Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем та програмної інженерії

Кафедра програмної інженерії

ЗВІТ

До лабораторної роботи №2.8

“Створення графічного інтерфейсу користувача додатку засобами Qt, проєктування GUI.Віджети і менеджери компонування”

з навчальної дисципліни

“Об’єктно-орієнтоване програмування”

Підготував:

студент групи СП-21

Бурило Вадим

Тернопіль 2023

**Мета:**

* Отримати навички з створення графічного інтерфейсу користувача додатку засобами Qt

IDE: Visual Studio Code

Посилання на повний проєкт:

Завдання Калькулятор:

Загальний рівень:

Калькулятор повинен виконувати стандартні операції “+”, “-”, “\*”, “/”, “%” і операції з пам’яттю, також повинен містити кнопку скидання даних (очищення поля вводу).

Підвищений рівень:

Потрібно виконати вимоги загального рівня з додатковими вимогами:

додати функції “інженерного” калькулятора: степінь, корінь, тригонометричні функції тощо (приклад - стандартний калькулятор Windows);

додати функції конвертування величин;

Розв’язання завдання:

Файл Calculation.h

Лістинг коду:

#include <string>

#include <vector>

using namespace std;

#pragma once

class Calculation

{

    private:

        vector<string> operations = {"+", "-", "\*", "/", "ln", "%", "lg", "tan", "ctan", "sin", "cos", "sqrt", "not", "^"};

    public:

        Calculation();

        ~Calculation();

        string eval(string expression);

};

Файл Calculation.cpp

Лістинг коду:

#include "Calculation.h"

#include <vector>

#include <string>

#include <iostream>

#include <valarray>

using namespace std;

Calculation::Calculation()

{

}

Calculation::~Calculation()

{

}

string Calculation::eval(string expression)

{

    int c\_operations = 0;

    string operation;

    int operation\_index;

    bool is\_percent = false;

    for(int i = 0; i < operations.size(); i++) {

        // if operations[i] is in expression

        if(expression.find(operations[i]) != string::npos) {

            c\_operations++;

            operation = operations[i];

            operation\_index = expression.find(operations[i]);

            if (operations[i] == "%") {

                is\_percent = true;

            }

        }

    }

    if(c\_operations == 0) {

        return "Cannot solve this expression!";

    }

    if(operation == "not"){

        // like not232 => -232

        int first\_number = stoi(expression.substr(3, expression.length() - 1));

        return to\_string(-first\_number);

    }

    else if(operation == "ln"){

        // like ln-232

        int first\_number = stoi(expression.substr(2, expression.length() - 1));

        return to\_string(log(first\_number));

    }

    else if(operation == "lg"){

        // like log-232

        int first\_number = stoi(expression.substr(2, expression.length() - 1));

        return to\_string(log10(first\_number));

    }

    else if(operation == "tan"){

        // like tan-232

        int first\_number = stoi(expression.substr(3, expression.length() - 1));

        return to\_string(tan(first\_number));

    }

    else if(operation == "ctan"){

        // like ctan-232

        int first\_number = stoi(expression.substr(4, expression.length() - 1));

        return to\_string(1 / tan(first\_number));

    }

    else if(operation == "sin"){

        // like sin-232

        int first\_number = stoi(expression.substr(3, expression.length() - 1));

        return to\_string(sin(first\_number));

    }

    else if(operation == "cos"){

        // like cos-232

        int first\_number = stoi(expression.substr(3, expression.length() - 1));

        return to\_string(cos(first\_number));

    }

    else if(operation == "sqrt"){

        // like sqrt-232

        int first\_number = stoi(expression.substr(4, expression.length() - 1));

        return to\_string(sqrt(first\_number));

    }

    else if(operation == "^"){

        // like 2^3

        int first\_number = stoi(expression.substr(0, operation\_index));

        int second\_number = stoi(expression.substr(operation\_index + 1, expression.length() - 1));

        return to\_string(static\_cast<int>(pow(first\_number, second\_number)));

    }

    else if(operation == "%"){

        // like 2%3

        int first\_number = stoi(expression.substr(0, operation\_index));

        int second\_number = stoi(expression.substr(operation\_index + 1, expression.length() - 1));

        return to\_string(first\_number % second\_number);

    }

    else if(operation == "+"){

        // like 2+3

        int first\_number = stoi(expression.substr(0, operation\_index));

        int second\_number = stoi(expression.substr(operation\_index + 1, expression.length() - 1));

        return to\_string(first\_number + second\_number);

    }

    else if(operation == "-"){

        // like 2-3

        int first\_number = stoi(expression.substr(0, operation\_index));

        int second\_number = stoi(expression.substr(operation\_index + 1, expression.length() - 1));

        return to\_string(first\_number - second\_number);

    }

    else if(operation == "\*"){

        // like 2\*3

        int first\_number = stoi(expression.substr(0, operation\_index));

        int second\_number = stoi(expression.substr(operation\_index + 1, expression.length() - 1));

        return to\_string(first\_number \* second\_number);

    }

    else if(operation == "/"){

        // like 2/3

        int first\_number = stoi(expression.substr(0, operation\_index));

        int second\_number = stoi(expression.substr(operation\_index + 1, expression.length() - 1));

        return to\_string(first\_number / second\_number);

    }

    return "None";

}

Файл calcwindow.h

Лістинг коду:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "Calculation.h"

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

    Q\_OBJECT

public:

    MainWindow(QWidget \*parent = nullptr);

    ~MainWindow();

private:

    QString last\_result;

    Calculation calc;

    Ui::MainWindow \*ui;

    void set\_last\_result(QString result);

private slots:

    void append\_result();

    void button\_controls\_clicked();

    void on\_meters\_changed();

    void on\_kilometers\_changed();

};

#endif // MAINWINDOW\_H

Файл calcwindow.cpp

Лістинг коду:

#include "calcwindow.h"

#include "./ui\_calcwindow.h"

#include <QMessageBox>

#include <QPushButton>

#include "Calculation.h"

#include <iostream>

using namespace std;

bool is\_in\_cannot\_repeat(vector<string> v, string s)

{

    for(int i = 0; i < v.size(); i++) {

        if (v[i] == s) {

            return true;

        }

    }

    return false;

}

MainWindow::MainWindow(QWidget \*parent)

    : QMainWindow(parent)

    , ui(new Ui::MainWindow)

{

    ui->setupUi(this);

    // DIGITS

    connect(ui->button\_0, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_1, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_2, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_3, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_4, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_5, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_6, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_7, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_8, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_9, &QPushButton::clicked, this, &MainWindow::append\_result);

    // OPERATIONS

    connect(ui->button\_plus, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_minus, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_mult, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_divide, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_ln, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_lg, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_tan, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_ctan, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_sin, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_cos, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_perc, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_sqrt, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_pow, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_equals, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_dot, &QPushButton::clicked, this, &MainWindow::append\_result);

    connect(ui->button\_not, &QPushButton::clicked, this, &MainWindow::append\_result);

    // OTHER

    connect(ui->button\_clear, &QPushButton::clicked, this, &MainWindow::button\_controls\_clicked);

    connect(ui->button\_restore, &QPushButton::clicked, this, &MainWindow::button\_controls\_clicked);

    connect(ui->button\_backspace, &QPushButton::clicked, this, &MainWindow::button\_controls\_clicked);

    // CONVERTER

    connect(ui->meters, &QLineEdit::textChanged, this, &MainWindow::on\_meters\_changed);

    connect(ui->kilometers, &QLineEdit::textChanged, this, &MainWindow::on\_kilometers\_changed);

}

MainWindow::~MainWindow()

{

    delete ui;

}

void MainWindow::set\_last\_result(QString result)

{

    this->last\_result = result;

}

void MainWindow::append\_result()

{

    QPushButton \*button = (QPushButton \*)sender();

    // Отримуємо текст кнопки

    QString buttonText = button->text();

    if(buttonText == "="){

        string result = ui->result->text().toStdString();

        string res = calc.eval(result);

        ui->result->setText(QString::fromStdString(res));

        this->set\_last\_result(QString::fromStdString(res));

        return;

    }

    // check if text in cannot\_repeat

    // string to QString

    QString result = ui->result->text();

    // if(buttonText == "="){

    //     calc.eval(result.toStdString());

    // }

    // if prev char is in cannot\_repeat and current char is in cannot\_repeat

    result += buttonText;

    ui->result->setText(result);

}

void MainWindow::button\_controls\_clicked()

{

    QPushButton \*button = (QPushButton \*)sender();

    // Отримуємо текст кнопки

    QString buttonText = button->text();

    QString result = ui->result->text();

    if (buttonText == "Clear") {

        result = "";

    } else if (buttonText == "Memory") {

        result = this->last\_result;

    } else if (buttonText == "Backspace") {

        result = result.left(result.length() - 1);

    }

    ui->result->setText(result);

}

void MainWindow::on\_meters\_changed()

{

    QString meters = ui->meters->text();

    QString kilometers = QString::number(meters.toDouble() / 1000);

    ui->kilometers->setText(kilometers);

}

void MainWindow::on\_kilometers\_changed()

{

    QString kilometers = ui->kilometers->text();

    QString meters = QString::number(kilometers.toDouble() \* 1000);

    ui->meters->setText(meters);

}

Файл main.cpp

Лістинг коду:

#include "calcwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

    QApplication a(argc, argv);

    MainWindow w;

    w.show();

    return a.exec();

}

Файл CMakeLists.txt

Лістинг коду:

cmake\_minimum\_required(VERSION 3.15.0)

project(calculator VERSION 0.1.0 LANGUAGES C CXX)

set(Qt6\_DIR "C:/Qt/6.6.1/mingw\_64/lib/cmake/Qt6")

set(CMAKE\_CXX\_STANDARD 17)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_AUTOUIC ON)

set(CMAKE\_AUTOMOC ON)

set(CMAKE\_AUTORCC ON)

if(CMAKE\_VERSION VERSION\_LESS "3.7.0")

    set(CMAKE\_INCLUDE\_CURRENT\_DIR ON)

endif()

find\_package(Qt6 REQUIRED COMPONENTS Widgets)

set(PROJECT\_SOURCES

        main.cpp

        calcwindow.cpp

        calcwindow.h

        Calculation.cpp

        Calculation.h

        calcwindow.ui

)

add\_executable(calculator

    ${PROJECT\_SOURCES}

)

target\_link\_libraries(calculator PRIVATE Qt6::Widgets)

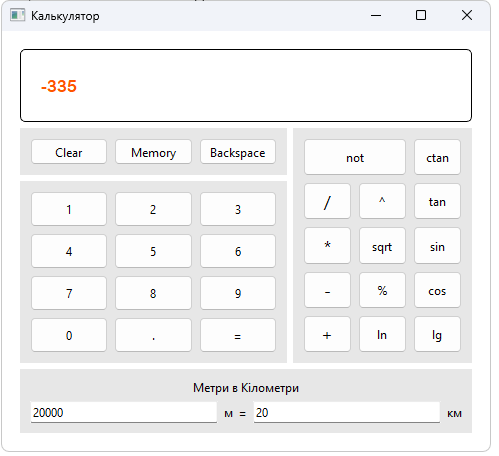
set(CPACK\_PROJECT\_NAME ${PROJECT\_NAME})

set(CPACK\_PROJECT\_VERSION ${PROJECT\_VERSION})

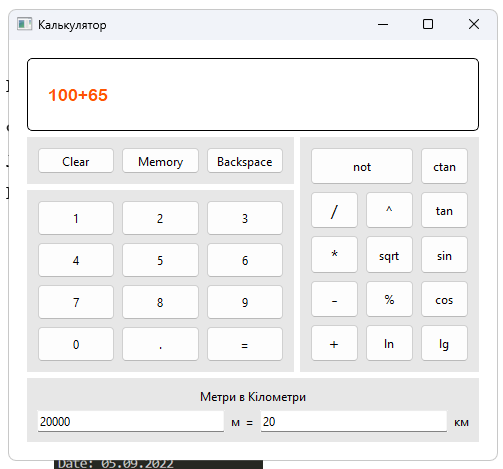
include(CPack)

Результат виконання завдання:

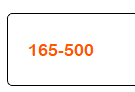
1. Зовнішній вигляд



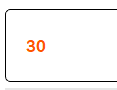
1. Додавання

 =>

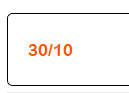
1. Віднімання

 => 

1. Множення

 => 

1. Ділення

 => 

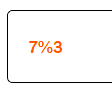
1. Піднесення до степеню

 => 

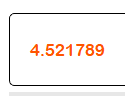
1. Корінь

 => 

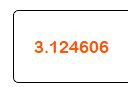
1. Остача від ділення

 => 

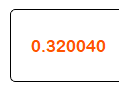
1. Натуральний логарифм

 => 

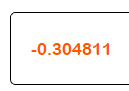
1. Котангенс

 => 

1. Тангенс

 =>

1. Синус

 => 

1. Косинус

 => 

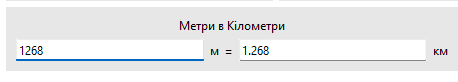
1. Логарифм з основою 10

 => 

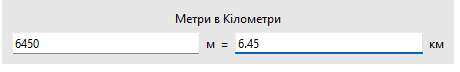
1. Операція not, not 30 = -30, not -30 = 30

 => 

1. Конвертування метрів у кілометри



1. Конвертування кілометрів у метри



Висновок:

* Отримав навички з створення графічного інтерфейсу користувача додатку засобами Qt