Міністерство освіти і науки, молоді та спорту України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 1-4**

«Ознайомлення із середовищем»

**з дисципліни «Об’єктно-орієнтоване програмування»**

**Лектор:**

доцент кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-15

Марущак А.С.

**Прийняв:**

доцент кафедри ПЗ

Яцишин С.І.

«\_\_\_» \_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема роботи:** Ознайомлення із середовищем розробки. Створення проекту та налаштування його властивостей.

**Мета роботи:** Засвоїти принцип візуального програмування шляхом створення та налаштування проекту.

**Теоретичні відомості**

Інтерфейс Borland C++ Builder, QT Creator, Visual Studio 2019 та под. називають середовищем швидкої розробки застосувань RAD (Rapid Application Development) або середовищем візуальної розробки. Таку назву ці інтерфейси отримали за те, що створення застосування в них зводиться в основному до простого конструювання вікна майбутнього застосування із набору готових компонент, а більшу частину стандартних операцій виконує комп’ютер.

Починаючи з версії 4.5, до комплекту Qt включене середовище розробки Qt Creator, яке містить у собі редактор коду, довідку, графічні засоби Qt Designer і можливість дебагінгу застосунків. Qt Creator може використовувати GCC або Microsoft VC++ як компілятори.

Qt Creator — середовище розробки, призначене для створення крос-платформових застосунків із використанням бібліотеки Qt. Підтримується розробка як класичних програм мовою C++, так і використання мови QML для визначення сценаріїв, у якій використовується JavaScript, а структура й параметри елементів інтерфейсу задаються CSS-подібними блоками.

Qt комплектується візуальним середовищем розробки графічного інтерфейсу Qt Designer, що дозволяє створювати діалоги й форми мишею У комплекті постачання Qt є Qt Linguist — графічна утиліта, що дозволяє спростити локалізацію й переклад вашої програми багатьма мовами, та Qt Assistant — довідкова система Qt, що спрощує роботу з документацією для бібліотеки та дозволяє створювати крос-платформову довідку для ПЗ, що розробляється на основі Qt.

Qt містить безліч готових компонент та засобів для створення нових. Серед вбудованих компонент, наприклад:

* QLabel
* QPushButton
* QLineEdit
* QTextEdit
* Різні види розміток: QVBoxLayout, QHBoxLayout, QGridLayout, QformLayout
* QFrame.
* Різноманітні діалоги: QFontDialog, QPrintDialog, QColorDialog тощо.

**Індивідуальне завдання**

**Лабараторна робота №1**

1. Ознайомитись із середовищем.
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.
4. Запустити на виконання застосування.
5. Відкрити опції проекту, змінити налаштування на закладках **Application, Compiler, Packages.** Запустити на виконання застосування.

**Лабараторна робота №2**

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

**Лабараторна робота №3**

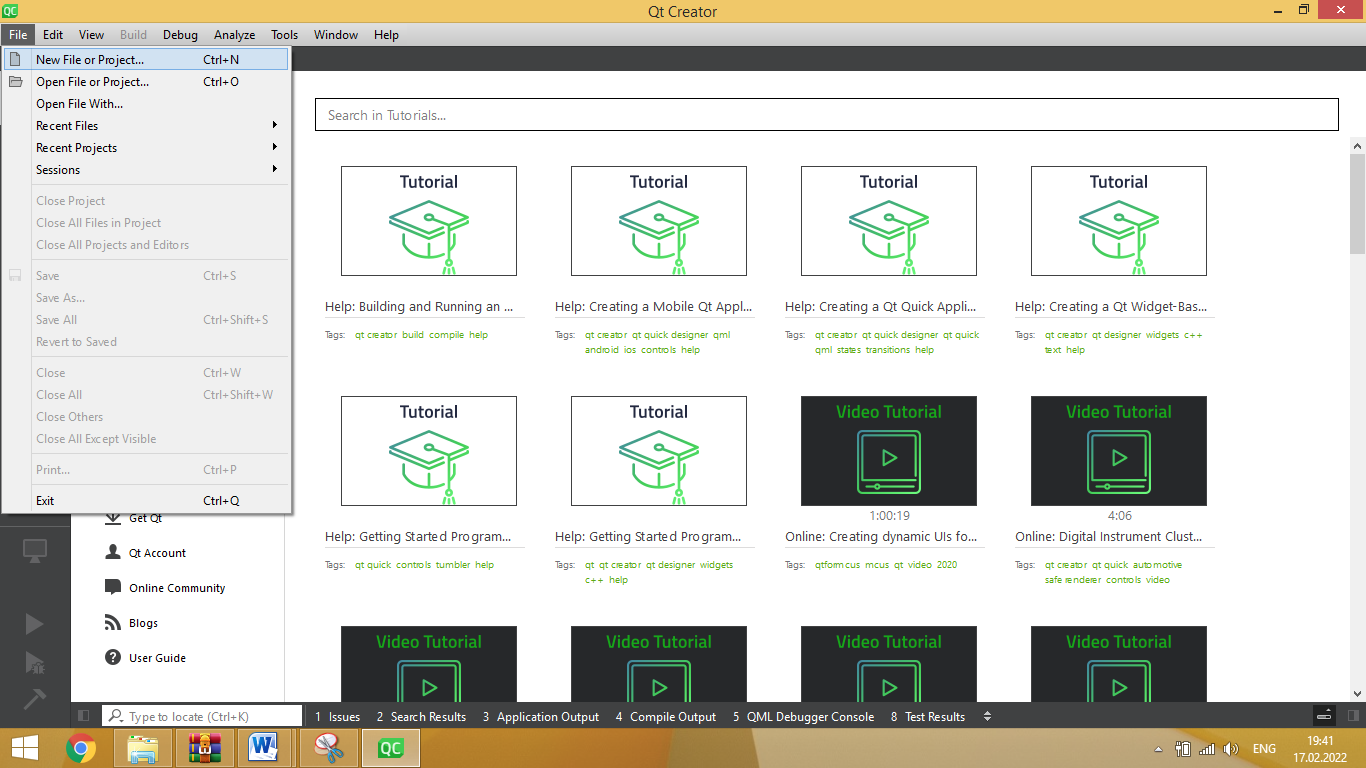
1. Створити віконний проект. Додати головне та контекстне меню, необхідні системні діалоги.
2. Реалізувати текстовий редактор і переглядач графічних файлів.

**Лабараторна робота №4**

1. Ознайомитись із компонентою StringGrid.
2. Реалізувати гру.

**Хід роботи**

При виконанні цієї лабараторної роботи я буду використовувти середовище розробки QT Creator. Почнемо зі створення проекту:



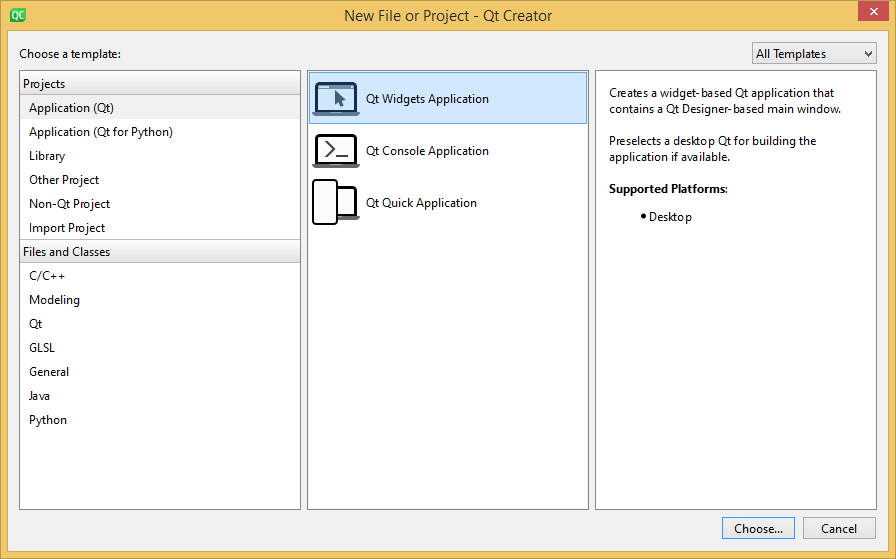


Рис 1.1-2. Процес створення віконного додатку в QT Creator.

Далі нам необхідно вказати шлях, а також назву проекту:

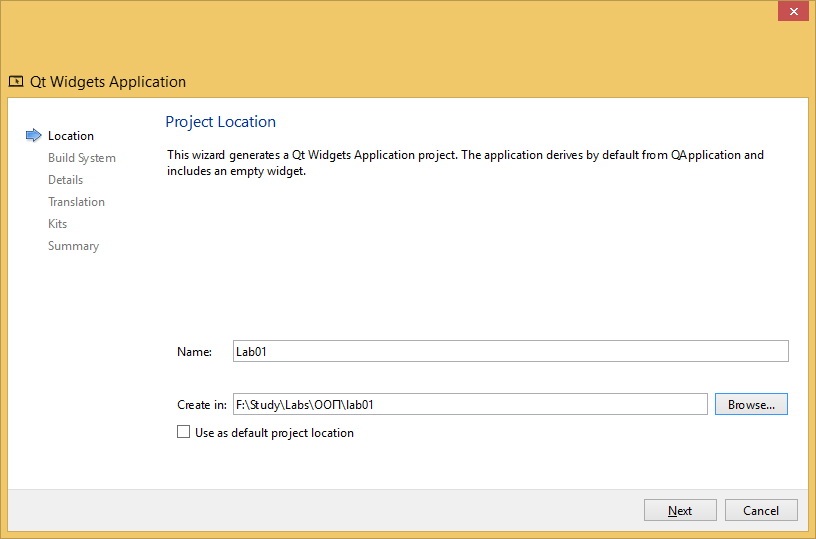


Рис 1.3 Вікно вводу назви та розміщення проекту.

Далі нас зустрічає вікно налаштування головної форми, де я виставляю наступні налаштування:

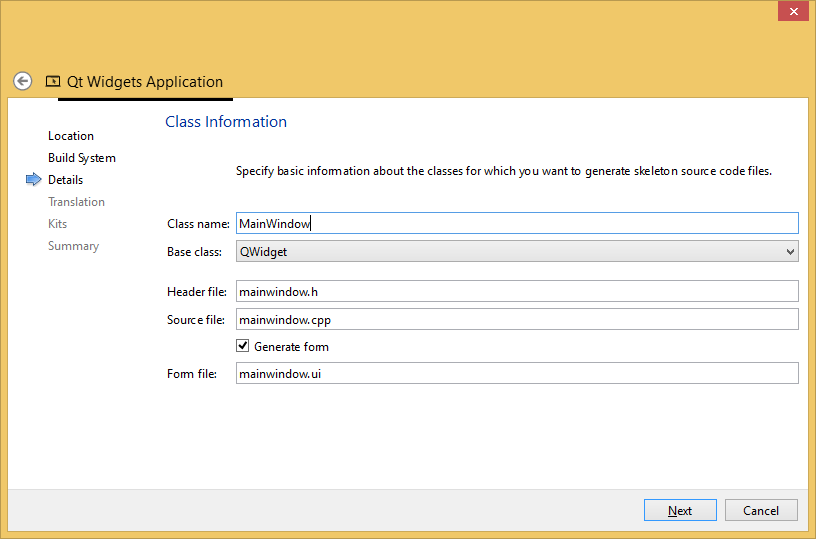


Рис 1.4 Налаштування головної форми.

Всі інші налаштування залишаємо за замовчуванням і тепер середовище створило нам новий проект. Збергіаю його за допомогою спеціального пункту меню:

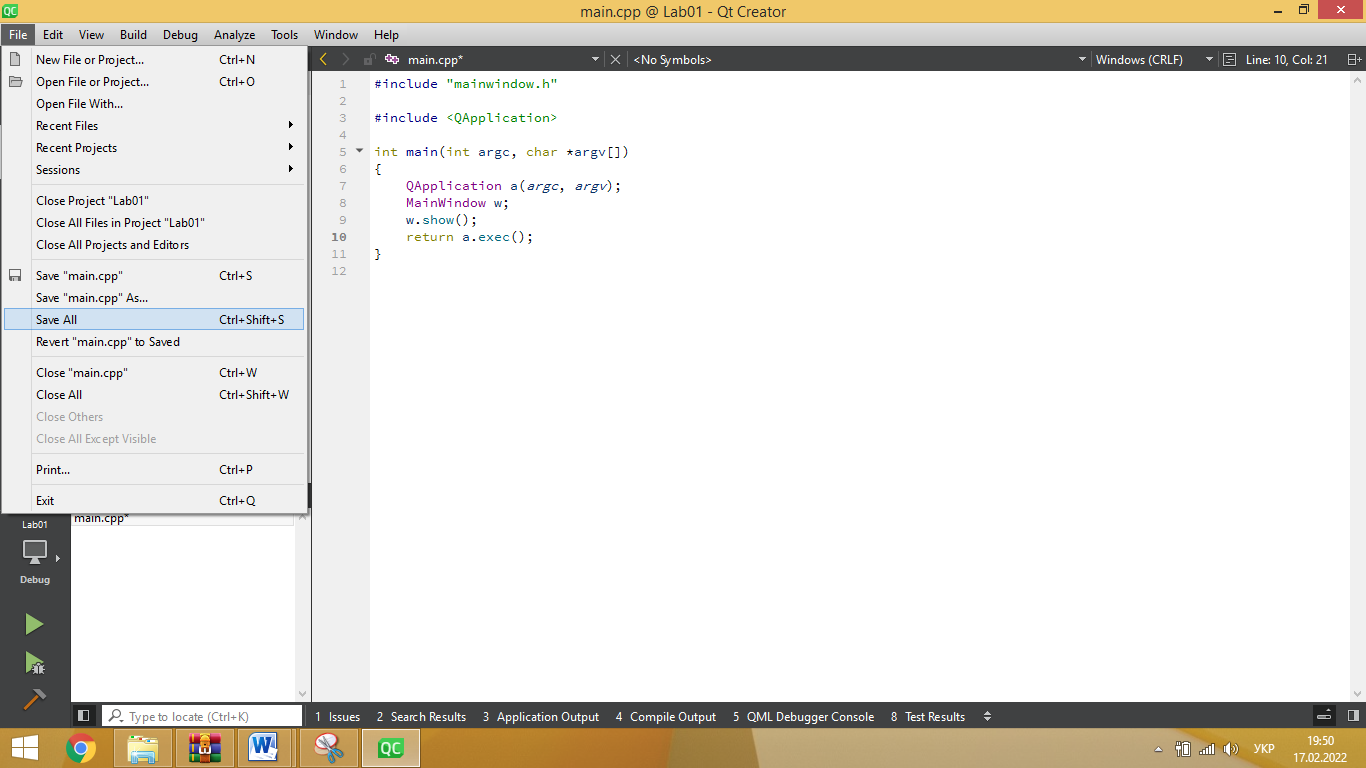


Рис 1.5 Збереження проекту.

Також, як видно з рис 1.5, збереження проекту можна виконати за допомогою комбінації клавіш Ctrl+Shift+S.

Для роботи з формою в файлах оберемо файл mainwindow.ui. Редактор форм виглядає наступним чином:

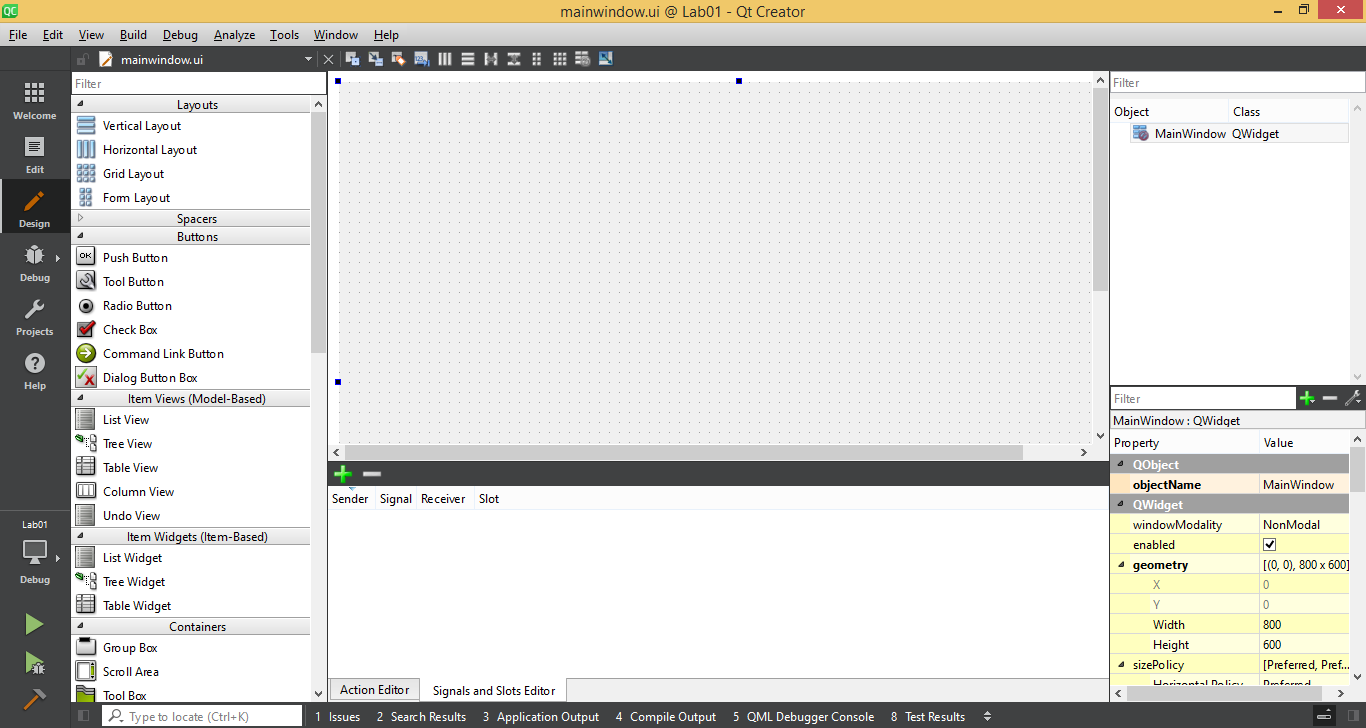


Рис 1.6. Редактор форм.

Для редагування властивостей вікна, я обираю його в списку об’єктів(на рис. 1.6 він знаходиться в правій верхній частині екрану). Потім нам необхідно скористатись вікном редагування властивостей(на рис 1.6 воно знаходиться в правому нижньому кутку). Там нам необхідно знайти необхідні властивості і змінити її значення, як показано на рисунку:

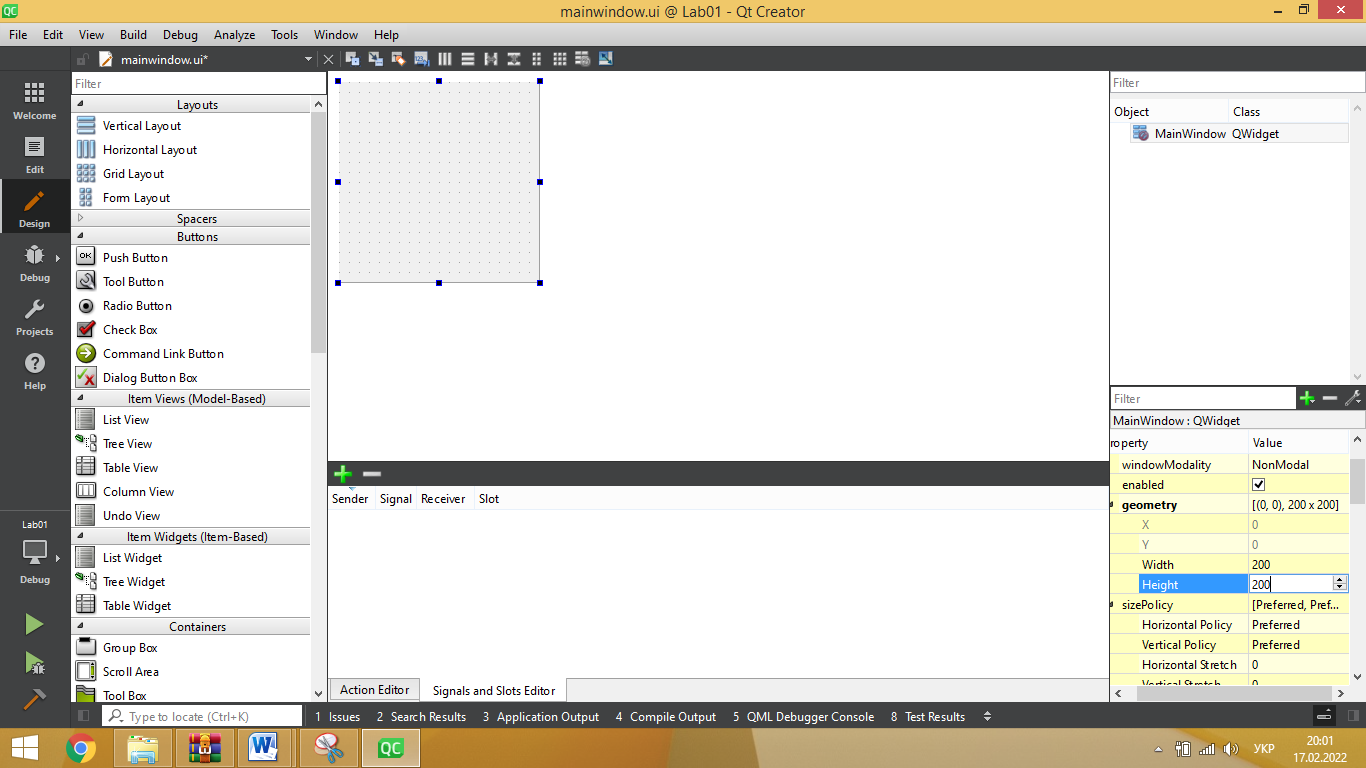
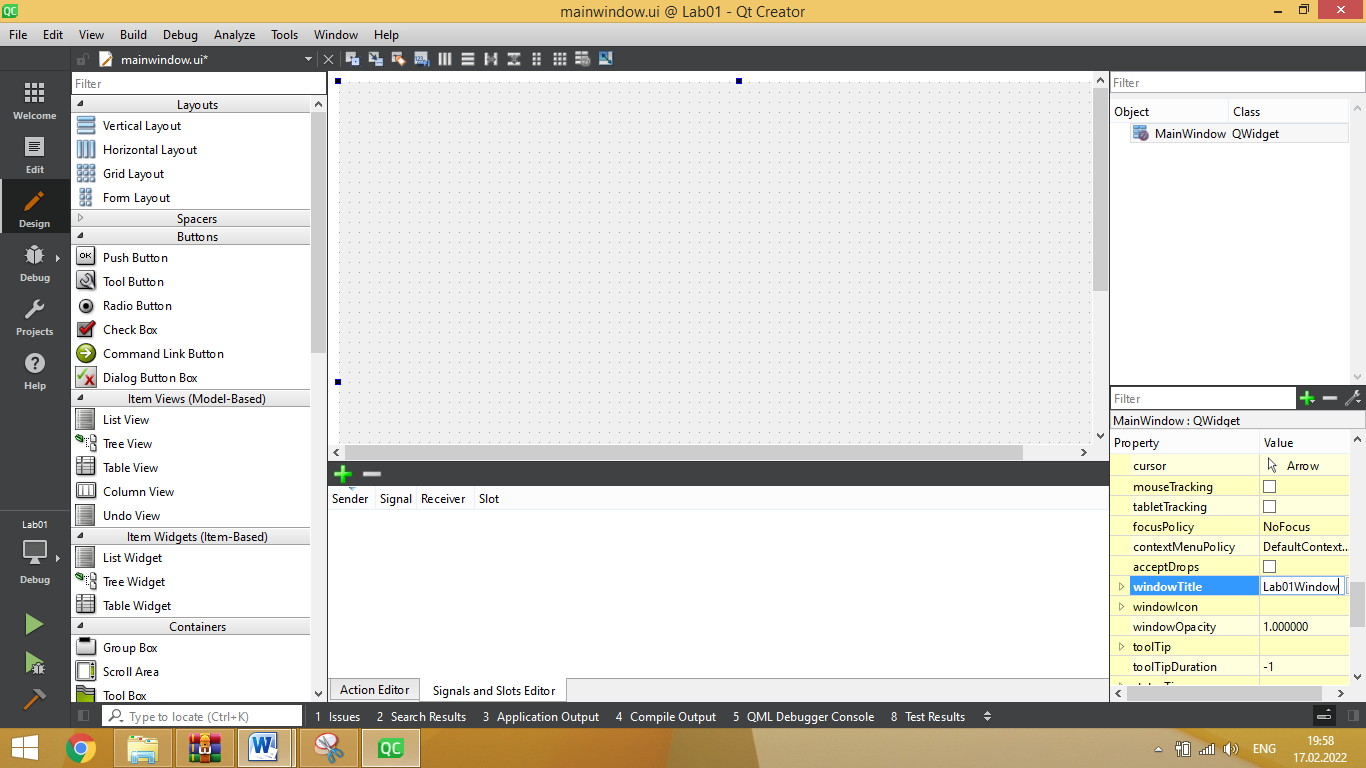


Рис. 1.7-8. Редагування назви (рис. зліва) та розмірів(рис. справа).

Запустимо проект на виконання. Для цього натиснемо на зелену стрілочку(на рис 1.5 вона знаходиться в лівому нижньому кутку) або натиснемо комбінацію клавіш Ctrl+R:

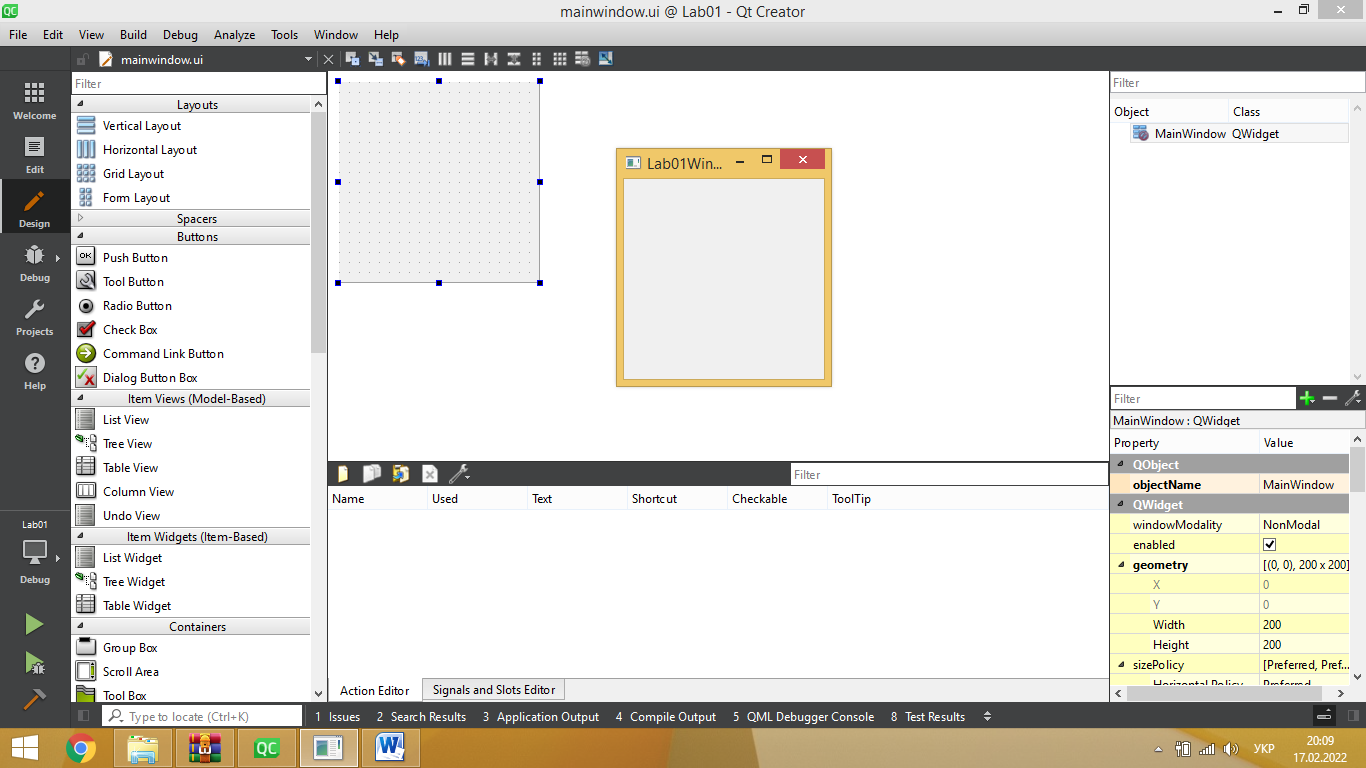


Рис 1.9. Результат виконання програми.

Також не рідко постає необхідність змінити деякі параметри додатку, для цього ми можемо використати вкладку Projects (на рис 1.9 вона знаходиться зліва). У цій вкладці ми можемо налаштувати шлях збірки проекту, деякі опції дебагінгу, а також змінити систему збірки.

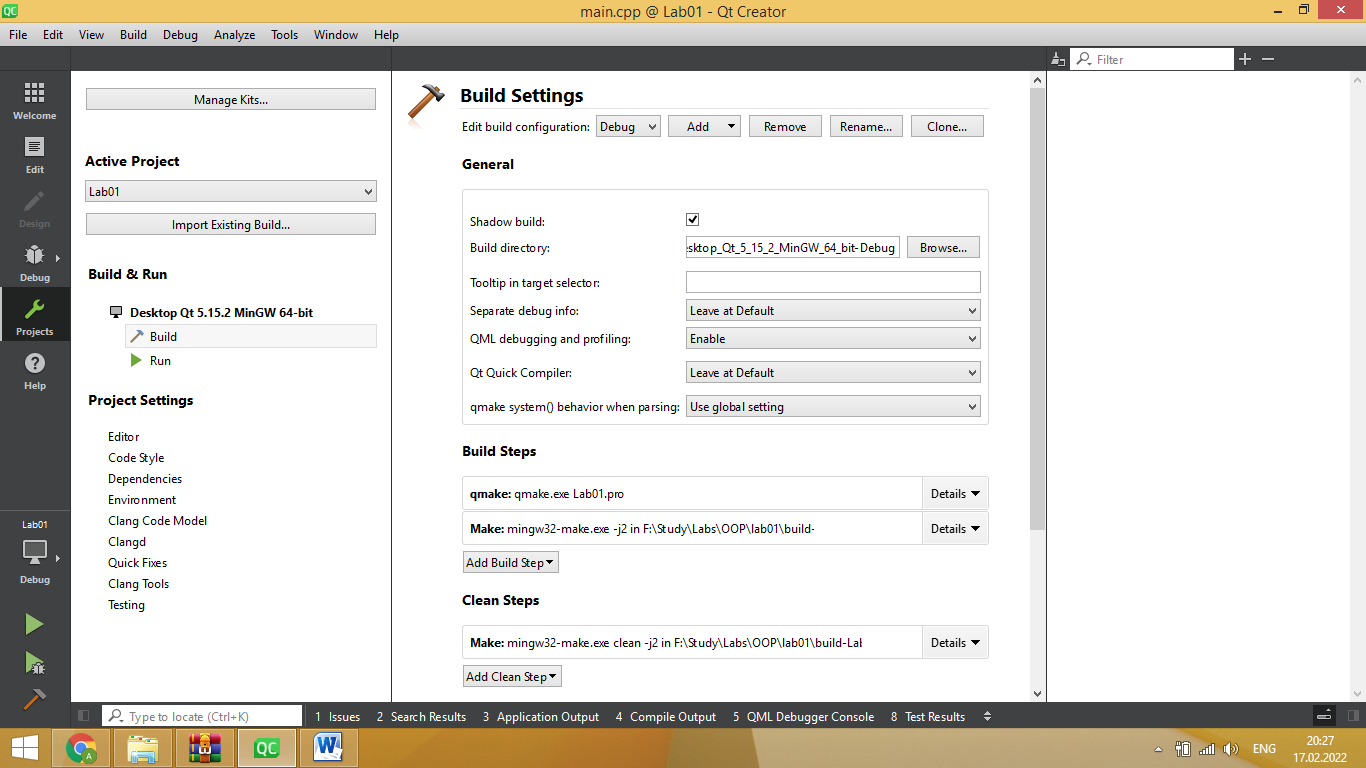


Рис 1.10 Вікно налаштування параметрів компіляції та збірки.

Також можливо змінювати режим виконання програми (так звані Debug та Release), використовуючи меню, як показано на наступному рисунку:

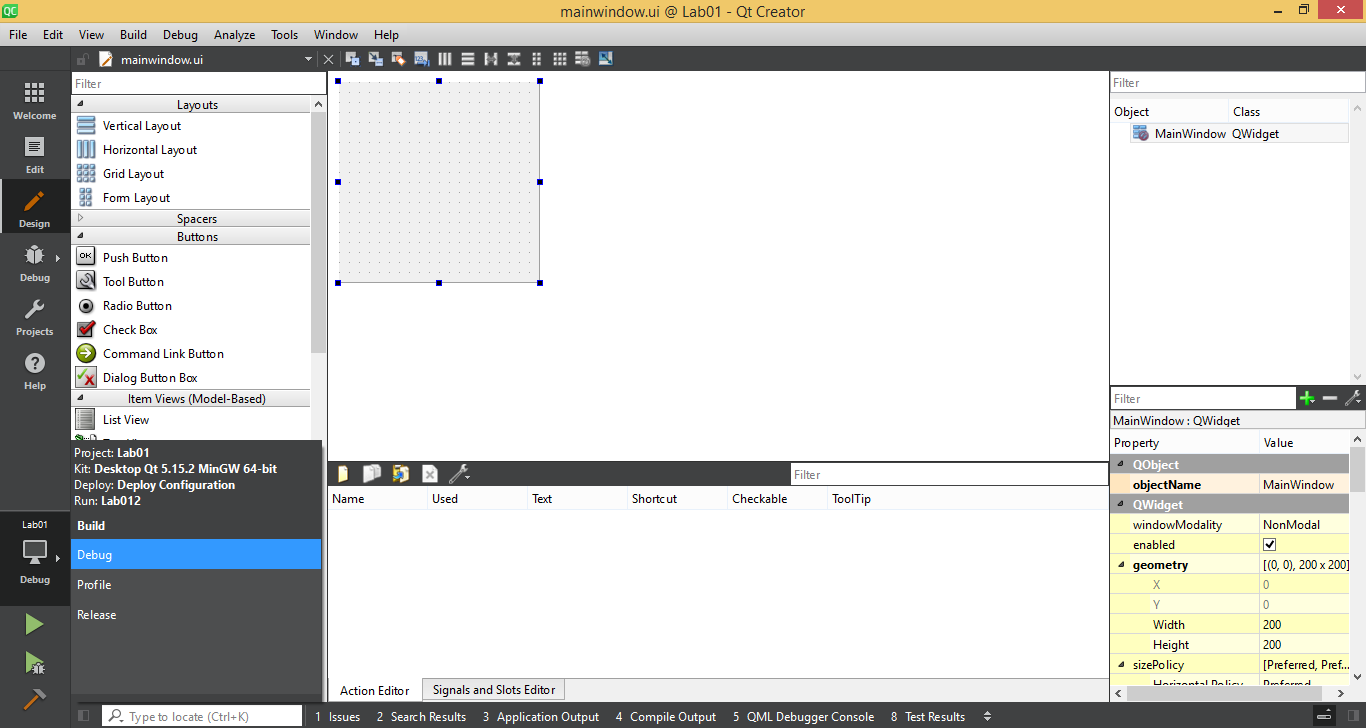


Рис 1.11 Змінення режиму виконання програми:

Змінимо режим виконання на Release та запустимо програму:

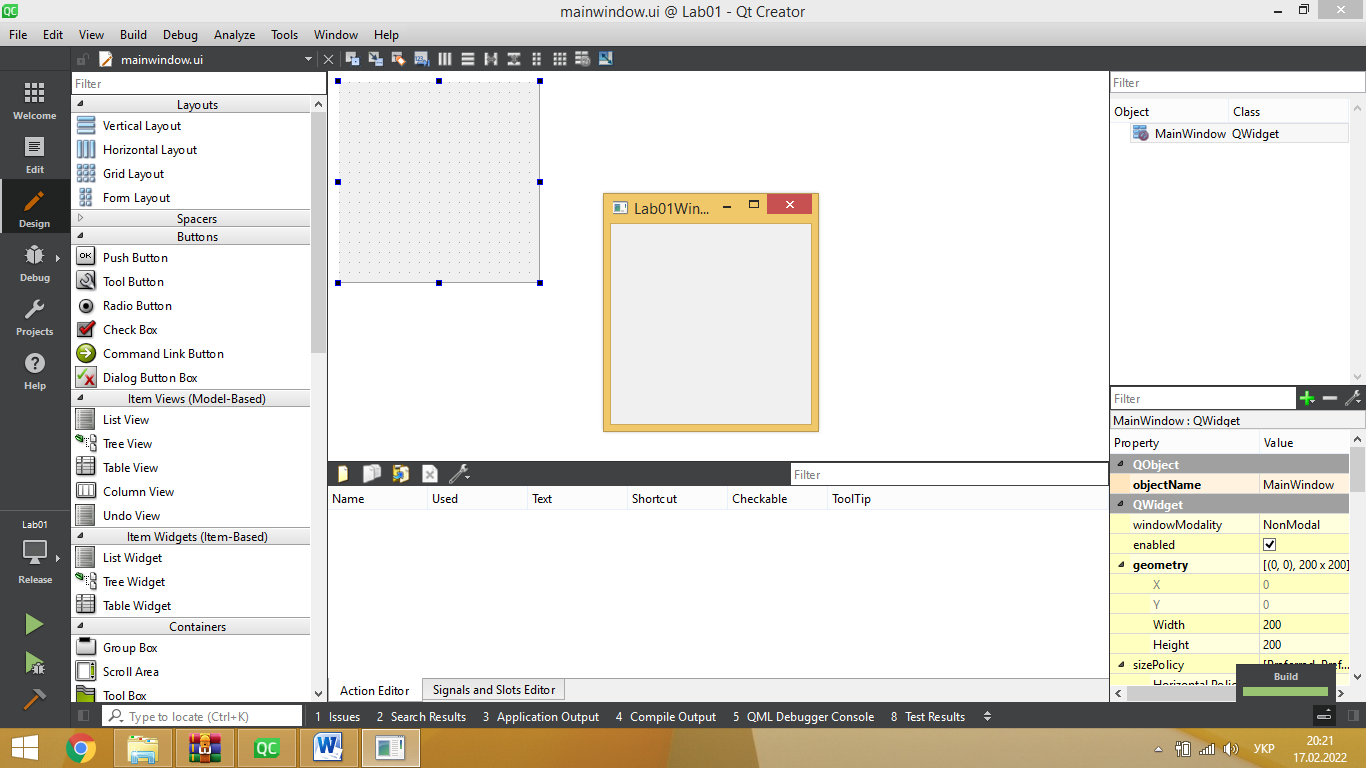


Рис 1.12. Запуск програми зі зміненим режимом.

Якщо ми подивимось у директорії, де знаходяться виконавчі файли, то ми побачимо, що їх розміри дуже сильно відрізняються:

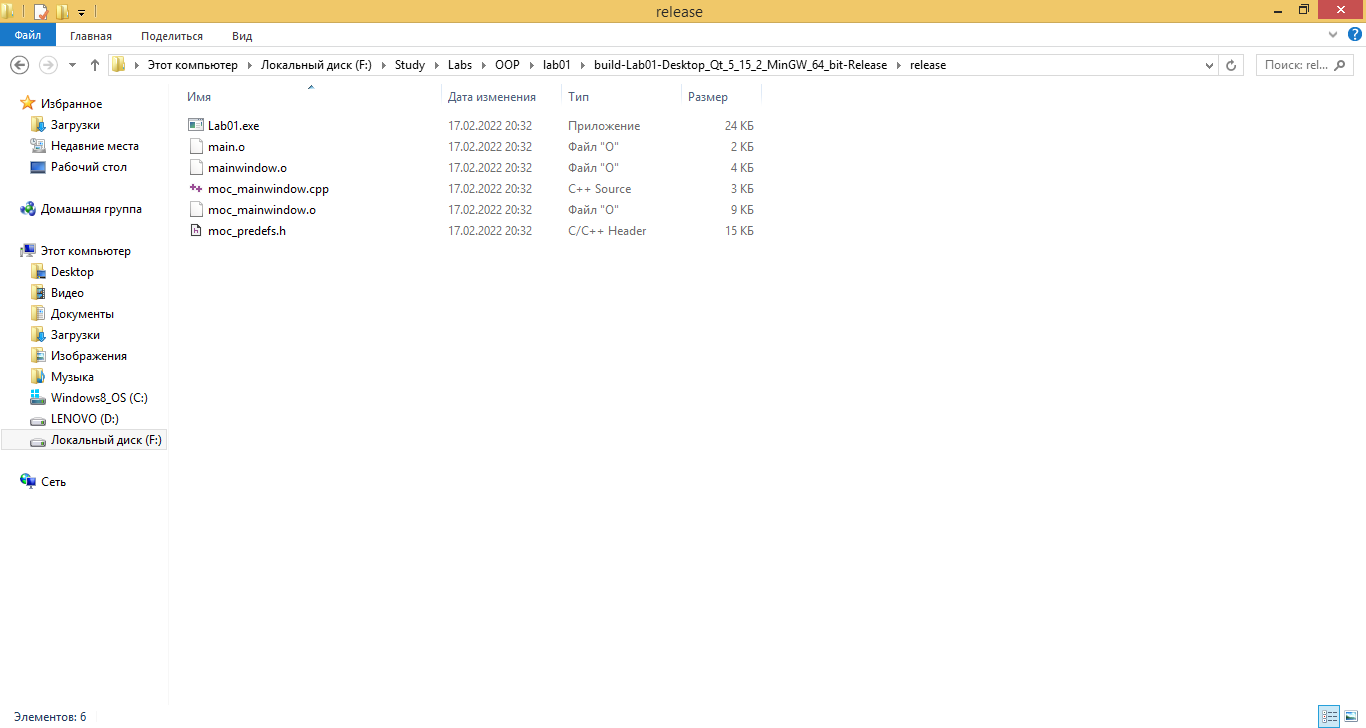
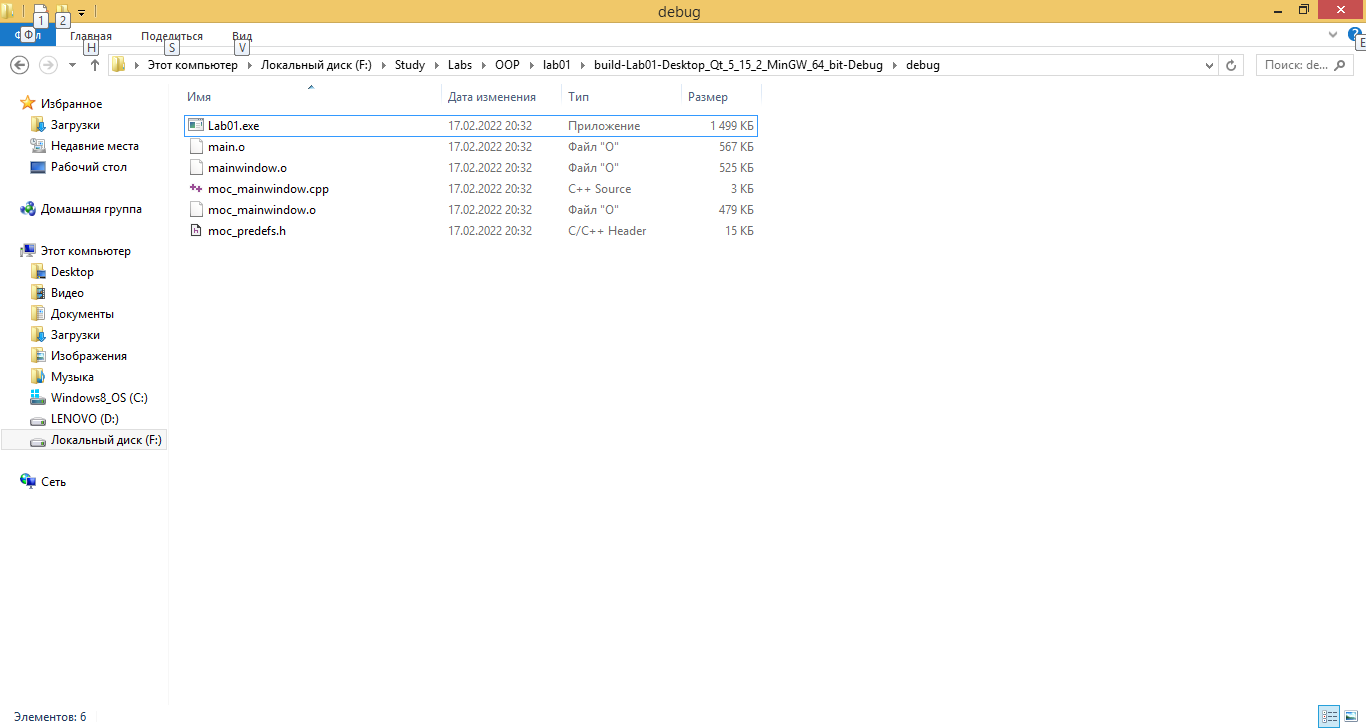


Рис 1.13. Різниця в розмірах виконавчих файлів(вгорі Debug, знизу Release).

Також, покажемо, як додати всі необхідні .dll-файли до директорії з зібраним проектом. Для цього зберемо проект в Release-режимі. Потім, нам необхідно перейти в директорію, де знаходяться файли Qt. В папці bin нам необхідно викликати командний рядок і запустити на виконання програму windeployqt, передавши їй як параметр повне ім’я нашого виконавчого файлу:

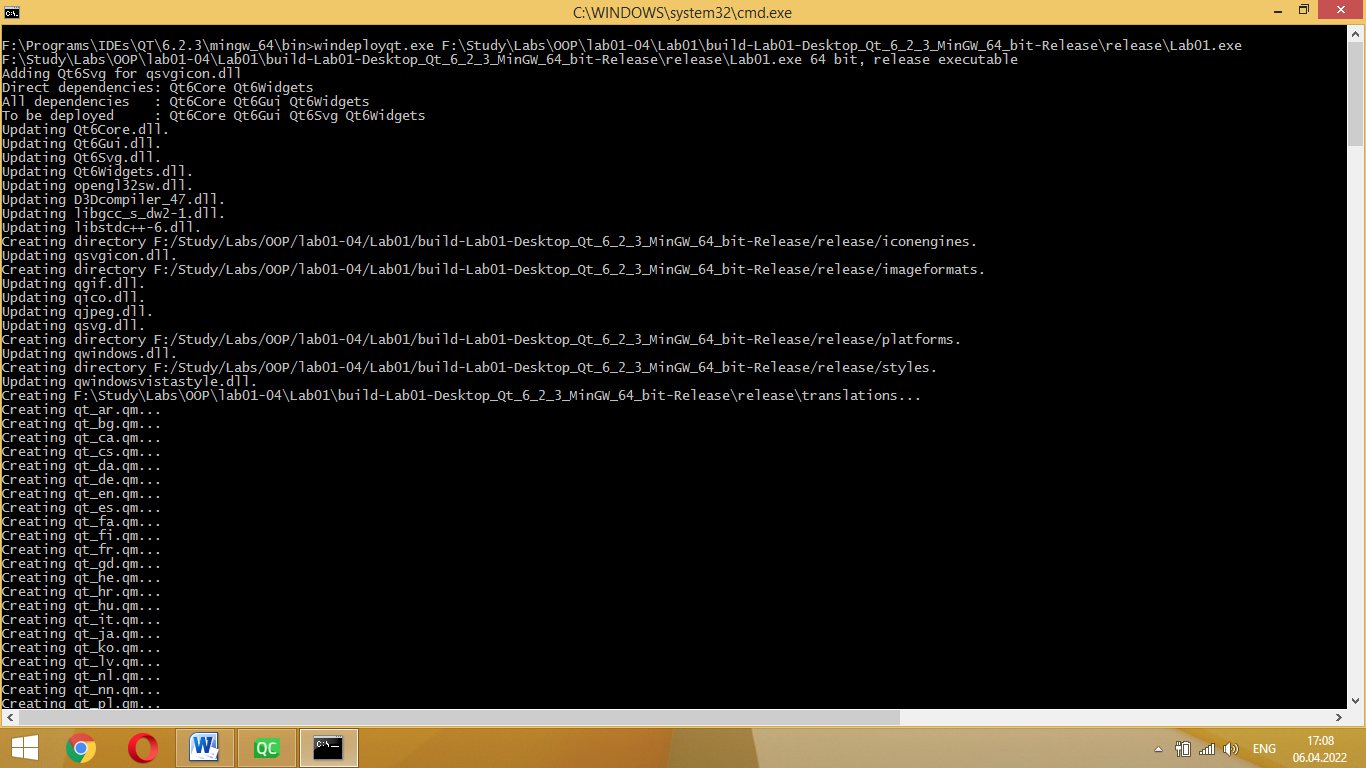


Рис 1.14 Виконання програми windeployqt.exe

І тепер, в папці з виконавчим файлом опинилися всі потрібні бібліотеки:

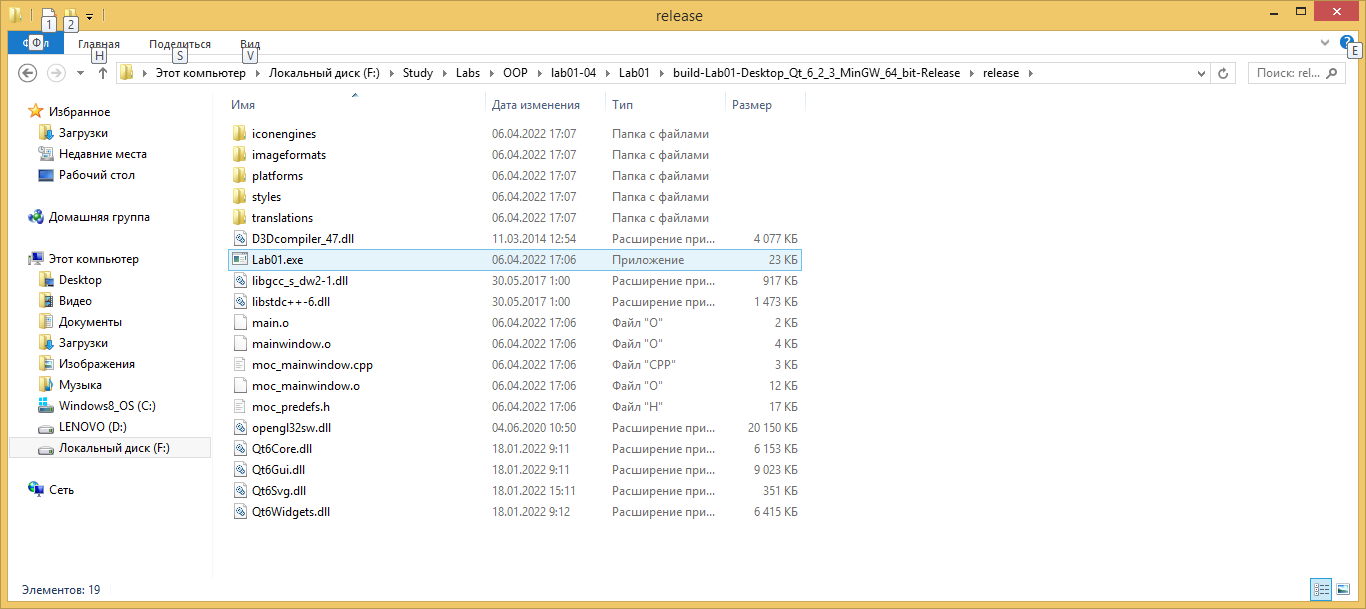


Рис 1.15 Результат

І нашу програму тепер можна передавати користувачу, який не має встановленого Qt на комп'ютері.

Далі, за завданням, необхідно створити калькулятор. Почнемо з форми – вона має наступний вигляд:

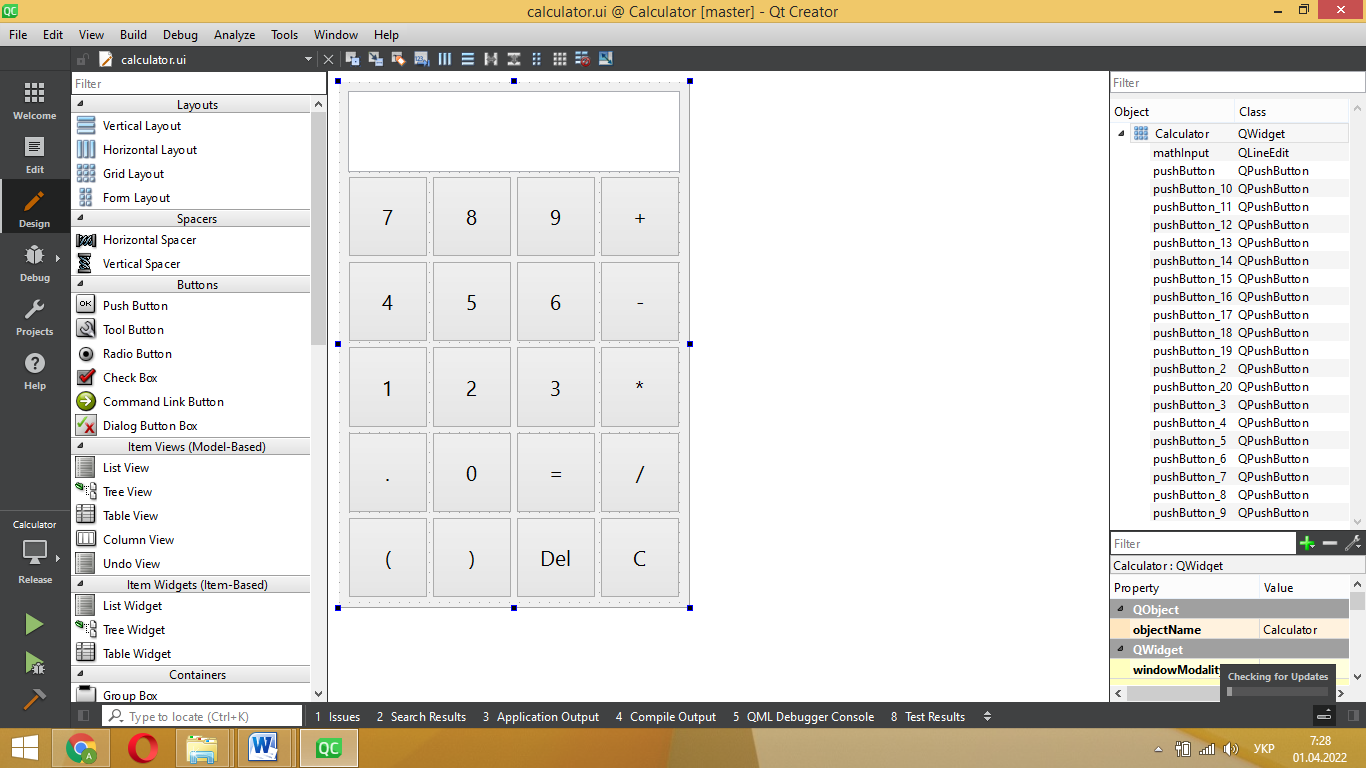


Рис 2.1. Форма калькулятора.

За допомогою натискання правою кнопкою миші на кнопку, і вибору пунку меню “Go to slot” можна встановити обробники(слоти) того чи іншого сигналу. Покажемо код обробника деяких кнопок (показати весь код не є доцільним через його великий розмір)

Код-обробник натиску на кнопку «1»:

void Calculator::**on\_pushButton\_9\_clicked**()

{

char text[256];

strcpy\_s(*text*, 255, ui->mathInput->text().toStdString().c\_str());

strcat(*text*, "1");

ui->mathInput->setText(text);

}

Код-обробник натиску на кнопку «+»:

void Calculator::**on\_pushButton\_6\_clicked**()

{

char text[256];

strcpy\_s(*text*, 255, ui->mathInput->text().toStdString().c\_str());

if(strlen(text))

{

char last = text[strlen(text) - 1];

if(last != '+' && last != '-' && last != '\*' && last != '/' && last != '.')

{

strcat(*text*, "+");

ui->mathInput->setText(text);

}

}

}

Код-обробник натиску на кнопку «=»:

typedef double(\*operation)(double, double);

int **getPrior**(char op)

{

switch (op)

{

case '(':

case ')':

return 0;

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

default:

return 0;

}

}

double **add**(double a, double b)

{

return a + b;

}

double **sub**(double a, double b)

{

return a - b;

}

double **mul**(double a, double b)

{

return a \* b;

}

double **div**(double a, double b)

{

return a / b;

}

operation **getOperation**(char op)

{

switch (op)

{

case '+':

return add;

case '-':

return sub;

case '\*':

return mul;

case '/':

return div;

}

}

void Calculator::**on\_pushButton\_15\_clicked**()

{

char buffer[256];

double stack\_n[256];

int peek\_n = -1;

char stack\_o[256];

int peek\_o = -1;

strcpy\_s(*buffer*, 256, ui->mathInput->text().toStdString().c\_str());

qDebug() << buffer;

double value;

char op;

for (char\* p = buffer; \*p != 0; p++)

{

if (isdigit(\*p) || (\*p == '-' && (p == buffer || \*(p-1) == '+' || \*(p-1) == '-' || \*(p-1) == '\*' || \*(p-1) == '/')))

{

value = strtod(p, *&p*);

p--;

stack\_n[++peek\_n] = value;

qDebug() << "putting number " << value << " to stack";

}

else

{

op = \*p;

if(op == '(') stack\_o[++peek\_o] = op;

else if (op == ')')

{

while (stack\_o[peek\_o] != '(')

{

char op = stack\_o[peek\_o--];

double last = stack\_n[peek\_n];

double prelast = stack\_n[peek\_n - 1];

qDebug() << "Performing: " << prelast << op << last;

value = getOperation(op)(prelast, last);

peek\_n -= 2;

stack\_n[++peek\_n] = value;

}

peek\_o--;

}

else {

while (getPrior(op) <= getPrior(stack\_o[peek\_o]))

{

char op = stack\_o[peek\_o--];

double last = stack\_n[peek\_n];

double prelast = stack\_n[peek\_n - 1];

qDebug() << "Performing: " << prelast << op << last;

value = getOperation(op)(prelast, last);

peek\_n -= 2;

stack\_n[++peek\_n] = value;

}

qDebug() << "putting operation " << op << " to stack";

stack\_o[++peek\_o] = op;

}

}

}

while (peek\_o >= 0)

{

char op = stack\_o[peek\_o--];

double last = stack\_n[peek\_n];

double prelast = stack\_n[peek\_n - 1];

qDebug() << "Performing: " << prelast << op << last;

value = getOperation(op)(prelast, last);

peek\_n -= 2;

stack\_n[++peek\_n] = value;

}

sprintf(*buffer*, "%.2lf", value);

ui->mathInput->setText(buffer);

qDebug() << "------------------------------------------------------";

}

Ну і код-обробник натиску на кнопку «С»:

void Calculator::**on\_pushButton\_20\_clicked**()

{

ui->mathInput->setText("");

}

Запустимо на виконання:

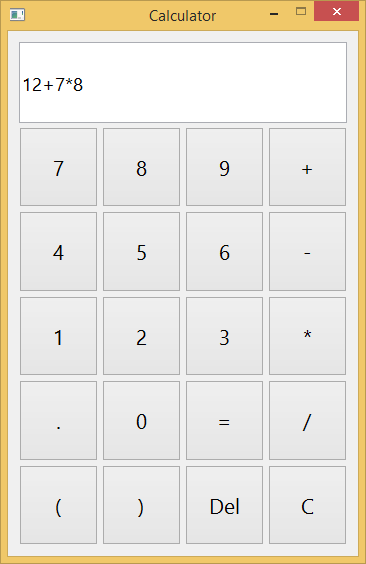
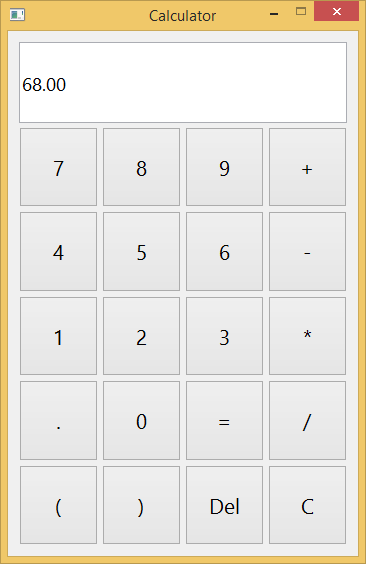
 

Рис 2.2 Результат виконання.

Отже, лабараторну роботу №2 було виконано.

Для виконання наступної нам теж необхідно спочатку створити форму, яка в результаті має наступний вигляд:

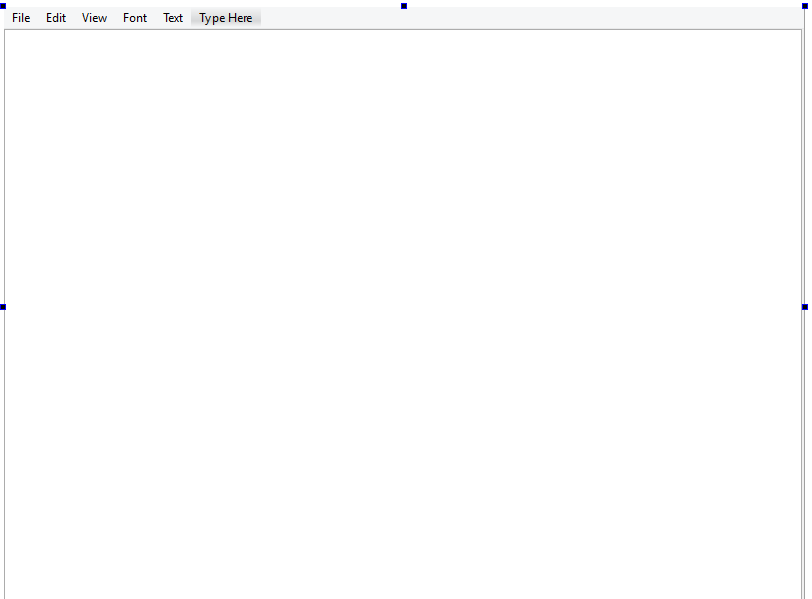


Рис 3.1 Форма головного вікна.

А також нам необхідно створити форму допоміжного діалогу для реалізації пошуку/заміни:

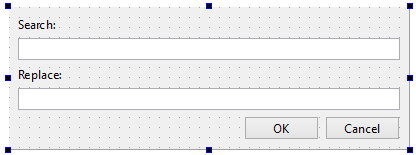


Рис 3.2 Форма діалогу.

Вся робота довкола головного вікна крутиться довколо елемента QTabWidget( біла область на рис. 3.1). А всі функціональні можливості винесені в меню згори. Покажемо обробники подій деяких з цих пунктів:

Додавання нового файлу:

void EditorWindow::**on\_actionNew\_text\_file\_triggered**()

{

QWidget \*tabWidget = new QWidget;

QVBoxLayout \*tabLayout = new QVBoxLayout;

QTextEdit \*tabTextEdit = new QTextEdit;

connect(tabTextEdit, &QTextEdit::textChanged, this, &EditorWindow::on\_textEditTextChanged);

tabLayout->setContentsMargins(2,2,2,2);

tabLayout->addWidget(*tabTextEdit*);

tabWidget->setLayout(*tabLayout*);

ui->tabContainer->addTab(*tabWidget*, "[Text] Untitled");

ui->tabContainer->setCurrentIndex(ui->tabContainer->count() - 1);

}

Відкриття файлу:

void EditorWindow::**on\_actionOpen\_file\_triggered**()

{

QString filename = QFileDialog::getOpenFileName(this, "Open file", "C:\\", "Text files (\*.txt);;Image files(\*.png \*.jpg)");

if(filename.size() == 0) return;

for(int i = 0; i < ui->tabContainer->count(); i++)

{

qDebug() << filename << " " << ui->tabContainer->tabText(i);

if(ui->tabContainer->tabText(i).endsWith(filename))

{

ui->tabContainer->setCurrentIndex(i);

return;

}

}

QWidget \*tabWidget = new QWidget;

QVBoxLayout \*tabLayout = new QVBoxLayout;

tabLayout->setContentsMargins(2,2,2,2);

if(filename.endsWith(".txt"))

{

QTextEdit \*tabTextEdit = new QTextEdit;

FILE \*openedFile = fopen(filename.toStdString().data(), "r");

char currentChar;

QString fileText;

while((currentChar = getc(*openedFile*)) != EOF)

{

fileText.append(currentChar);

}

tabTextEdit->setText(fileText);

tabLayout->addWidget(*tabTextEdit*);

tabWidget->setLayout(*tabLayout*);

ui->tabContainer->addTab(*tabWidget*, "[Text] " + filename);

ui->tabContainer->setCurrentIndex(ui->tabContainer->count() - 1);

connect(tabTextEdit, &QTextEdit::textChanged, this, &EditorWindow::on\_textEditTextChanged);

fclose(*openedFile*);

}

else

{

QScrollArea \*tabScrollArea = new QScrollArea;

QPixmap tabImage(filename);

QLabel \*tabLabel = new QLabel;

tabLabel->setScaledContents(true);

tabLabel->setPixmap(tabImage);

tabScrollArea->setAlignment(Qt::AlignCenter);

tabScrollArea->setWidget(*tabLabel*);

tabLayout->addWidget(*tabScrollArea*);

tabWidget->setLayout(*tabLayout*);

ui->tabContainer->addTab(*tabWidget*, "[Image] " + filename);

ui->tabContainer->setCurrentIndex(ui->tabContainer->count() - 1);

}

}

Збереження файлу:

void EditorWindow::**on\_actionSave\_file\_triggered**()

{

QString tabName = ui->tabContainer->tabText(ui->tabContainer->currentIndex());

if(!tabName.startsWith("[Text]")) return;

if(tabName.endsWith('\*'))

{

tabName.chop(1);

ui->tabContainer->setTabText(ui->tabContainer->currentIndex(), tabName);

}

if(tabName == "[Text] Untitled")

{

on\_actionSave\_as\_triggered();

}

else

{

char \*filename = strchr(tabName.toStdString().data(), ' ') + 1;

FILE \*openedFile = fopen(filename, "w");

QTextEdit\* textField = qobject\_cast<QTextEdit\*>(*ui->tabContainer->currentWidget()->layout()->itemAt(*0*)->widget()*);

fputs(textField->toPlainText().toStdString().c\_str(), *openedFile*);

fclose(*openedFile*);

}

}

Збільшення зображення:

static double zoomFactor = 1;

void EditorWindow::**on\_actionZoom\_in\_triggered**()

{

if(zoomFactor > 3) return;

QString tabName = ui->tabContainer->tabText(ui->tabContainer->currentIndex());

if(!tabName.startsWith("[Image]")) return;

QScrollArea \*scrollWidget = qobject\_cast<QScrollArea\*>(*ui->tabContainer->currentWidget()->layout()->itemAt(*0*)->widget()*);

QLabel \*label = qobject\_cast<QLabel\*>(*scrollWidget->widget()*);

zoomFactor \*= 1.25;

qDebug() << zoomFactor;

label->resize(label->size()\*1.25);

}

Зміна шрифту додатку:

void EditorWindow::**on\_actionGlobally\_triggered**()

{

bool gotFont;

QFont newFont = QFontDialog::getFont(*&gotFont*);

if(gotFont) setFont(newFont);

}

Є також багато інших функцій, але їх код я не став додавати сюди по причині його громіздкості. Запустимо проект на виконання:

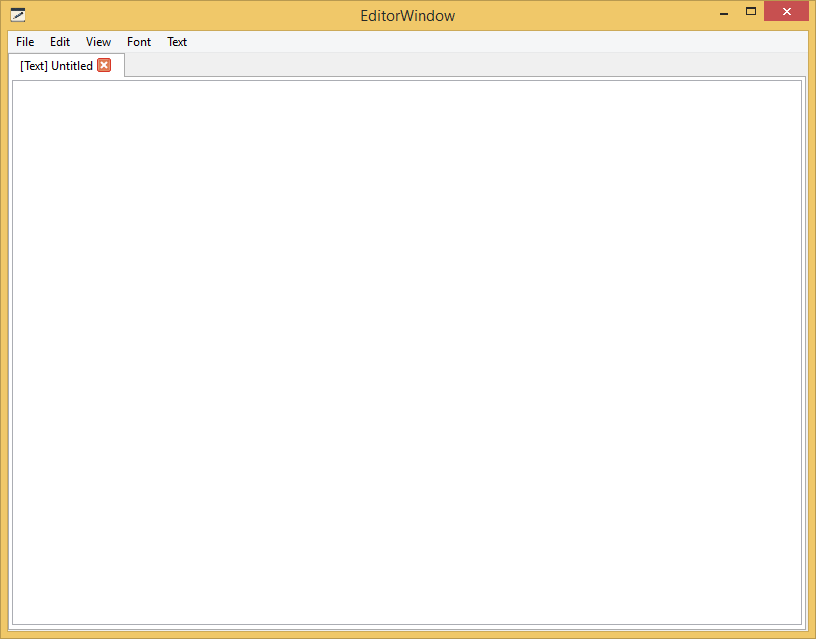


Рис 3.3 Результат запуску і створення нового тектового файлу.

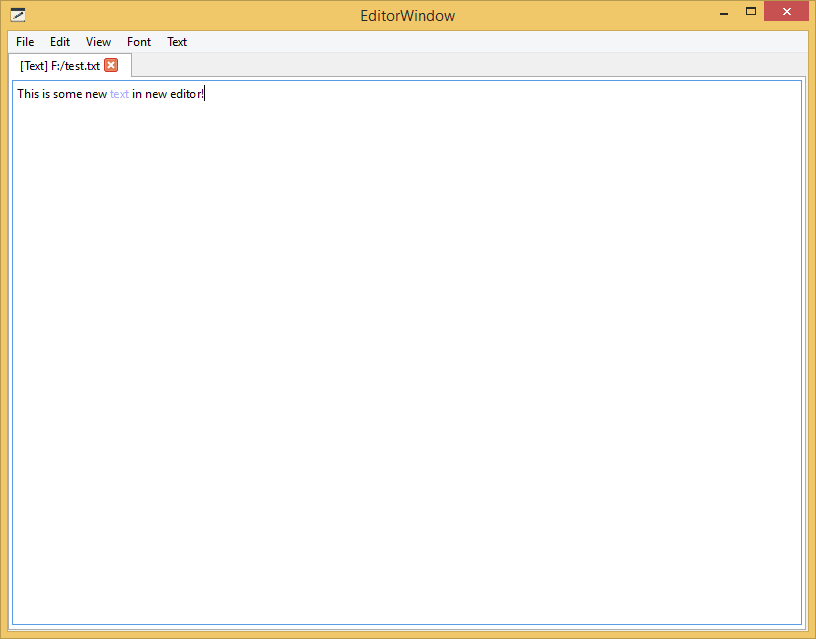


Рис 3.4 Введення тексту та його збереження.

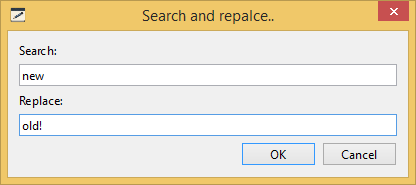


Рис 3.5 Діалог заміни тексту.

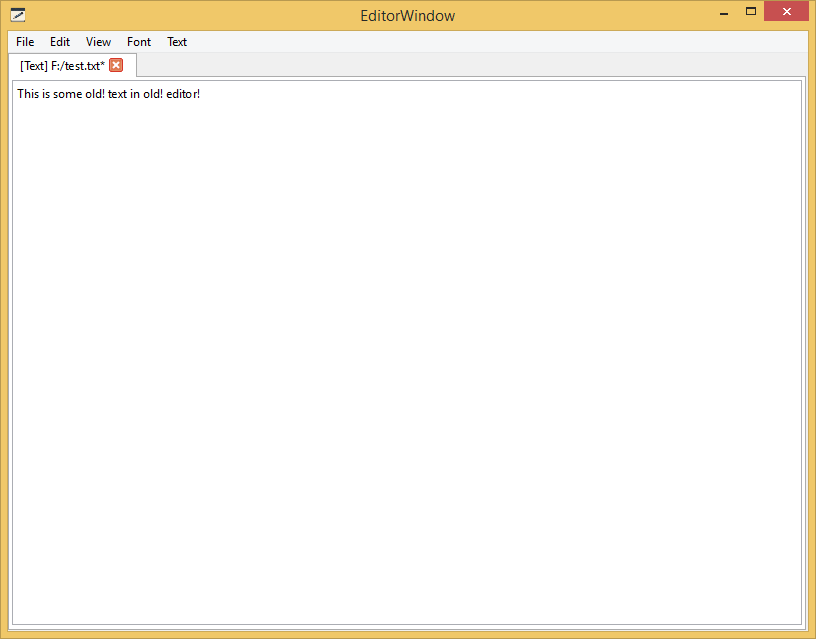


Рис 3.6 Резултат заміни тексту.

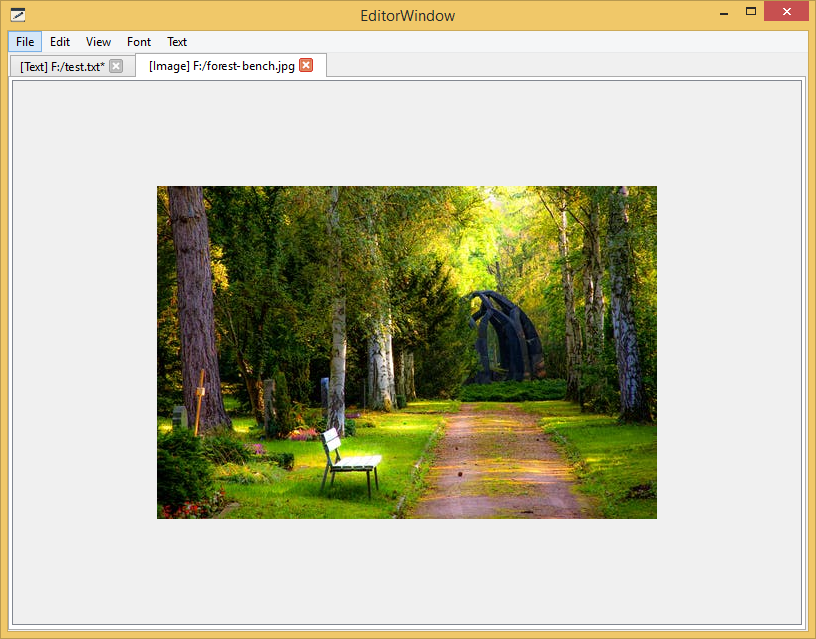


Рис 3.7 Відкриття зображення

Отож, завдання лабараторної роботи №3 виконано.

Для виконання завдання ЛР№4 нам теж необхідно спочатку створити форму. Оскільки створювати будемо гру «Сапер» , вона буде мати наступний вигляд:

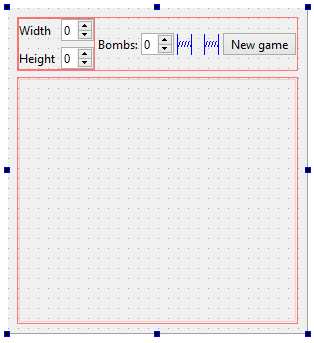


Рис 4.1 Вікно гри сапер

Але, як наслідок динаміних налаштувань гри, велику частину інтерфейсу необхідно будувати в коді. Покажемо деякі функціональні можливості в коді:

Запуск гри, створення початкового поля:

int horizontalSize = 10;

int verticalSize = 10;

int gameOver;

int firstClick;

int bombsCount;

int cellsOpened = 0;

int leftMouseBtnClicked = 0;

QPushButton \*\*\*buttons;

int \*\*adjacent;

MineSweeper::**MineSweeper**(QWidget \*parent)

: QWidget(*parent*)

, ui(new Ui::MineSweeper)

{

ui->setupUi(this);

ui->bombCounter->setValue(10);

ui->widthCounter->setValue(10);

ui->heightCounter->setValue(10);

gameOver = 1;

srand(time(0));

buttons = new QPushButton\*\*[verticalSize];

adjacent = new int\*[verticalSize];

for(int i = 0; i < horizontalSize; i++)

{

buttons[i] = new QPushButton\*[horizontalSize];

adjacent[i] = new int[horizontalSize];

}

for(int i = 0; i < verticalSize; i++)

{

for (int j = 0; j < horizontalSize; j++ ) {

QPushButton \*btn = new QPushButton(this);

btn->setMinimumSize(32, 32);

btn->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

ui->gameField->addWidget(*btn*, i, j);

buttons[i][j] = btn;

btn->setDisabled(true);

adjacent[i][j] = 0;

connect(btn, SIGNAL(clicked()), this, SLOT(buttonClicked()));

}

}

}

Натиск на елемент поля:

void MineSweeper::**buttonClicked**()

{

if(gameOver) return;

int BtnX = -1, BtnY = -1;

QPushButton \*clickedButton = qobject\_cast<QPushButton\*>(*sender()*);

if(!clickedButton->isEnabled()) return;

for(int i = 0; i < verticalSize; i++)

{

for (int j = 0; j < horizontalSize; j++ ) {

if(buttons[i][j] == clickedButton)

{

BtnY = i;

BtnX = j;

}

}

}

qDebug() << "Clicked" << BtnY << BtnX;

if(firstClick)

{

firstClick = 0;

for(int i = 0; i < bombsCount; i++)

{

int x,y;

do

{

x = rand() % horizontalSize;

y = rand() % verticalSize;

} while(adjacent[y][x] == -1 || (x == BtnX && y == BtnY));

adjacent[y][x] = -1;

}

for(int i = 0; i < verticalSize; i++)

{

for(int j = 0; j < horizontalSize; j++)

{

if(adjacent[i][j] == -1) continue;

int surroundingBombs = 0;

for(int adjY = std::max(0, i-1); adjY < std::min(verticalSize, i+2); adjY++)

{

for(int adjX = std::max(0, j-1); adjX < std::min(horizontalSize, j+2); adjX++)

{

if(adjacent[adjY][adjX] == -1) surroundingBombs++;

}

}

adjacent[i][j] = surroundingBombs;

}

}

}

if(adjacent[BtnY][BtnX] == -1)

{

gameOver = 1;

ui->statusLabel->setText("Game over!");

for(int i = 0; i < verticalSize; i++)

{

for(int j = 0; j < horizontalSize; j++)

{

if(adjacent[i][j] == -1)

{

buttons[i][j]->setText("X");

}

}

}

return;

}

buttons[BtnY][BtnX]->setDisabled(true);

cellsOpened++;

if(adjacent[BtnY][BtnX] == 0)

{

if(BtnX > 0) buttons[BtnY][BtnX - 1]->click();

if(BtnX < horizontalSize - 1) buttons[BtnY][BtnX + 1]->click();

if(BtnY > 0) buttons[BtnY - 1][BtnX]->click();

if(BtnY < verticalSize - 1) buttons[BtnY + 1][BtnX]->click();

}

else

{

buttons[BtnY][BtnX]->setText(QString::number(adjacent[BtnY][BtnX]));

QFont font = buttons[BtnY][BtnX]->font();

font.setPointSize(adjacent[BtnY][BtnX] \* 2 + 6);

buttons[BtnY][BtnX]->setFont(font);

}

if(cellsOpened == verticalSize \* horizontalSize - bombsCount)

{

gameOver = 1;

ui->statusLabel->setText("You win!");

for(int i = 0; i < verticalSize; i++)

{

for(int j = 0; j < horizontalSize; j++)

{

if(adjacent[i][j] == -1)

{

buttons[i][j]->setText("X");

}

}

}

return;

}

}

Початок нової гри:

void MineSweeper::**on\_newGameButton\_clicked**()

{

cellsOpened = 0;

firstClick = 1;

gameOver = 0;

bombsCount = ui->bombCounter->value();

ui->statusLabel->setText("");

while (auto item = ui->gameField->*takeAt*(0))

{

ui->gameField->removeItem(*item*);

delete item->*widget*();

}

for(int i = 0; i < verticalSize; i++)

{

ui->gameField->setRowStretch(i, 0);

}

for(int i = 0; i < horizontalSize; i++)

{

ui->gameField->setColumnStretch(i, 0);

}

for(int i = 0; i < verticalSize; i++)

{

delete[] buttons[i];

delete[] adjacent[i];

}

delete[] buttons;

delete[] adjacent;

verticalSize = ui->heightCounter->value();

horizontalSize = ui->widthCounter->value();

buttons = new QPushButton\*\*[verticalSize];

adjacent = new int\*[verticalSize];

for(int i = 0; i < verticalSize; i++)

{

buttons[i] = new QPushButton\*[horizontalSize];

adjacent[i] = new int[horizontalSize];

}

qDebug() << ui->gameField->*count*();

for(int i = 0; i < verticalSize; i++)

{

ui->gameField->setRowStretch( i, 1);

for (int j = 0; j < horizontalSize; j++ ) {

if(i == 0)

ui->gameField->setColumnStretch(j, 1);

QPushButton \*btn = new QPushButton(this);

btn->setMinimumSize(32, 32);

btn->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

ui->gameField->addWidget(*btn*, i, j);

buttons[i][j] = btn;

adjacent[i][j] = 0;

connect(btn, SIGNAL(clicked()), this, SLOT(buttonClicked()));

}

}

for(int i = 0; i < verticalSize; i++)

{

for(int j = 0; j < horizontalSize; j++)

{

buttons[i][j]->setDisabled(false);

buttons[i][j]->setText("");

adjacent[i][j] = 0;

}

}

}

Запустимо проект на виконання:



Рис 4.2 Запуск программи.



Рис 4.3 Перший натиск на кнопку.

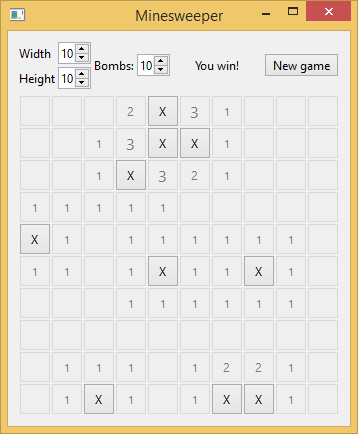


Рис 4.4 Перемога у грі.

Отже, лабараторну роботу №4 було також виконано.

**Висновок:** виконавши лабараторні роботи №1-4, я ознайомився з середовищем розробки, навчився використовувати різні вбудовані компоненти, обробляти сигнали, що йдуть від них і в цілому працювати з інтерфейсом. Ми реалізували 3 повноцінних програми: калькулятор, редактор тексту/переглядач зображень, гру «Сапер». Ці додатки можна цілком використовувати в повсякденному житті.