Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення


**Звіт**
Про виконання лабораторних робіт №1-4

**На тему:**

«Середовище програмування»
з дисципліни «Об'єктно-орієнтоване програмування»

**Лектор:**
доцент каф. ПЗ
Коротєєва Т.О.
**Виконав:**
ст. гр. ПЗ-11
Морозов О. Р.
**Прийняла:**
доцент каф. ПЗ
Коротєєва Т.О.

« __ » _____ 2022 р.

∑ = _____ _____ .

Львів – 2022

**Тема:** Ознайомлення з середовищем розробки Qt. Створення проекту та налаштування його властивостей.
**Мета:** Засвоїти принцип візуального програмування шляхом створення та налаштування проекту.

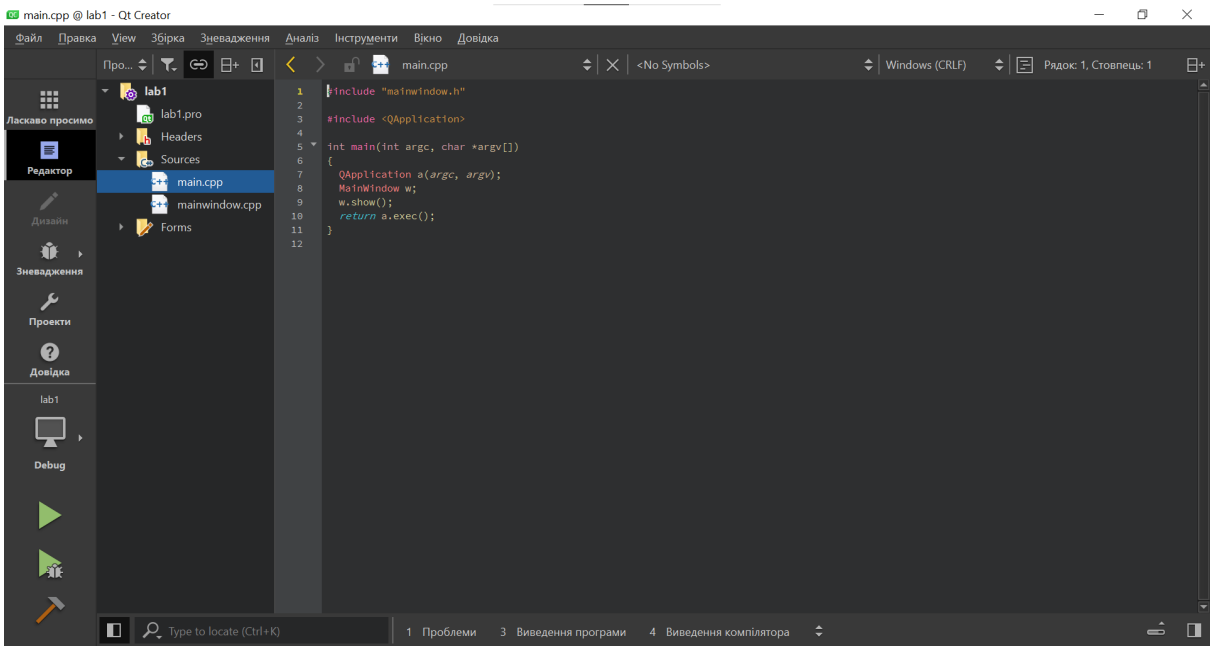
# ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Середовище розробки Qt можна умовно поділити на три частини:
   1) Навігаційне меню зліва
   2) Вікно огляду складових проекту
   3) Вікно редактора коду.

Навігаційне меню складається з п'яти вкладок, які слугують для переміщення між розділами середовища, його налаштування чи переходу в режим «зневадження» проекту. Знизу знаходяться кнопки для взаємодії з програмою, а саме: Запуск в звичайному режимі, запуск у режимі «зневадження» та збірка проекту.

Вікно огляду складових дає змогу розглянути всі компоненти, програми та перейти до роботи з ними. Стандартний проект Qt складається із файлу збірки (Example.pro), header-файлу, двох .cpp файлів та UI-форми, яка слугує для розробки інтерфейсу.

Вікно редактора коду відображає код програми та дає змогу користувачу працювати з ним.

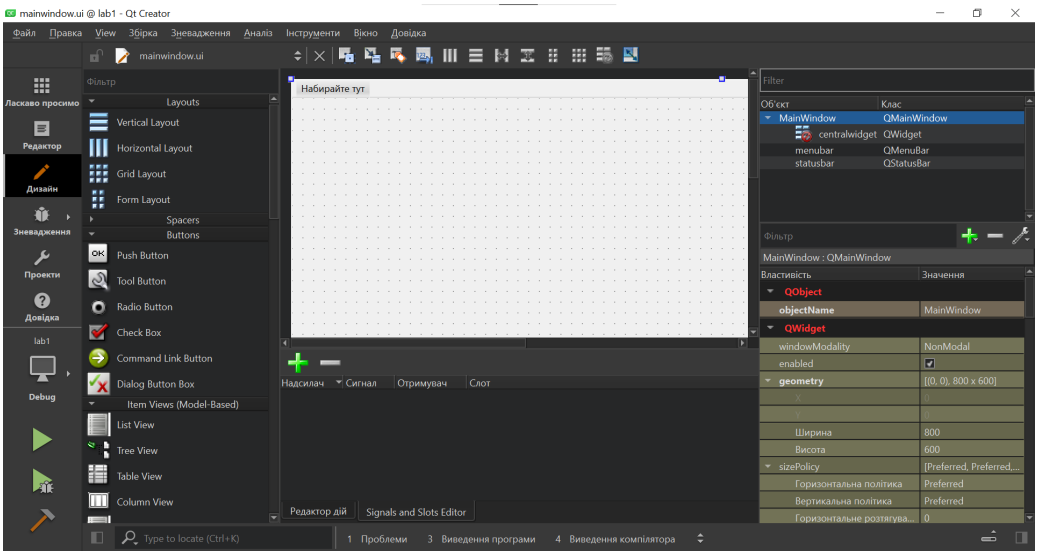


## 1.2 Організація проекту в Qt

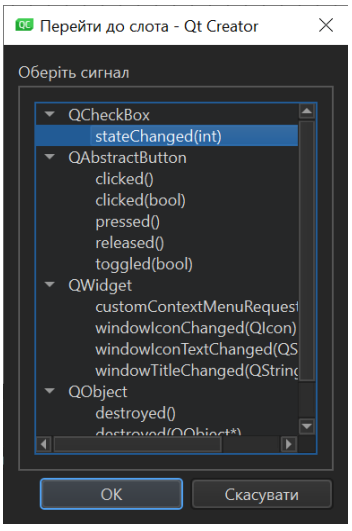
***Mainwindow.ui*** є файлом інтерфейсу проекту. У вікні редактора форми містяться: форма вікна, інспектор об'єктів, вікно властивостей об'єкта, список об'єктів.

Список об'єктів поділений на категорії, залежно від використання.
**Layouts** – використовується для впорядкованого розміщення об’єктів на формі.
**Spacers** – дозволяє розмежовувати структури на формі.
**Buttons** – кнопки та елементи взаємодії користувача з програмою.

Будь-який об’єкт має свої властивості, які вказані у відведеному вікні. Воно дає змогу переглядати та змінювати характеристики об’єктів та форми. При редагуванні візуальних властивостей складових інтерфейсу використовують **CSS Stylesheets**, що надає більші можливості в плані стилізації.



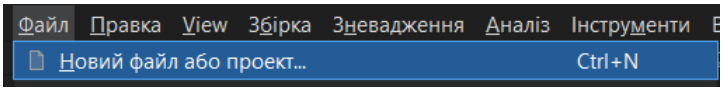

Об’єкти можна розмістити на формі перетягнувши їх. Після цього можна змінити його властивості або перейти до слота.



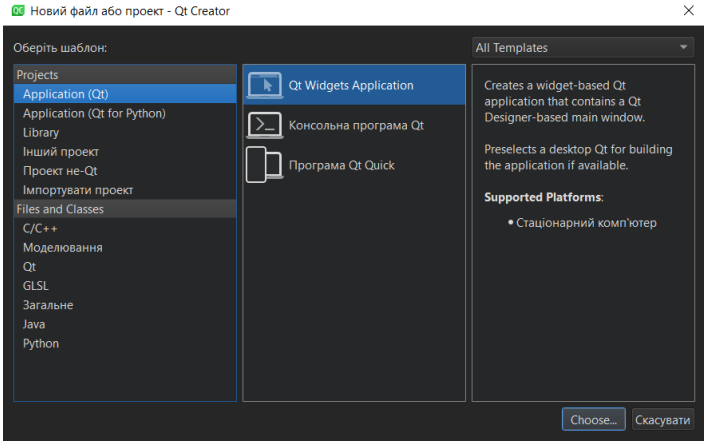

Зробивши це, ми обираємо тип сигналу та переходимо до редактора коду, де й описуємо дії, які виконуватиме програма у разі взаємодії користувача з даним об’єктом.

```
void MainWindow::on_checkBox_stateChanged(int arg1)
{

}
```

### 1.3 Створення та збереження нового проекту.



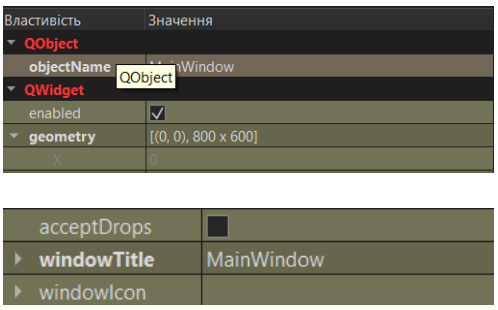

Новий проект можна створити натиснувши на вкладку Файл Новий файл або проект або ж використати комбінацію клавіш **Ctrl+N**.




Щоб зберегти проект, потрібно перейти до вкладки Файл Save All або використати комбінацію **Ctrl+Shift+S**.

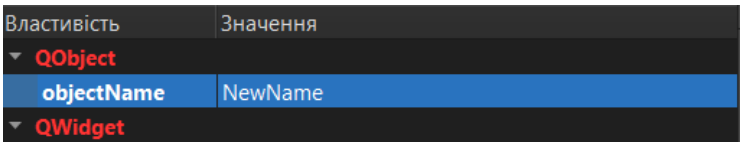

# ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Ознайомитись із середовищем Qt.
2. Створити новий проект. Зберегти його двома способами – через комбінації швидких клавіш та через меню.
3. Проглянути у вікні інспектора об'єктів властивості форми. Змінити назву форми та її розміри.

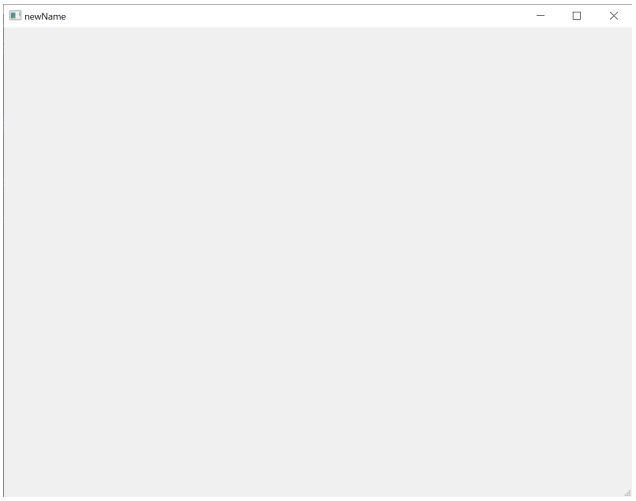



Назву форми можна змінити двома способами: у полі **ObjectName** та в полі **WindowTitle**. Проте в першому випадку ми змінюємо саме назву вікна, як об'єкта, тому при подальшій роботі з кодом, повинні будемо звертатись до нього саме за цим іменем. У другому зміна імені не впливатиме на роботу з кодом.

Виконання:








1. Запустити на виконання застосування.

2. Відкрити опції проекту, змінити налаштування на вкладках **Application, Compiler, Packages**. Запустити на виконання застосування.

# ВИСНОВКИ

Виконавши лабораторну роботу №1, я ознайомився із середовищем розробки **Qt Creator** та засвоїв принцип створення та налаштування віконних проектів.

# ЛАБОРАТОРНА РОБОТА №2

**Тема:** Базові візуальні компоненти Qt Creator. Створення проекту із використанням візуальних компонент.
**Мета:** Створити віконний проект та продемонструвати використання візуальних компонент Qt Creator.

# ТЕОРЕТИЧНІ ВІДОМОСТІ

Середовище **Qt Creator** містить набори візуальних компонент, які дозволяють створювати користувацький інтерфейс для взаємодії з програмою. Крім цього, є можливість створити власну компоненту і розмістити її на закладці, імпортувати набір компонент сторонніх розробників, змінювати розташування закладок.

Компоненти додаються на форму способом їх перетягування на робочу область.

Компонента **QPushButton** – це клас, який ідентифікує візуальну компоненту типу «кнопка». Ця кнопка має визначені типи взаємодії: натиснути, клацнути, натиснути й відпустити тощо. Об'єкти мають властивості, деякі з них: колір, колір тексту, обрамлення, його товщина і заокруглення. Об'єкти також можуть вміщувати текст, який, у свою чергу має й свої властивості.

Компонента **QLabel** – це клас, об'єктом якого є область, яка вміщує текстовий напис. Цей клас дозволяє також розміщувати в об'єкті графічні зображення, оскільки безпосереднє вставлення зображень не передбачене у Qt Creator. Основною функцією таких об'єктів є відображення тексту. Він може бути введений безпосередньо у редакторі форм, чи за допомогою команд у редакторі коду, проте не може бути змінений способом вводу під час виконання програми. QLabel, як і кожен **QWidget** має властивості: колір, колір тексту, обрамлення тощо.

Компонента **QLineEdit** – це клас, об'єкти якого за функціоналом схожі до **QLabel**, проте володіючи всіма його властивостями, **QLineEdit** дозволяє змінювати текст у собі, в ході виконання програми.

# ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Ознайомитись із палітрою компонент **Qt Creator**.
2. Створити віконний проект, додати розглянуті візуальні компоненти.
3. Реалізувати калькулятор.

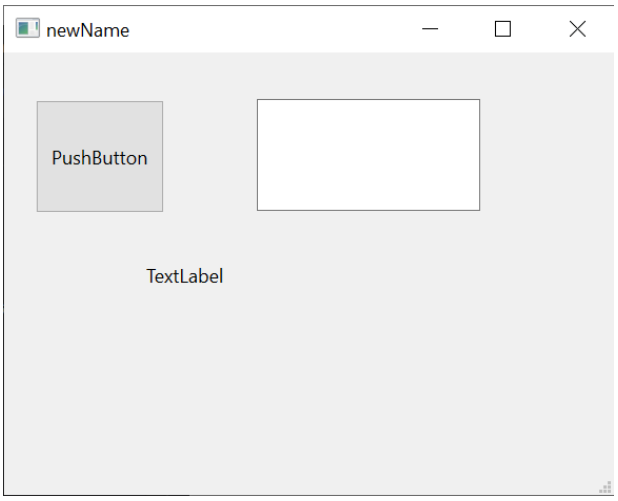



Рис.1 Об’єкти, розміщені на формі

## Реалізація калькулятора


**UI-Форма**

**mainwindow.h**

```
#ifndef MainWindow_H
#define MainWindow_H

#include <QMainWindow>
#include <QWidget>
#include <QString>

#include <cmath>

class QPushButton;

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
 Q_OBJECT

public:
 MainWindow(QWidget *parentWidget = nullptr);
 ~MainWindow();
 void calculate();

private:
 Ui::MainWindow *ui;
```

```
private slots:
 void digits();
 void operations();
 void math_operations();
 void on_clearButton_clicked();
 void on_pointButton_clicked();
 void on_resultButton_clicked();
};
#endif // MAINWINDOW_H
```

**mainwindow.cpp**

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent)  , ui(new
Ui::MainWindow) {
  ui->setupUi(this);

  connect(ui->numButton_0, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_1, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_2, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_3, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_4, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_5, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_6, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_7, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_8, SIGNAL(clicked()), this, SLOT(digits()));
  connect(ui->numButton_9, SIGNAL(clicked()), this, SLOT(digits()));

  connect(ui->changeButton, SIGNAL(clicked()), this, SLOT(operations()));
  connect(ui->sqrtButton, SIGNAL(clicked()), this, SLOT(operations()));

  connect(ui->plusButton, SIGNAL(clicked()), this, SLOT(math_operations()));
  ui->plusButton->setCheckable(true);
  connect(ui->minusButton, SIGNAL(clicked()), this, SLOT(math_operations()));
  ui->minusButton->setCheckable(true);
  connect(ui->multipleButton, SIGNAL(clicked()), this, SLOT(math_operations()));
  ui->multipleButton->setCheckable(true);
  connect(ui->divideButton, SIGNAL(clicked()), this, SLOT(math_operations()));
  ui->divideButton->setCheckable(true);

}

MainWindow::~MainWindow()
{
```

```
 delete ui;
}

void MainWindow::digits() {
 QPushButton *pbutton = (QPushButton *)sender();
 QString strNum;
 if(ui->lbl->text().contains('.') && pbutton->text()=="0") {
    ui->lbl->text()+pbutton->text();
  }
 else {
    strNum = ui->lbl->text()+pbutton->text();
 }
 ui->lbl->setText(QString::number((strNum).toDouble()));//
}
void MainWindow::operations() {
 QString str = ((QPushButton *)sender())->text();
 double numbers=(ui->lbl->text()).toDouble();

 if(str == "+/-") {
   numbers=numbers * (-1);
   ui->lbl->setText(QString::number(numbers));
 }
 else if(str == "√" && numbers > 0) {
    numbers=sqrt(numbers);
    ui->lbl->setText(QString::number(numbers));
    ui->lbl_2->clear();
 }
 else {
    ui->lbl_2->setText("Error");
    ui->lbl->setText("click CE");
  }

}

void MainWindow::math_operations() {

 //if(!(ui->divideButton->isChecked() ||
ui->multipleButton->isChecked()||ui->minusButton->isChecked()||ui->plusButton-
>isChecked())){
    QPushButton *pbutton = (QPushButton *)sender();
    ui->lbl_2->setText((ui->lbl->text())+pbutton->text());
    ui->lbl->setText("0");
    pbutton->setChecked(true);
  //}
}
```

```
void MainWindow::on_clearButton_clicked()
{
 ui->lbl->clear();
 ui->lbl->setText("0");
 ui->lbl_2->clear();
 ui->divideButton->setChecked(false);
 ui->multipleButton->setChecked(false);
 ui->minusButton->setChecked(false);
 ui->plusButton->setChecked(false);
}


void MainWindow::on_pointButton_clicked()
{
 if(!(ui->lbl->text().contains('.'))){
      ui->lbl->setText(ui->lbl->text()+".");
 }
}


void MainWindow::on_resultButton_clicked()
{
    double result, fnum, snum;
    fnum = ui->lbl_2->text().remove((ui->lbl_2->text().length())-1, 1).toDouble();
    snum = ui->lbl->text().toDouble();

  if(ui->plusButton->isChecked()) {
      ui->lbl_2->setText(ui->lbl_2->text()+ui->lbl->text());
      result = fnum+snum;
      ui->lbl->setText(QString::number(result));
      ui->plusButton->setChecked(false);
    }
  else if(ui->minusButton->isChecked()) {
      ui->lbl_2->setText(ui->lbl_2->text()+ui->lbl->text());
      result = fnum-snum;
      ui->lbl->setText(QString::number(result));
      ui->minusButton->setChecked(false);
    }
  else if(ui->divideButton->isChecked() && snum != 0) {
      ui->lbl_2->setText(ui->lbl_2->text()+ui->lbl->text());
      result = fnum/snum;
      ui->lbl->setText(QString::number(result));
      ui->divideButton->setChecked(false);
    }
```

```
  else if(ui->multipleButton->isChecked()) {
     ui->lbl_2->setText(ui->lbl_2->text()+ui->lbl->text());
     result = fnum*snum;
     ui->lbl->setText(QString::number(result));
     ui->multipleButton->setChecked(false);
    }
  else {
     ui->lbl_2->setText("Error");
     ui->lbl->setText("click CE");
    }
}
```

**main.cpp**

```
#include "mainwindow.h"

#include <QtWidgets>

int main(int argc, char *argv[])
{
 QApplication app(argc, argv);
 MainWindow calc;

 calc.show();

 return app.exec();
}
```

# ВИСНОВКИ

Виконавши лабораторну роботу №2, я ознайомився із візуальними компонентами середовища розробки **Qt Creator**, створив та налаштував віконний проект калькулятору з візуальними компонентами.

# ЛАБОРАТОРНА РОБОТА №3

**Тема:** Базові візуальні компоненти Qt Creator. Створення проекту із використанням невізуальних компонент.
**Мета:** Створити віконний проект та продемонструвати використання невізуальних компонент Qt Creator.

# ТЕОРЕТИЧНІ ВІДОМОСТІ

Невізуальні компоненти – це компоненти невидимі при виконанні програми, а у режимі конструювання відображаються у вигляді іконки. Такі компоненти можна розміщувати в будь-якому місці форми.
Основними невізуальними компонентами є системні діалоги. Існує всього 10 таких діалогів передбачених операційною системою **Microsoft Windows: OpenDialog, SaveDialog, OpenPictureDialog, SavePictureDialog, FontDialog, ColorDialog, PrintDialog, PrinterSetupDialog, FindDialog, ReplaceDialog.**
При виклику кожного з них з'являється нове діалогове вікно, і залежно від його типу, користувачу надаються певні можливості. Системні діалоги можуть бути викликані лише під час виконання програми.
Коротка характеристика:
**OpenDialog** – використовується для відкриття файлів.
**SaveDialog** - дає змогу зберігати файл з певним розширенням.
**PrintDialog** – дозволяє роздрукувати документ на папері.
І т.д.
У процесі роботи із системними діалогами, широко використовуються фільтри форматів. Вони задають параметри файлів, з якими взаємодіятиме діалог, встановлюючи дозволені розширення файлів.
Ще одним із видів невізуальних компонент є об'єкти класу **QMenuBar**. Вони розташовуються на верхній панелі віконного застосунку та виконують роль способу навігації функціями програми. Одним із різновидів цього меню є **PopUpMenu**, яке є додатковою вкладкою функцій, яка розгортається при наведенні на неї курсором.


Рис.1 Навігаційне меню

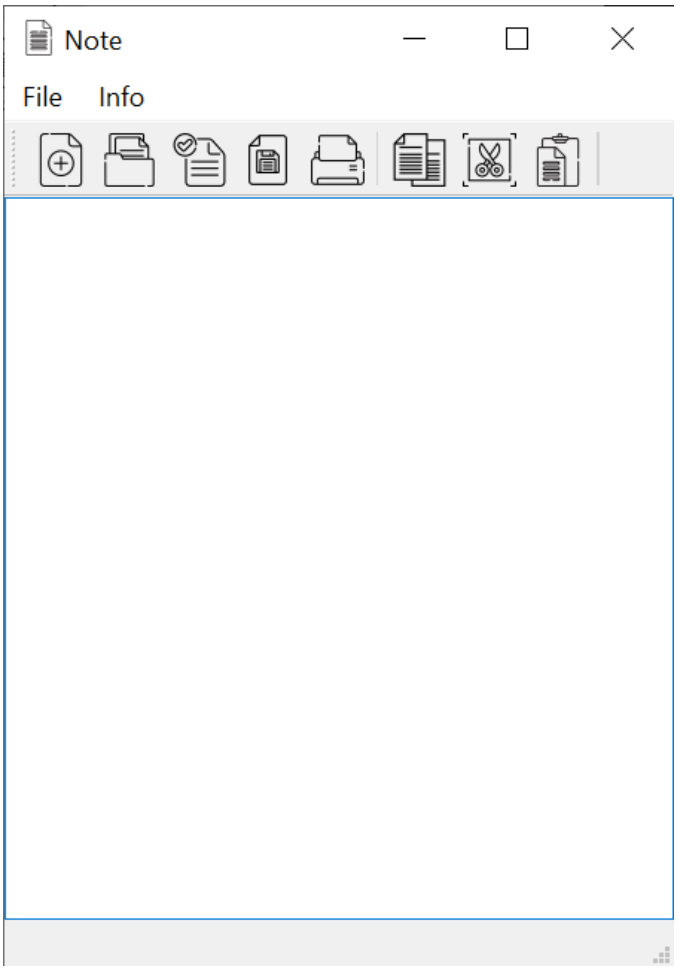
# ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Створити віконний проект. Додати головне та контекстне меню, необхідні системні діалоги.
2. Реалізувати текстовий редактор і переглядач графічних файлів

## Текстовий редактор

**note.h**

```
#ifndef NOTE_H
#define NOTE_H

#include <QMainWindow>

#include <QFile>
#include <QFileDialog>
#include <QTextStream>
#include <QMessageBox>
#include <QPrinter>
#include <QPrintDialog>

QT_BEGIN_NAMESPACE
namespace Ui { class Note; }
QT_END_NAMESPACE

class Note : public QMainWindow
{
 Q_OBJECT

public:
 Note(QWidget *parent = nullptr);
 ~Note();

private slots:
  void newDocument();

  void open();

  void save();

  void save_as();

  void print();
```




**Вигляд програми**

```
    void exit();

    void copy();

    void cut();

    void paste();

    void undo();

    void redo();

    void about();

private:
 Ui::Note *ui;
 QString currentFile = "";
};
#endif // NOTE_H
```

**note.cpp**

```
#include "note.h"
#include "ui_note.h"

Note::Note(QWidget *parent)
 : QMainWindow(parent)
 , ui(new Ui::Note)
{
 ui->setupUi(this);
 this->setCentralWidget(ui->textEdit);

 connect(ui->actionNew, &QAction::triggered, this, &Note::newDocument);
 connect(ui->actionOpen, &QAction::triggered, this, &Note::open);
 connect(ui->actionSave, &QAction::triggered, this, &Note::save);
 connect(ui->actionSave_as, &QAction::triggered, this, &Note::save_as);
 connect(ui->actionPrint, &QAction::triggered, this, &Note::print);
 connect(ui->actionExit, &QAction::triggered, this, &Note::exit);

 connect(ui->actionCopy, &QAction::triggered, this, &Note::copy);
 connect(ui->actionCut, &QAction::triggered, this, &Note::cut);
 connect(ui->actionPaste, &QAction::triggered, this, &Note::paste);

 connect(ui->actionUndo, &QAction::triggered, this, &Note::undo);
 connect(ui->actionRedo, &QAction::triggered, this, &Note::redo);
```

```
 connect(ui->actionAbout, &QAction::triggered, this, &Note::about);
}

Note::~Note(){
 delete ui;
}

void Note::about(){
 QMessageBox w;
 w.setWindowTitle("About");
 w.setWindowIcon(QIcon(":/image/NoteIcons/about.png"));
 w.setText("Notepad app created by Oleksii Morozov.");
 w.exec();
}

void Note::newDocument(){
   currentFile.clear();
   ui->textEdit->setText(QString());
   setWindowTitle("*New");
   statusBar()->showMessage(tr("File created"), 1000);
}

void Note::open(){
   QString fileName = QFileDialog::getOpenFileName(this, "Open the file");
   QFile file(fileName);
   currentFile = fileName;
   if (!file.open(QIODevice::ReadOnly | QFile::Text)) {
      QMessageBox::warning(this, "Warning", "Cannot open file: " + file.errorString());
      return;
   }
   setWindowTitle(fileName);
   QTextStream in(&file);
   QString text = in.readAll();
   ui->textEdit->setText(text);
   file.close();
   statusBar()->showMessage(tr("File opened"), 1000);
}

void Note::save(){
   QString fileName;
   // If we don't have a filename from before, get one.
   if (currentFile.isEmpty()) {
      fileName = QFileDialog::getSaveFileName(this, "Save");
      currentFile = fileName;
```

```
    } else {
        fileName = currentFile;
    }
    QFile file(fileName);
    if (!file.open(QIODevice::WriteOnly | QFile::Text)) {
        QMessageBox::warning(this, "Warning", "Cannot save file: " + file.errorString());
        return;
    }
    setWindowTitle(fileName);
    QTextStream out(&file);
    QString text = ui->textEdit->toPlainText();
    out << text;
    file.close();
    statusBar()->showMessage(tr("File saved"), 1000);
}


void Note::save_as(){
    QString fileName = QFileDialog::getSaveFileName(this, "Save as");
    QFile file(fileName);

    if (!file.open(QFile::WriteOnly | QFile::Text)) {
        QMessageBox::warning(this, "Warning", "Cannot save file: " + file.errorString());
        return;
    }
    currentFile = fileName;
    setWindowTitle(fileName);
    QTextStream out(&file);
    QString text = ui->textEdit->toPlainText();
    out << text;
    file.close();
    statusBar()->showMessage(tr("File saved"), 1000);
}

void Note::print(){
    QPrinter printer;
    QPrintDialog dialog(&printer, this);
    if (dialog.exec() == QDialog::Rejected)
        return;
    ui->textEdit->print(&printer);
    statusBar()->showMessage(tr("File printed"), 1000);
}

void Note::exit(){
    QCoreApplication::quit();
```

```
}

void Note::copy(){
  ui->textEdit->copy();
  statusBar()->showMessage(tr("Text copied"), 1000);
}

void Note::cut(){
  ui->textEdit->cut();
  statusBar()->showMessage(tr("Text cutted"), 1000);
}

void Note::paste(){
  ui->textEdit->paste();
  statusBar()->showMessage(tr("text pasted"), 1000);
}

void Note::undo(){
   ui->textEdit->undo();
}

void Note::redo(){
  ui->textEdit->redo();
}
```

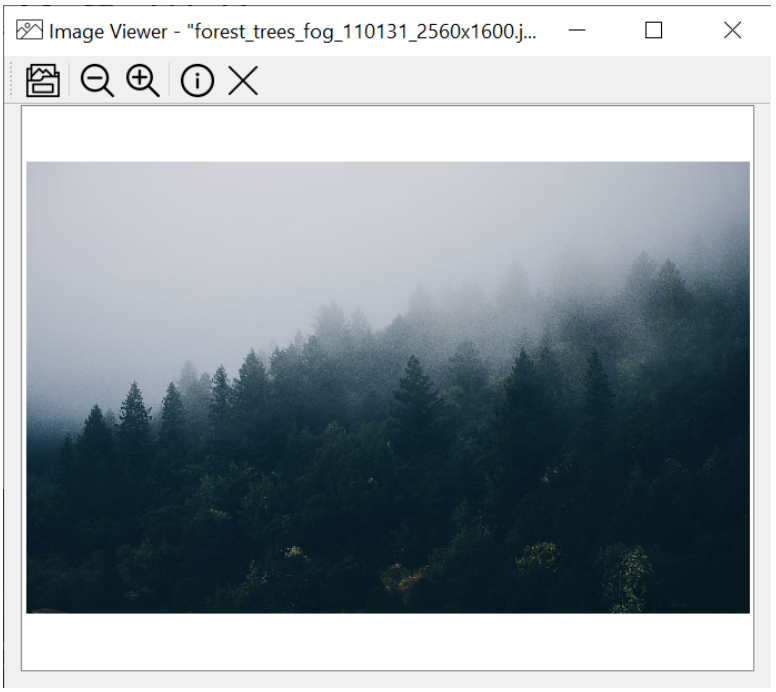
**main.cpp**

```
#include "note.h"

#include <QApplication>

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 Note window;
 window.show();

 return a.exec();
}
```

## Реалізація графічного переглядача

**imageviewer.h**

```
#pragma once

#include <scenemanager.h>

#include <QMainWindow>
#include <QMessageBox>
#include <QStringList>
#include <QFileDialog>
#include <QWheelEvent>

#include
<QtPrintSupport/QPrintDialog>

QT_BEGIN_NAMESPACE
namespace Ui { class
ImageViewer; }
QT_END_NAMESPACE

class SceneManager;

class ImageViewer : public QMainWindow
{
 Q_OBJECT

public:
 ImageViewer(SceneManager &manager, QWidget *parent = nullptr);
 ~ImageViewer();

 void openImage(const QString &path);

protected:
 void wheelEvent(QWheelEvent *event) override;
 void mouseDoubleClickEvent(QMouseEvent *event) override;

private slots:

 void ImageChanged(const QString &filename);

 void fitInWindow();
 void zoomIn();
 void zoomOut();

 void showOpenDialog();
```




**Вигляд програми**

```
 void exit();
 void about();

private:
 Ui::ImageViewer *ui;

 SceneManager &sceneManager;

 const QString defaultWindowTitle = "Image Viewer";
};
```

**imageviewer.cpp**

```
#include "imageviewer.h"
#include "ui_imageviewer.h"
#include <scenemanager.h>

ImageViewer::ImageViewer(SceneManager &manager, QWidget *parent)
 : QMainWindow(parent)
 , ui(new Ui::ImageViewer)
 , sceneManager(manager)
{
 ui->setupUi(this);
 this->setCentralWidget(ui->imageView);
 connect(ui->actionExit, &QAction::triggered, this, &ImageViewer::exit);
 connect(ui->actionAbout, &QAction::triggered, this, &ImageViewer::about);
 connect(ui->actionOpen, &QAction::triggered, this,
&ImageViewer::showOpenDialog);
 connect(ui->actionZoomIn, &QAction::triggered, this, &ImageViewer::zoomIn);
 connect(ui->actionZoomOut, &QAction::triggered, this, &ImageViewer::zoomOut);

 setWindowTitle(defaultWindowTitle);
 ui->imageView->setScene(sceneManager.scene());

 connect(&sceneManager, &SceneManager::imageChanged, this,
&ImageViewer::ImageChanged);
}

ImageViewer::~ImageViewer(){
 delete ui;
}

void ImageViewer::openImage(const QString &path)
{
 if(sceneManager.isFileSupported(path)){
```

```
    sceneManager.openImage(path);
    fitInWindow();
  }
}

void ImageViewer::wheelEvent(QWheelEvent *event)
{
 if(event->angleDelta().y() > 0){
    zoomIn();
  }
 else{
    zoomOut();
  }
}

void ImageViewer::mouseDoubleClickEvent(QMouseEvent *event)
{
 if(event->button() == Qt::LeftButton){
    fitInWindow();
  }
}


void ImageViewer::ImageChanged(const QString &filename)
{
 setWindowTitle(QString("%0 - \"%1\"").arg(defaultWindowTitle).arg(filename));
}

void ImageViewer::fitInWindow()
{
 sceneManager.fitInWindow();
}

void ImageViewer::zoomIn()
{
 ui->imageView->scale(1.1, 1.1);
}

void ImageViewer::zoomOut()
{
  ui->imageView->scale(0.9, 0.9);
}

void ImageViewer::showOpenDialog()
{
```

```
 QString imageExtensions =
QString("").arg(sceneManager.supportedExtensions().join(" "));

 QString path = QFileDialog::getOpenFileName(this, "Open image", "",
imageExtensions);

 if(path.size()){
    openImage(path);
  }
}

void ImageViewer::exit(){
 QCoreApplication::quit();
}

void ImageViewer::about(){
 QMessageBox w;
 w.setWindowTitle("About");
 w.setWindowIcon(QIcon(":/icons/Icons/information.png"));
 w.setText("Image viewer app created by Oleksii Morozov.");
 w.exec();
}
```

**scenemanager.h**

```
#pragma once

#include <QObject>
#include <imageviewer.h>

class QGraphicsScene;

class SceneManager : public QObject
{
 Q_OBJECT
public:
 explicit SceneManager(QObject *parent = nullptr);

 QGraphicsScene* scene();
 QStringList supportedExtensions();
 void openImage(const QString &path);

 void fitInWindow();
 bool isFileSupported(const QString &path);
```

```
 void setExtensions(const QStringList &extensions);
 void openFile(const QString &path);
 QString getCurrentFilename();

signals:
 void imageChanged(const QString &filename);

private:
 void showImage(const QString &path);

 QStringList fileNamesInCurrentPath;

 QString currentFilePath;
 QString currentFileName;

 QGraphicsScene *imageScene = nullptr;
 const QStringList imgExtensions = {".png", ".bmp", ".jpg", ".jpeg"};
};
```

**scenemanager.cpp**

```
#include "scenemanager.h"

#include <QGraphicsScene>
#include <QGraphicsView>
#include <QGraphicsPixmapItem>
#include <QImageReader>

SceneManager::SceneManager(QObject *parent): QObject{parent}{
 imageScene = new QGraphicsScene(this);
 supportedExtensions();
}

QGraphicsScene *SceneManager::scene(){
 return imageScene;
}

QStringList SceneManager::supportedExtensions()
{
 return imgExtensions;
}

void SceneManager::openImage(const QString &path){
 openFile(path);
```

```
 showImage(path);
}

void SceneManager::fitInWindow(){
 if(imageScene->items().size() && imageScene->views().size()){
    auto view = imageScene->views().at(0);
    view->fitInView(imageScene->items().at(0), Qt::KeepAspectRatio);
    view->setSceneRect(imageScene->items().at(0)->boundingRect());
  }
}

bool SceneManager::isFileSupported(const QString &path){
 for(auto ext : imgExtensions){
    ext = ext.remove(0,1);
    if(path.endsWith(ext, Qt::CaseInsensitive)){
       return 1;
     }
  }
 return 0;
}

void SceneManager::showImage(const QString &path){
 imageScene->clear();

 QImageReader imgReader(path);
 imgReader.setAutoTransform(1);

 QImage img = imgReader.read();

 if(!img.isNull()){
    QPixmap pixmap = QPixmap::fromImage(img);

    imageScene->addPixmap(pixmap);
    imageScene->update();

    emit imageChanged(getCurrentFilename());
  }
}

void SceneManager::openFile(const QString &path){
 QFileInfo current(path);
 currentFilePath = path;
 currentFileName = current.fileName();
}
```

```
QString SceneManager::getCurrentFilename(){
 return currentFileName;
}
```

**main.cpp**

```
#include "imageviewer.h"
#include <scenemanager.h>

#include <QApplication>

int main(int argc, char *argv[]){
 QApplication a(argc, argv);

 SceneManager sceneManager;
 ImageViewer w{sceneManager};
 w.show();
 return a.exec();
}
```

## ВИСНОВКИ

Виконавши лабораторну роботу №3, я ознайомився із невізуальними компонентами середовища розробки **Qt Creator**, створив та налаштував віконний проект з використовуючи ці компоненти.

# ЛАБОРАТОРНА РОБОТА №4

**Тема:** Компоненти Qt Creator для представлення даних.
**Мета:** Створити віконний проект та продемонструвати використання компонент призначених для відображення та опрацювання даних.

## ТЕОРЕТИЧНІ ВІДОМОСТІ

**Qt Creator**, як і будь-яке інше середовище розробки надає змогу використовувати таблиці для організації даних. Для їх створення

використовується компонента **TableWidget**. Вона є членом класу **QTableWidget**. Редагування таблиці може відбуватись за допомогою команд, написаних у редакторі коду. За допомогою методів **setRowCount(int value)** та **setColumnCount(int value)** задаються розмірність таблиці, а саме кількість рядків та стовпців. Методи **setHorizontalHeaderLabels()** та **setVerticalHeaderLabels()** дають назву кожному із стовпців та рядків. Скориставшись методами **setColumnWidth()** та **setRowHeight()** можна вказати розміри кожної з клітинок таблиці.

# ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Ознайомитись із компонентою **TableWidget**.
2. Реалізувати гру.

## Реалізація гри “Судоку”

**game.h**

```
#ifndef GAME_H
#define GAME_H

#include <QMainWindow>
#include <QTableWidget>
#include <QWidget>
#include <QMessageBox>
#include <matrix.h>
#include "ui_game.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Game; }
QT_END_NAMESPACE

class Game : public QMainWindow
{
 Q_OBJECT

public:
 Game(QWidget *parent = nullptr);
 ~Game();

private slots:
```

```
 void newGame();
 void howToPlay();
 void checkTable();
 void exit();

 void on_pushButton_1_clicked();
 void on_pushButton_2_clicked();
 void on_pushButton_3_clicked();
 void on_pushButton_4_clicked();
 void on_pushButton_5_clicked();
 void on_pushButton_6_clicked();
 void on_pushButton_7_clicked();
 void on_pushButton_8_clicked();
 void on_pushButton_9_clicked();
 void on_tableWidget_cellClicked(int row, int column);

private:
 Ui::Game *ui;
 int Row, Column;
 void clickOnButton(int num);
};
#endif // GAME_H
```

**game.cpp**

```
#include "game.h"

#include <QTableWidget>
#include <QWidget>
#include <QMessageBox>

sudokuGame::Matrix matrix;

Game::Game(QWidget *parent): QMainWindow(parent) , ui(new
Ui::Game)
{
 ui->setupUi(this);
 setWindowTitle("Sudoku");
 Game::newGame();
 connect(ui->actionExit, &QAction::triggered, this, &Game::exit);
 connect(ui->actionNew, &QAction::triggered, this,
&Game::newGame);
```

```
 connect(ui->actionCheck, &QAction::triggered,this,
&Game::checkTable);
 connect(ui->actionQuestion, &QAction::triggered,this,
&Game::howToPlay);
}

void Game::newGame(){
matrix.start();
for (int i = 0; i < 9; ++i)
{
   for (int j = 0; j < 9; ++j)
   {
      QTableWidgetItem* Cell = ui->tableWidget->item(i, j);
      QString str = "";
      str += '0' + QString::number(matrix.read(i, j));
      const QString cstr = str;
      Cell->setText(cstr);
      if (matrix.read(i, j))
      {
         QColor color(188,188,188);
         Cell->setBackground(color);
      }
      else
      {
         QColor color(255,255,255);
         Cell->setBackground(color);
      }
   }
}
}

void Game::clickOnButton(int num)
{
  QTableWidgetItem* Cell = ui->tableWidget->item(Row, Column);
  matrix.write(Row, Column, num);
  QString str = "";
  str += '0' + QString::number(matrix.read(Row, Column));
  const QString cstr = QString::number(num);
  Cell->setText(cstr);
}
```

```
void Game::howToPlay(){
 QMessageBox faq;
 faq.setWindowTitle("How to play sudoku");
 faq.setWindowIcon(QIcon(":/icons/Icons/sudoku.png"));
 faq.setText("Click on a cell, which you want to change.\nThen click on
a button with number you want to enter on cell.");
 faq.exec();
}

void Game::checkTable(){
 if (matrix.checkWin())
 {
    QMessageBox youWin;
    youWin.setWindowIcon(QIcon(":/icons/Icons/check.png"));
    youWin.setText("You win!");
    youWin.exec();
 }
 else
 {
    QMessageBox youWin;
    youWin.setWindowIcon(QIcon(":/icons/Icons/forbidden.png"));
    youWin.setText("Sudoku don't solved");
    youWin.exec();
 }
}

void Game::exit(){
 QCoreApplication::quit();
}

Game::~Game()
{
 delete ui;
}


void Game::on_tableWidget_cellClicked(int row, int column)
{
 Row = row;
 Column = column;
}
```

```
void Game::on_pushButton_1_clicked()
{
  clickOnButton(1);
}


void Game::on_pushButton_2_clicked()
{
 clickOnButton(2);
}


void Game::on_pushButton_3_clicked()
{
  clickOnButton(3);
}


void Game::on_pushButton_4_clicked()
{
  clickOnButton(4);
}


void Game::on_pushButton_5_clicked()
{
  clickOnButton(5);
}


void Game::on_pushButton_6_clicked()
{
  clickOnButton(6);
}


void Game::on_pushButton_7_clicked()
{
  clickOnButton(7);
```

```
}


void Game::on_pushButton_8_clicked()
{
  clickOnButton(8);
}


void Game::on_pushButton_9_clicked()
{
  clickOnButton(9);
}
```

**matrix.h**

```
#ifndef MATRIX_H
#define MATRIX_H

#include <randomise.h>
namespace sudokuGame{
class Matrix
{
public:
 bool checkWin();

 void write(int i, int j, int val);

 void start();

 int read(int i, int j);

 int input(int (*mat)[10]);

 Matrix();

private:
 int obj[10][10];
 int row[10];
 int col[10];
 int block[5][5];
```

```
 int memoryMatrix[10][10];
 sudokuGame::Randomise randomise;
 int dfs(int ni, int nj);
 void empty();
 void init();
};
}
#endif // MATRIX_H
```

**matrix.cpp**

```
#include "matrix.h"

void sudokuGame::Matrix::init()
{
 for (int i = 0; i < 9; ++i)
 {
  this->row[i] = this->col[i] = 0;
  this->block[i/3][i%3] = 0;
 }
 for (int i = 0; i < 9; ++i)
 {
  for (int j = 0; j < 9; ++j)
  {
   if(this->obj[i][j])
   {
    int sig = (1 << (obj[i][j] - 1));
    this->row[i] |= sig;
    this->col[j] |= sig;
    this->block[i/3][j/3] |= sig;
    this->memoryMatrix[i][j] = 1;
   }
   else
   {
    this->memoryMatrix[i][j] = 0;
   }
  }
 }
}

int sudokuGame::Matrix::input(int (*mat)[10])
{
```

```
 for (int i = 0; i < 9; ++i)
 {
 for (int j = 0; j < 9; ++j)
  {
    this->obj[i][j] = mat[i][j];
  }
 }
 return 1;
}

sudokuGame::Matrix::Matrix()
{
  for (int i = 0; i < 9; ++i)
  {
     this->row[i] = this->col[i] = 0;
     this->block[i/3][i%3] = 0;
  }

  for (int i = 0; i < 9; ++i)
  {
     for (int j = 0; j < 9; ++j)
     {
        this->obj[i][j] = 0;
     }
  }
}

void sudokuGame::Matrix::write(int i, int j, int val)
{
 if (memoryMatrix[i][j]){
  return;
  }
 this->obj[i][j] = val;
}

int sudokuGame::Matrix::read(int i, int j)
{
     return this->obj[i][j];
}

void sudokuGame::Matrix::empty()
```

```
{
  for (int i = 0; i < 9; ++i)
  {
     this->row[i] = this->col[i] = 0;
     this->block[i/3][i%3] = 0;
  }

  for (int i = 0; i < 9; ++i)
  {
     for (int j = 0; j < 9; ++j)
     {
        this->obj[i][j] = 0;
     }
  }
}

bool sudokuGame::Matrix::checkWin()
{
 for (int i = 0; i < 9; ++i)
 {
  row[i] = 0;
  col[i] = 0;
  block[i/3][i%3] = 0;
 }
 for (int i = 0; i < 9; ++i)
 {
  for (int j = 0; j < 9; ++j)
    {
     int sig = row[i] | col[j] | block[i/3][j/3];
     int sgn = (1 << (obj[i][j] - 1));
     if (!(sig&sgn))
     {
           row[i] |= sgn;
           col[j] |= sgn;
           block[i/3][j/3] |= sgn;
     }
     else
     {
           return false;
     }
    }
```

```
 }
 return true;
}

void sudokuGame::Matrix::start()
{
 empty();
 dfs(0, 0);
 srand(time(NULL));
 int rdbox = rand()%9 + 40;
 for (int k = 0; k < rdbox; ++k)
 {
   int i = rand()%9;
   int j = rand()%9;
   while(!obj[i][j])
   {
     i = rand()%9;
     j = rand()%9;
   }
   obj[i][j] = 0;
 }
 init();
}

int sudokuGame::Matrix::dfs(int ni, int nj)
{
        if (this->obj[ni][nj])
        {
                if (nj+1 < 9)
                        return dfs(ni, nj+1);
                else if (ni+1 < 9)
                        return dfs(ni+1, 0);
                else
                        return 1;
        }
        else
        {
                int sig = this->row[ni] | this->col[nj] | this->block[ni/3][nj/3];
                this->randomise.randarray();
                for (int i = 0; i < 9; ++i)
                {
```

```
                int ri = this->randomise.read(i);
                int sign = (1<<(ri-1));
                if (!(sig & sign))
                {
                        this->obj[ni][nj] = ri;
                        this->row[ni] |= sign;
                        this->col[nj] |= sign;
                        this->block[ni/3][nj/3] |= sign;
                        if (nj+1 < 9)
                        {
                                if (dfs(ni, nj+1) > 0)
                                        return 1;
                        }
                        else if (ni+1 < 9)
                        {
                                if (dfs(ni+1, 0) > 0)
                                        return 1;
                        }
                        else
                        {
                                return 1;
                        }
                        this->obj[ni][nj] = 0;
                        this->row[ni] ^= sign;
                        this->col[nj] ^= sign;
                        this->block[ni/3][nj/3] ^= sign;
                }
        }
        return -1;
    }
}
```

**randomise.h**

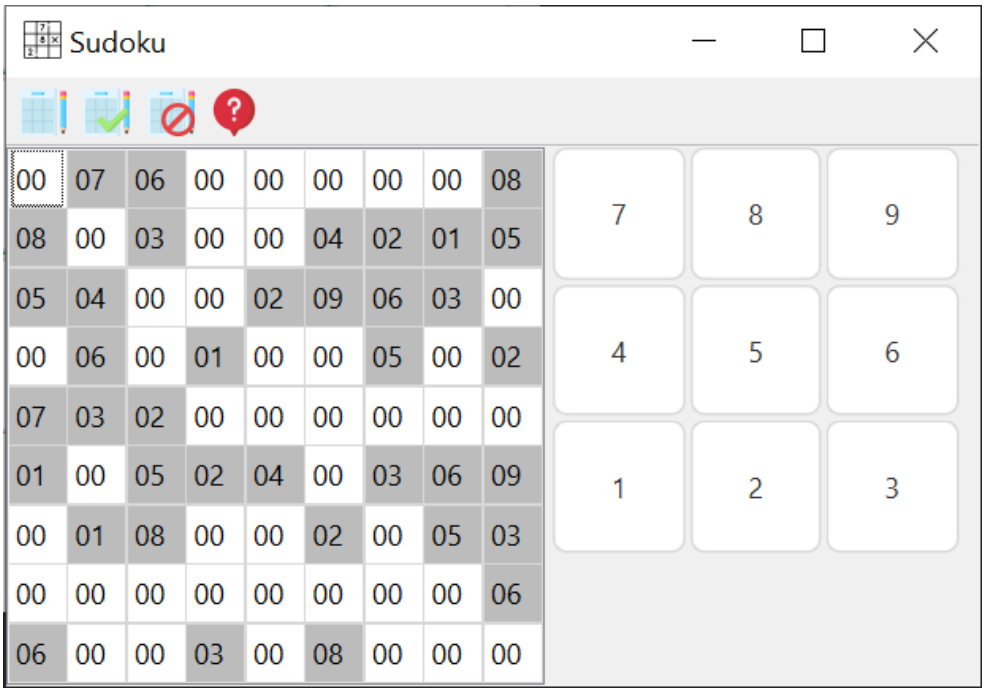
```
#ifndef RANDOMISE_H
#define RANDOMISE_H

#include <cstdlib>
#include <time.h>
#include <cstdio>
#include <algorithm>
```

```
namespace sudokuGame{
class Randomise
{
public:
 Randomise();
 void randarray();
 int read(int i);

private:
 int array[10];
 int rdseed;
};
}
#endif // RANDOMISE_H
```

**randomise.cpp**

```
#include "randomise.h"

sudokuGame::Randomise::Randomise()
{
 this->rdseed = time(NULL);
 randarray();
}

int sudokuGame::Randomise::read(int i)
{
 return this->array[i];
}

void sudokuGame::Randomise::randarray()
{
        srand(rdseed);
        this->rdseed = rand();
        int queue[10];
        for (int i = 0; i < 9; ++i)
        {
                queue[i] = i+1;
        }
        for (int i = 0; i < 9; ++i)
        {
                int stp = rand()%9;
```

```
            while(!queue[stp])
            {
                    stp ++;
                    stp %= 9;
            }
            this->array[i] = queue[stp];
            queue[stp] = 0;
    }
}
```




**main.cpp**

```
#include "game.h"
#include <QApplication>

int main(int argc, char *argv[])
{
 QApplication a(argc, argv);
 Game window;
 window.show();
 return a.exec();
}
```

Ui-вигляд програми

# ВИСНОВКИ

Виконавши лабораторну роботу №4, я ознайомився із організацією таблиць за допомогою компоненти **TableWidget** середовища розробки **Qt Creator**, створив та налаштував віконний проект з цією візуальною компонентою.