION_OF_NUMBER** 12

6.7.1 Detailed Description

Abstraction of UI math logic.

Author

Jakub Miko

6.8 mathlib_additional_tests.cpp File Reference

implementation of additional tests fro the math library

```
#include "mathlibrary.h"
#include <gtest/gtest.h>
```

Functions

- **TEST** (FACTORIAL, OF_Check)
- **TEST** (LN, functionality)
- **TEST** (ABSVAL, functionality)

6.8.1 Detailed Description

implementation of additional tests fro the math library

Author

Nikola Jordanov

6.9 mathlib_goniometry_test.cpp File Reference

implementation of tests for goniometry functions

```
#include "mathlibrary.h"
#include <math.h>
#include <gtest/gtest.h>
```

Functions

- **TEST** (SIN, suite)
- **TEST** (COSINE, suite)
- **TEST** (TANGENT, suite)

6.9.1 Detailed Description

implementation of tests for goniometry functions

Author

Nikola Jordanov

6.10 mathlib_tdd_tests.cpp File Reference

implementation of test driven development tests

```
#include "mathlibrary.h"
#include <gtest/gtest.h>
```

Functions

- **TEST** (ADD, natural_numbers)
- **TEST** (ADD, whole_numbers)
- **TEST** (ADD, real_numbers)
- **TEST** (SUB, natural_numbers)
- **TEST** (SUB, whole_numbers)
- **TEST** (SUB, real_numbers)
- **TEST** (MULT, natural_numbers)
- **TEST** (MULT, whole_numbers)
- **TEST** (MULT, real_numbers)
- **TEST** (DIV, natural_nubers)
- **TEST** (DIV, whole_numbers)
- **TEST** (DIV, real_numbers)
- **TEST** (FACTORIAL, suite)
- **TEST** (POWER, suite)
- **TEST** (ROOT, suite)

6.10.1 Detailed Description

implementation of test driven development tests

Author

Nikola Jordanov

6.11 mathlibrary.h File Reference

Library for functions mathematical functions used by the calculator.

```
#include <cstdint>
```

Functions

- long double Add (long double a, long double b)
- long double Sub (long double a, long double b)
- long double Mult (long double a, long double b)
- long double Div (long double a, long double b)
- std::uint64_t Factorial (std::uint64_t n)
- long double Power (long double a, std::uint64_t n)
- long double Root (long double a, std::uint64_t n)
- long double ln (long double a)
- long double AbsVal (long double a)
- long double Sine (long double a)
- long double Cosine (long double a)
- long double Tangent (long double a)

Variables

- constexpr double constants::const_pi = 3.14159265358979323846
- constexpr double constants::const_e = 2.7182818284590452354
- constexpr double constants::const_h = 6.62607015e-34
- constexpr double constants::const_k = 1.380649e-23
- constexpr double constants::const_light = 299792458.0

6.11.1 Detailed Description

Library for functions mathematical functions used by the calculator.

6.11.2 Function Documentation

6.11.2.1 AbsVal()

```
long double AbsVal (
            long double a )
```

Returns

Positive number

```
84 {
85     return std::abs(a);
86 }
```

6.11.2.2 Add()

```
long double Add (
            long double a,
            long double b )
```

Parameters

a	First number
b	Second number

Returns

Sum of 2 numbers

```
6 {
7     return a+b;
8 }
```

6.11.2.3 Cosine()

```
long double Cosine (
            long double a )
```

Returns

Cosine of argument value

```
94 {
95     return std::cos(a);
96 }
```

6.11.2.4 Div()

```
long double Div (
            long double a,
            long double b )
```

Parameters

a	Numerator
b	Denominator

Returns

Division of 2 numbers

```
21 {
22     if (b == 0) {
23         throw std::runtime_error("Can't divide by zero");
24     }
25     return a/b;
26 }
```

6.11.2.5 Factorial()

```
std::uint64_t Factorial (
            std::uint64_t n )
```

Parameters

n	(natural) Number

Returns

Multiplication of numbers from 1 to "n"

```
29 {
30     if (n == 0 || n == 1) {
31         return 1;
32     }
33     std::uint64_t result = 1;
34     while (n > 1) {
35         if (UINT64_MAX / result < n) {
36             throw std::overflow_error("The number is too large to get the factorial");
37         }
38         result *= n;
39         n--;
40     }
41     return result;
42 }
```

6.11.2.6 ln()

```
long double ln (
            long double a )
```

Parameters

a	Positive number

Returns

Natural logarithm of number "a"

```
76 {
77     if (a <= 0) {
78         throw std::runtime_error("Input must be a positive number");
79     }
80     return std::log(a);
81 }
```

6.11.2.7 Mult()

```
long double Mult (
            long double a,
            long double b )
```

Parameters

a	First factor
b	Second factor

Returns

Multiplication of 2 numbers

```
16 {
17     return a*b;
18 }
```

6.11.2.8 Power()

```
long double Power (
              long double a,
              std::uint64_t n )
```

Parameters

a	Base number
n	Exponent (natural number)

Returns

"a" to the power of "n"

```
45 {
46     if (n == 0) {
47         return 1;
48     }
49     long double result = 1;
50     while (n > 0) {
51         result *= a;
52         n--;
53         if (std::isinf(result)) {
54             throw std::runtime_error("Result is infinite.");
55         }
56     }
57     return result;
58 }
```

6.11.2.9 Root()

```
long double Root (
              long double a,
              std::uint64_t n )
```

Parameters

a	Radicand (If "n" is even, "a" has to be positive)
n	Index (natural number)

Returns

n-th root of number "a"

```
61 {
62     if ((n%2) == 0 && (a < 0)) {
```

```
63          throw std::runtime_error("Negative number cannot have even root");
64      }
65      if (n == 0) {
66          throw std::runtime_error("Root exponent cannot be zero");
67      }
68      if (a > 0) {
69          return std::pow(a, 1.0 / static_cast<long double>(n));
70      } else {
71           return -std::pow(-a, 1.0 / static_cast<long double>(n));
72      }
73 }
```

6.11.2.10 Sine()

```
long double Sine (
              long double a )
```

Returns

Sine of argument value

```
89 {
90     return std::sin(a);
91 }
```

6.11.2.11 Sub()

```
long double Sub (
              long double a,
              long double b )
```

Parameters

a	Minuend
b	Subtrahend

Returns

Subtraction of 2 numbers

```
11 {
12     return a-b;
13 }
```

6.11.2.12 Tangent()

```
long double Tangent (
              long double a )
```

Returns

Tangent of argument value

```
99 {
100     return std::tan(a);
101 }
```

6.11.3 Variable Documentation

6.11.3.1 const_e

```
constexpr double constants::const_e = 2.7182818284590452354  [constexpr]
```

Euler's number

6.11.3.2 const_h

```
constexpr double constants::const_h = 6.62607015e-34  [constexpr]
```

Planck's constant

6.11.3.3 const_k

```
constexpr double constants::const_k = 1.380649e-23  [constexpr]
```

Boltzmann's constant

6.11.3.4 const_light

```
constexpr double constants::const_light = 299792458.0  [constexpr]
```

Speed of light (m/s)

6.11.3.5 const_pi

```
constexpr double constants::const_pi = 3.14159265358979323846  [constexpr]
```

Ludolf's number

6.12 stddev.cpp File Reference

Calculation of sample standard deviation.

```
#include <iostream>
#include <vector>
#include "mathlibrary.h"
```

Functions

- long double CalculateMean (const std::vector< long double > &arr)
- long double CalculateSampleStandardDeviation (const std::vector< long double > &arr)
- int **main** ()

6.12.1 Detailed Description

Calculation of sample standard deviation.

Author

Alexander Žikla

6.12.2 Function Documentation

6.12.2.1 CalculateMean()

```
long double CalculateMean (
            const std::vector< long double > & arr )
```

Parameters

arr	Vector of long double numbers

Returns

Arithmetic mean

```
18 {
19     long double sum = 0;
20     for (long double number : arr) {
21         sum = Add(sum, number);
22     }
23     return Div(sum, arr.size());
24 }
```

6.12.2.2 CalculateSampleStandardDeviation()

```
long double CalculateSampleStandardDeviation (
            const std::vector< long double > & arr )
```

Parameters

arr	Vector of long double numbers

Returns

Sample Standard Deviation

```
31 {
32     long double mean = CalculateMean(arr);
33     long double deviation = 0;
34     for (long double number : arr) {
35         deviation = Add(deviation, Power(Sub(number, mean), 2) );
36     }
37     deviation = Div(deviation, arr.size());
38     return Root(deviation, 2);
39 }
```

6.13 svgbutton.cpp File Reference

Custom PushButton for svg rendering with hover effects.

```
#include "svgbutton.h"
#include <QtSvgWidgets/QtSvgWidgets>
#include <QMediaPlayer>
#include <QAudioOutput>
```

6.13.1 Detailed Description

Custom PushButton for svg rendering with hover effects.

Author

Jakub Miko

6.14 svgbutton.h File Reference

Custom PushButton for svg rendering with hover effects.

```
#include <QObject>
#include <QPushButton>
#include <qsvgrenderer.h>
#include <qvariantanimation.h>
```

Classes

- class SvgButton

  Custom PushButton for svg rendering with hover effects.

6.14.1 Detailed Description

Custom PushButton for svg rendering with hover effects.

Author

Jakub Miko

Index