Лекція 4

ПРОГРАМУВАННЯ GUI ІЗ ЗАСТОСУВАННЯМ QT

Лекція 4. Програмування GUI із застосуванням Qt

План

1. Огляд бібліотеки Qt.
2. Qt Designer.
3. Створення проекту з GUI в Qt.
4. Редагування форми в Qt Designer.

1. Огляд бібліотеки Qt

Qt – це **кросплатформна бібліотека**, що реалізована на мові програмування C++ та призначена для створення різноманітних прикладних програм з високоякісним графічним інтерфейсом користувача. Програми, написані із застосуванням Qt, можуть без переробки працювати в середовищі сучасних операційних систем Windows, Linux і maxOS.

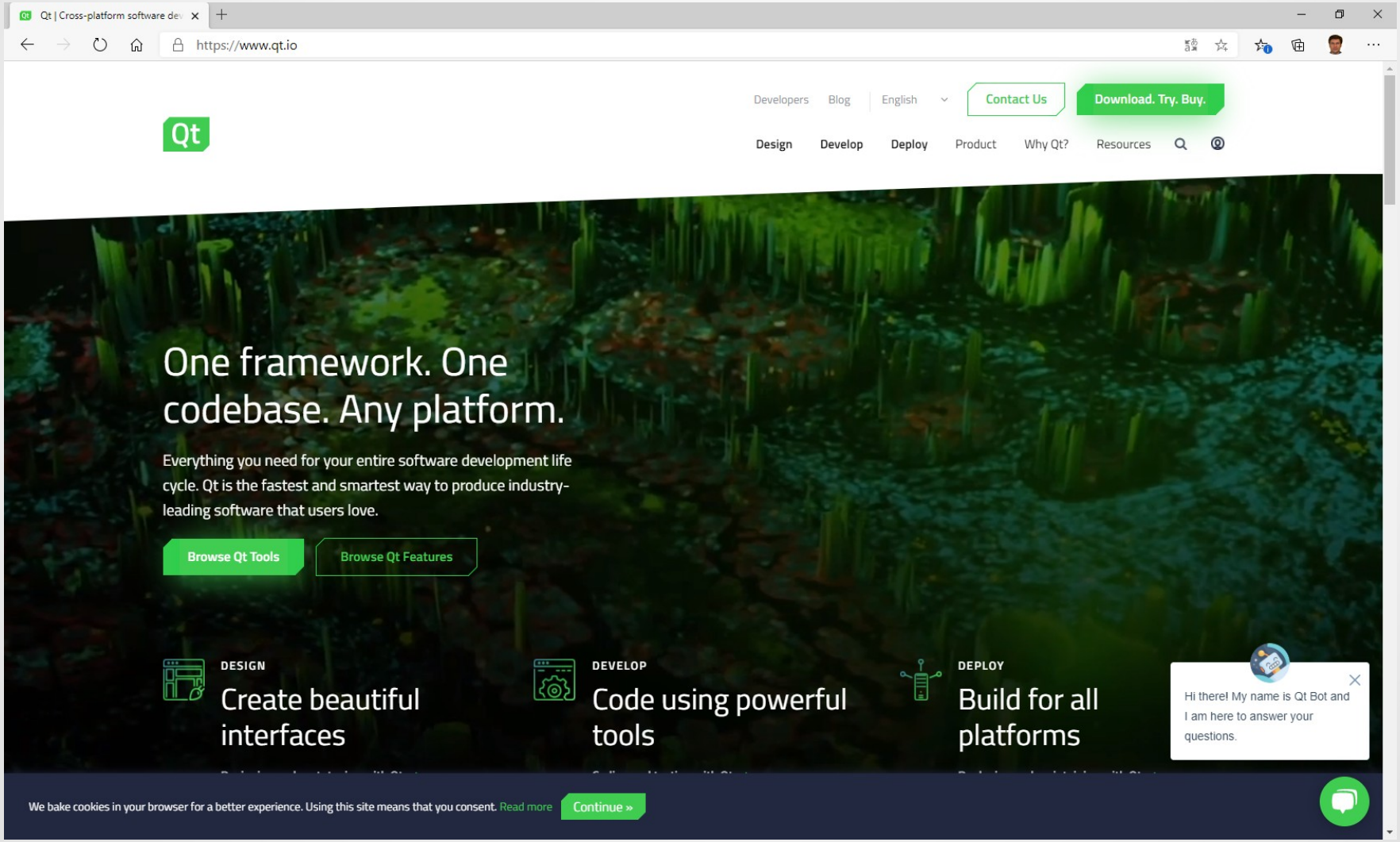
Бібліотека Qt була розроблена в 1996 році і з тих пір активно розвивається. Її використовують такі відомі IT-корпорації, як Autodesk, Google, Microsoft, Nokia, Panasonic, Siemens, Walt Disney Animation Studios та інші.

Крім програмування GUI Qt дозволяє:

- створювати багатопотокові програми;
- розробляти мережеві програми;
- розробляти програми з 2D- і 3D-графікою (у т. ч. і з використанням OpenGL);
- програмувати бази даних (у т. ч. з використанням SQL);
- працювати з форматами XML та JSON;
- розробляти веб-додатки;
- працювати з мультимедіа;
- взаємодіяти з ActiveX і COM (для Windows-розробників);
- і багато іншого.

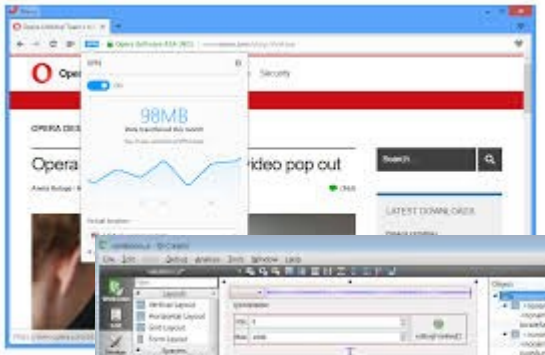
1. Огляд бібліотеки Qt

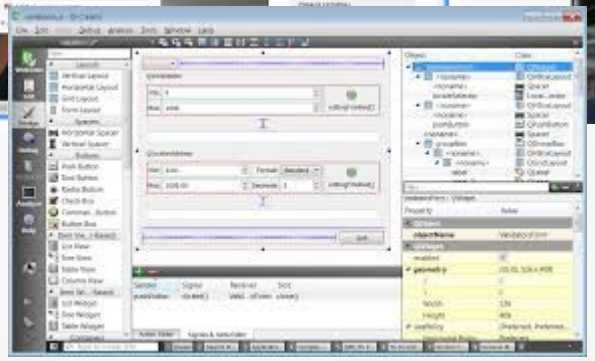
Офіційний веб-сайт Qt: https://www.qt.io/

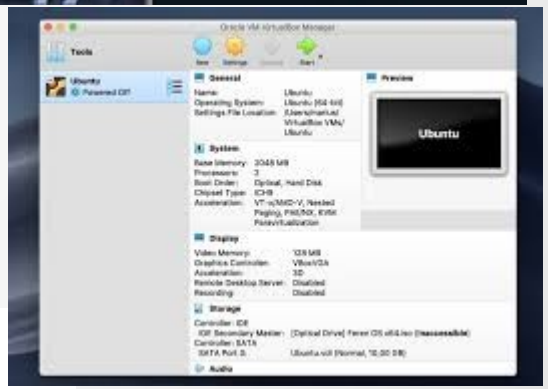


1. Огляд бібліотеки Qt

За допомогою Qt розроблено такі програми, як: KDE; Opera, Skype, Adobe Photoshop Album, Google Earth, VirtualBox, VCL media player і багато інших.

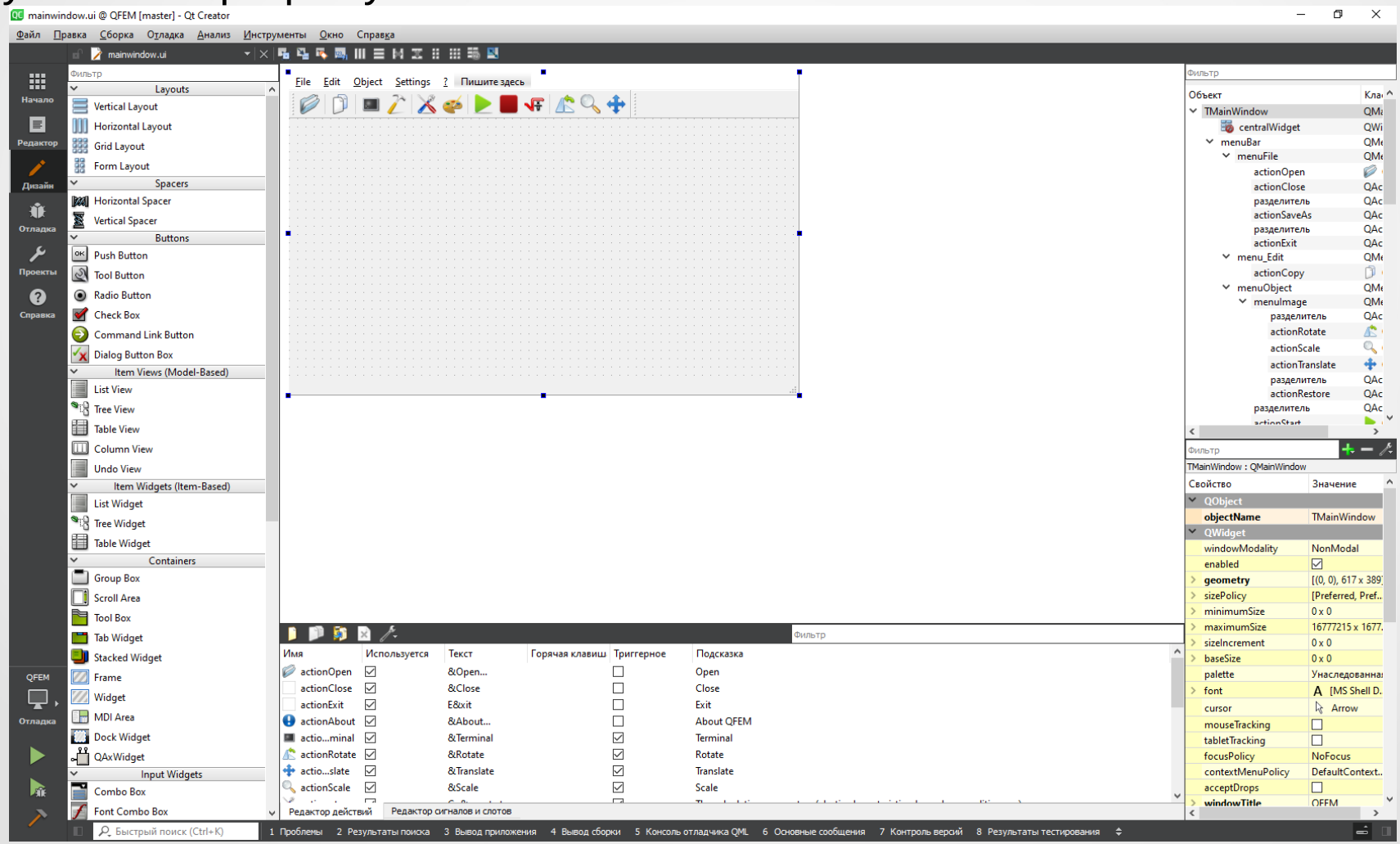






1. Огляд бібліотеки Qt

Широкою популярністю також користується багатоплатформна IDE Qt Creator, яка базується на використанні бібліотеки Qt і підтримує технологію візуального програмування.



1. Огляд бібліотеки Qt

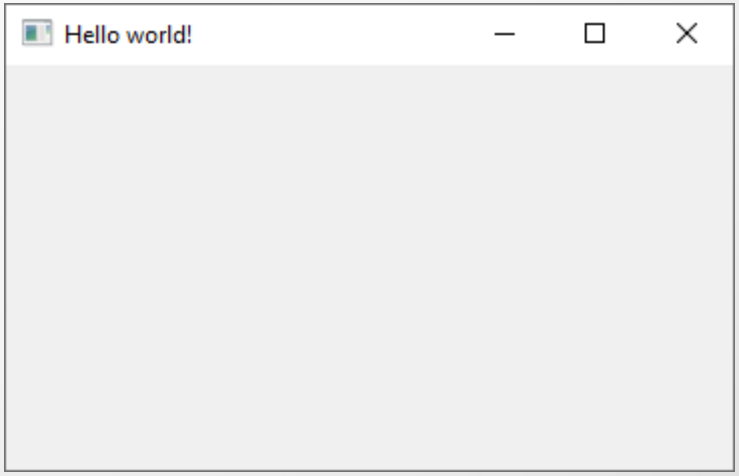
Найпростішу програму на Qt з GUI можна, наприклад, написати таким чином:

Початковий код

```
#include <QtWidgets>

int main(int argc, char **argv)
{
    QApplication app(argc, argv);
    QMainWindow mw;

    mw.setWindowTitle("Hello world!");
    mw.show();
    return app.exec();
}
```

Файл налаштувань компіляції

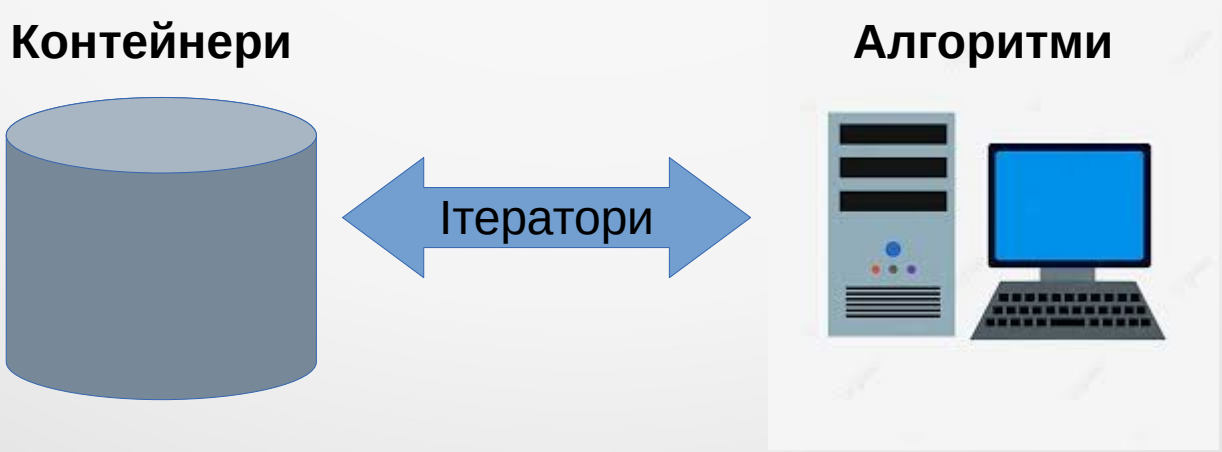
```
QT += gui widgets

CONFIG += c++17

SOURCES += \
        main.cpp
```



1. Огляд бібліотеки Qt

Окрім засобів для розробки GUI бібліотека Qt має дуже багато інших можливостей. Так, наприклад, для зберігання і обробки груп різним способом організованих даних (колекцій) Qt підтримує спеціальну бібліотеку контейнерів **Tulip**, яка є частиною ядра Qt і за своїм функціоналом схожа на STL.

Класи, що утворюють Tulip, розташовані в модулі **QtCore**. Вони реалізують найпоширеніші **контейнери** (стеки, черги, словники, списки тощо), **ітератори** для доступу до елементів контейнерів, а також стандартні **алгоритми** обробки даних (пошук, сортування, різноманітні маніпуляції і т. і.).



2. Qt Designer

Для **швидкої розробки програм** (**RAD** – Rapid Application Development) з використанням бібліотеки Qt використовується спеціальна програма **Qt Designer**. Її можна використовувати як окремо, так і спільно з **Qt Creator**.

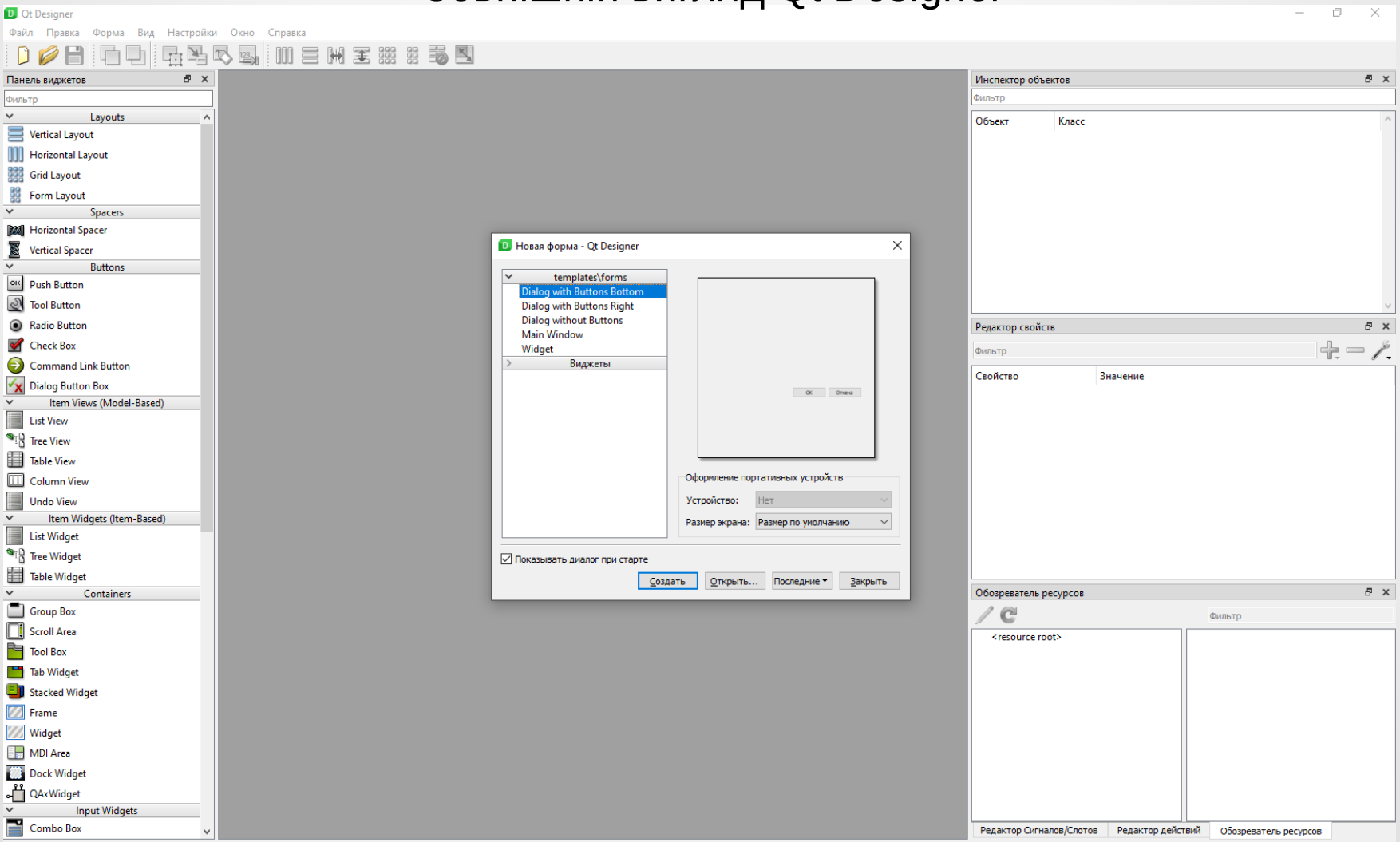
Ця програма призначена для дизайну інтерфейсу програми. Принцип її роботи відповідає **WYSIWYG** (What You See Is What You Get), що можна перевести як “**що бачиш, то і отримаєш**”.

Qt Designer дає можливість швидко створювати прототипи програм, які базуються на **головних вікнах**, що мають **меню**, **панель інструментів** і т. п.

Створені в програмі Qt Designer файли опису інтерфейсу (у форматі **XML**) можна конвертувати в вихідний код на мові C++.

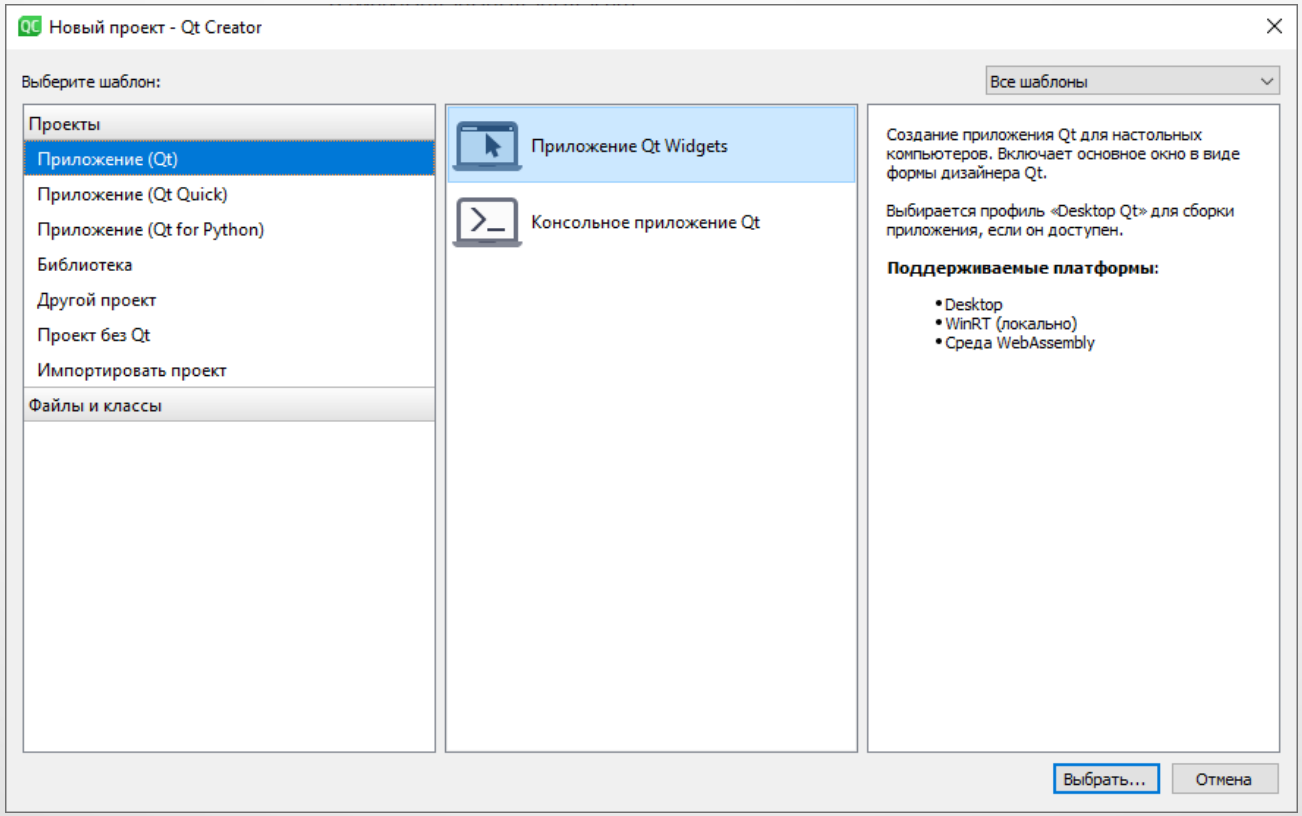
2. Qt Designer

Зовнішній вигляд Qt Designer



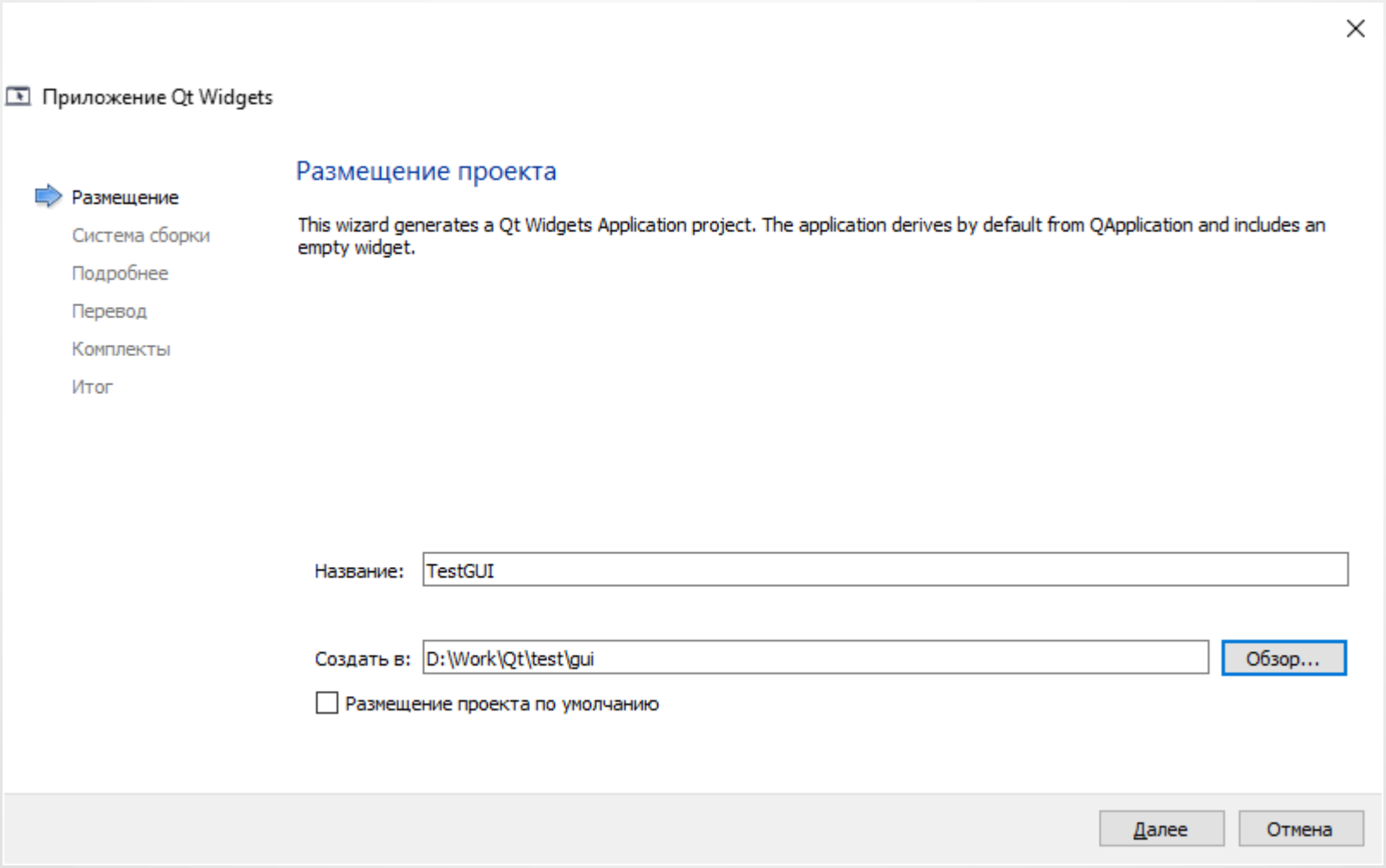
3. Створення проекту з GUI в Qt

Швидким способом створення проекту з графічним інтерфейсом користувача в Qt Creator є наступна послідовність дій:
1) виконати команду головного меню “Файл | Создать файл или проект...”;
2) вибрати шаблон “Приложение (Qt)“ => “Приложение Qt Widgets”;



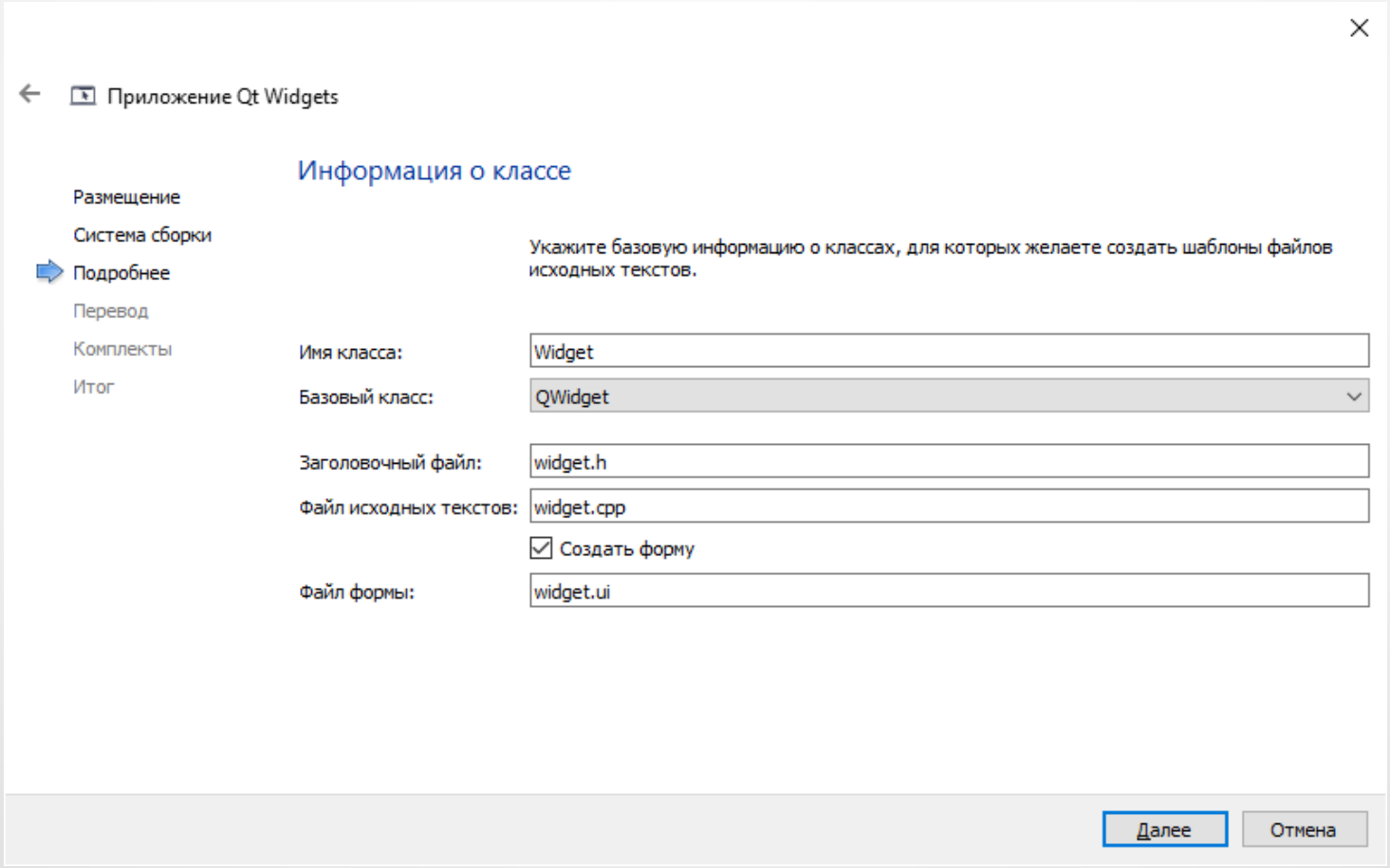
3. Створення проекту з GUI в Qt

3) вказати назву проекту і його місце розташування;



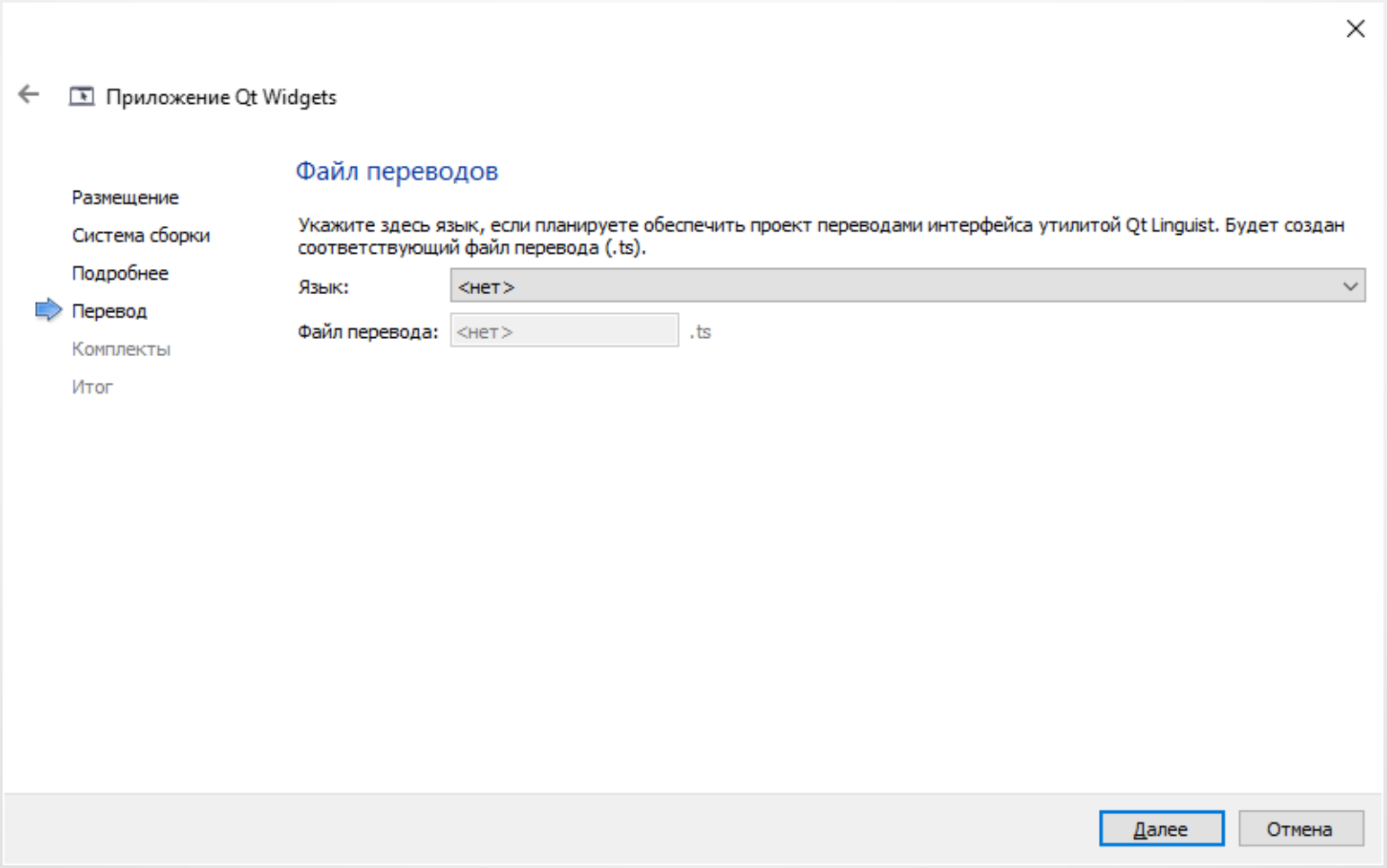
3. Створення проекту з GUI в Qt

4) вибрати систему збирання (оптимально – **qmake**);
5) задати інформацію про базовий клас, на основі якого буде будується інтерфейс головного вікна майбутньої програми;



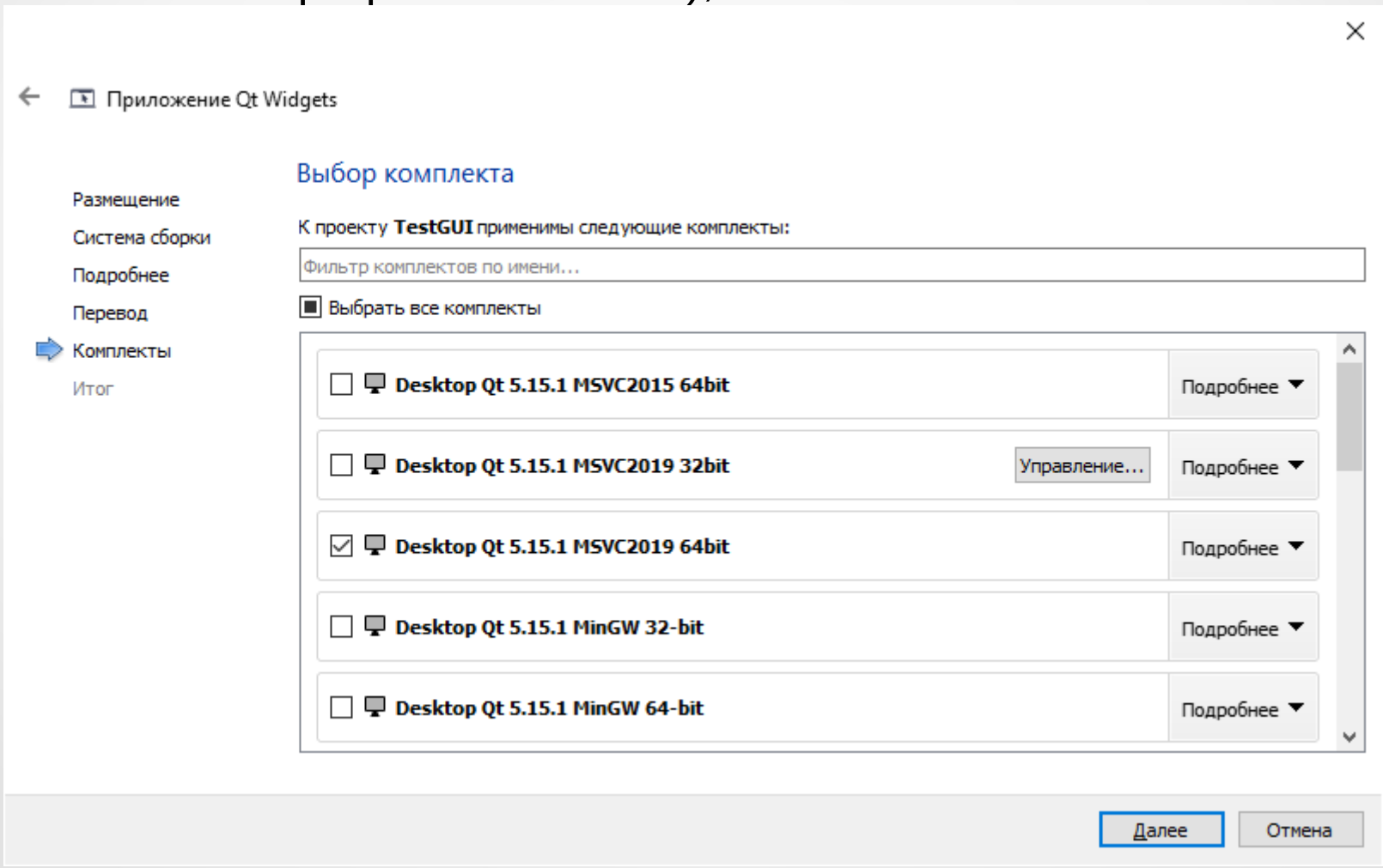
3. Створення проекту з GUI в Qt

6) вибрати при необхідності файл перекладів, що містить різні мови інтерфейсу програми;



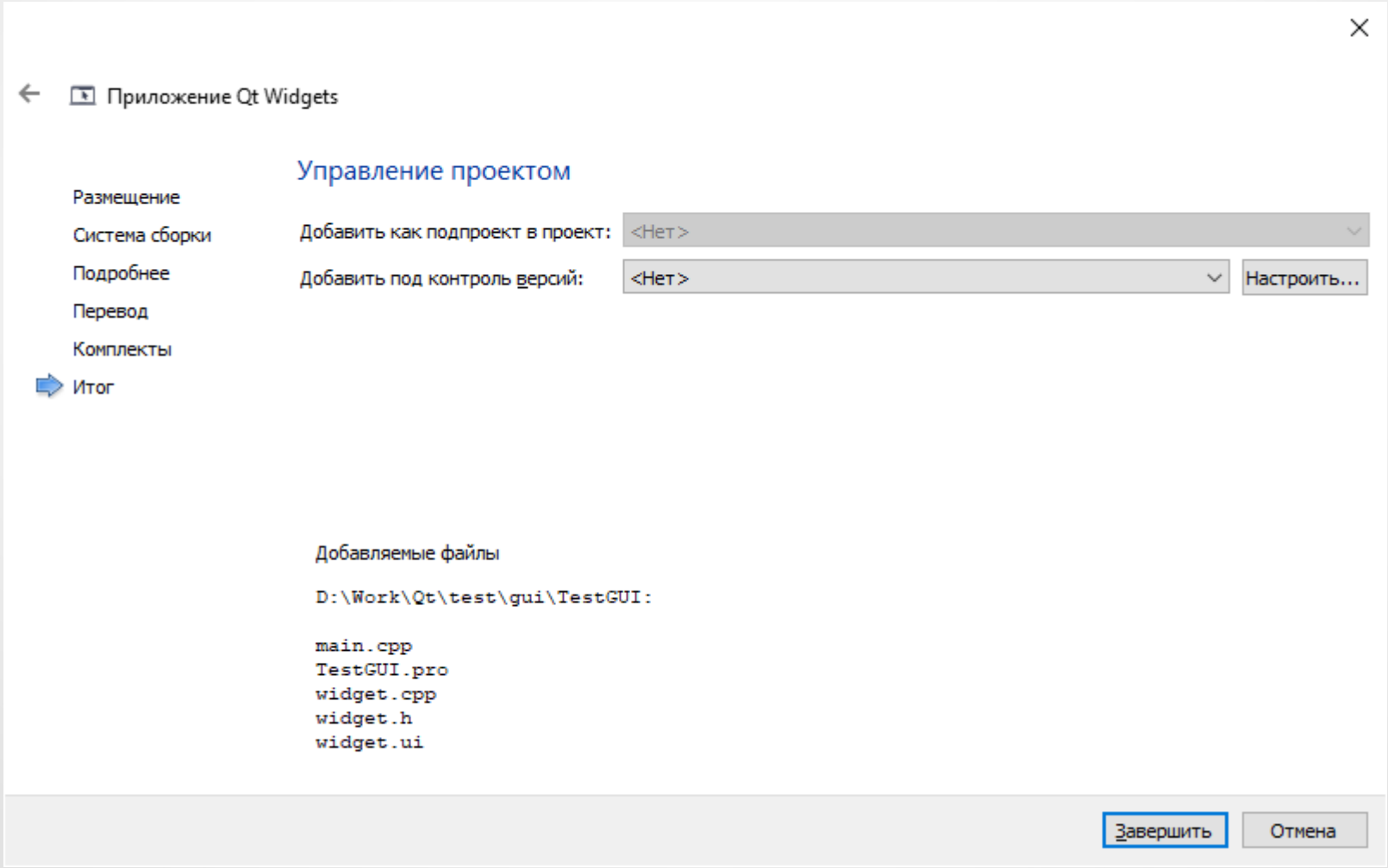
3. Створення проекту з GUI в Qt

7) вибрати комплект компілятора і налагоджувача (залежить від вживаної ОС, встановлених в ній програм і таке інше);



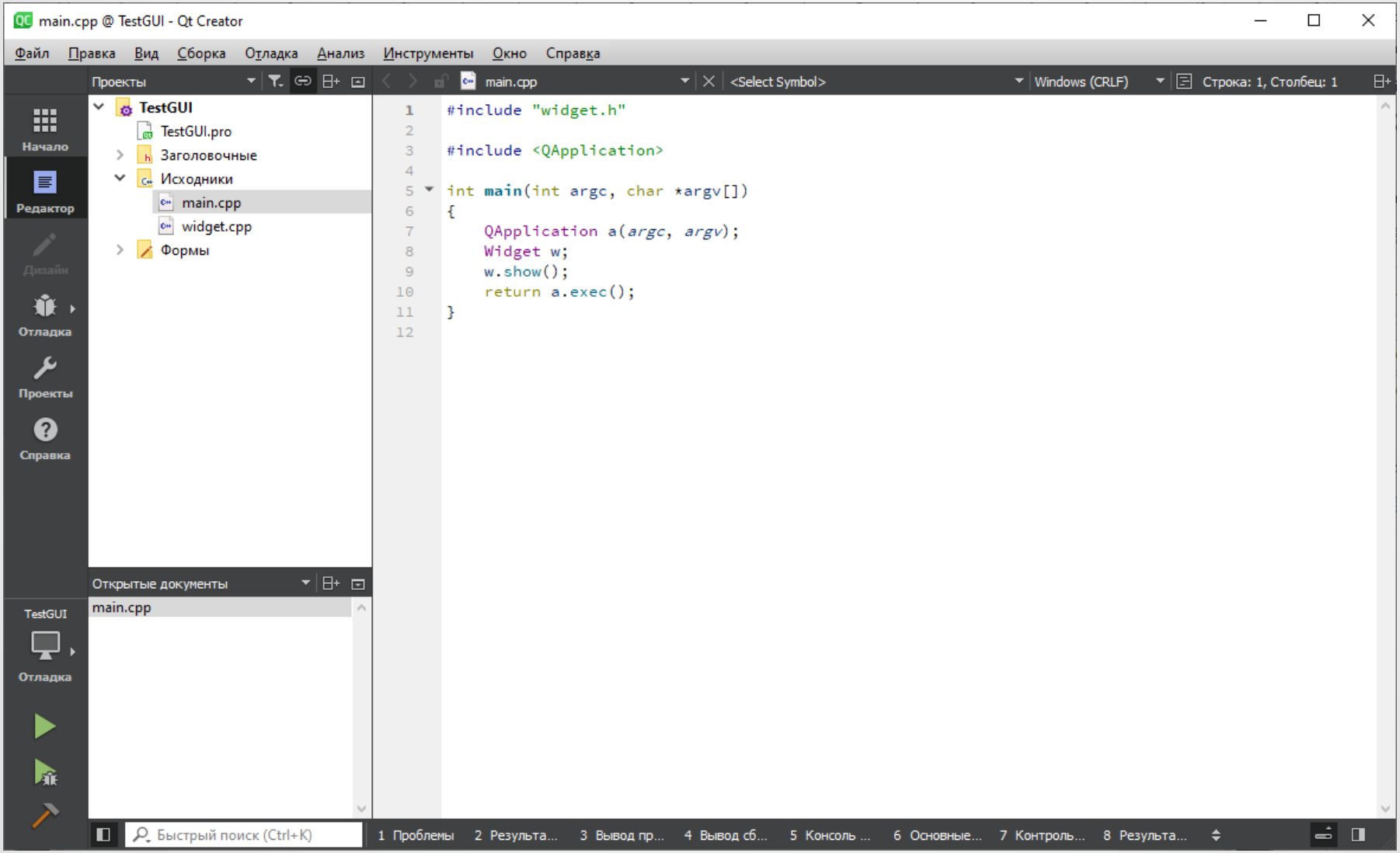
3. Створення проекту з GUI в Qt

8) вибрати спосіб управління проектом;



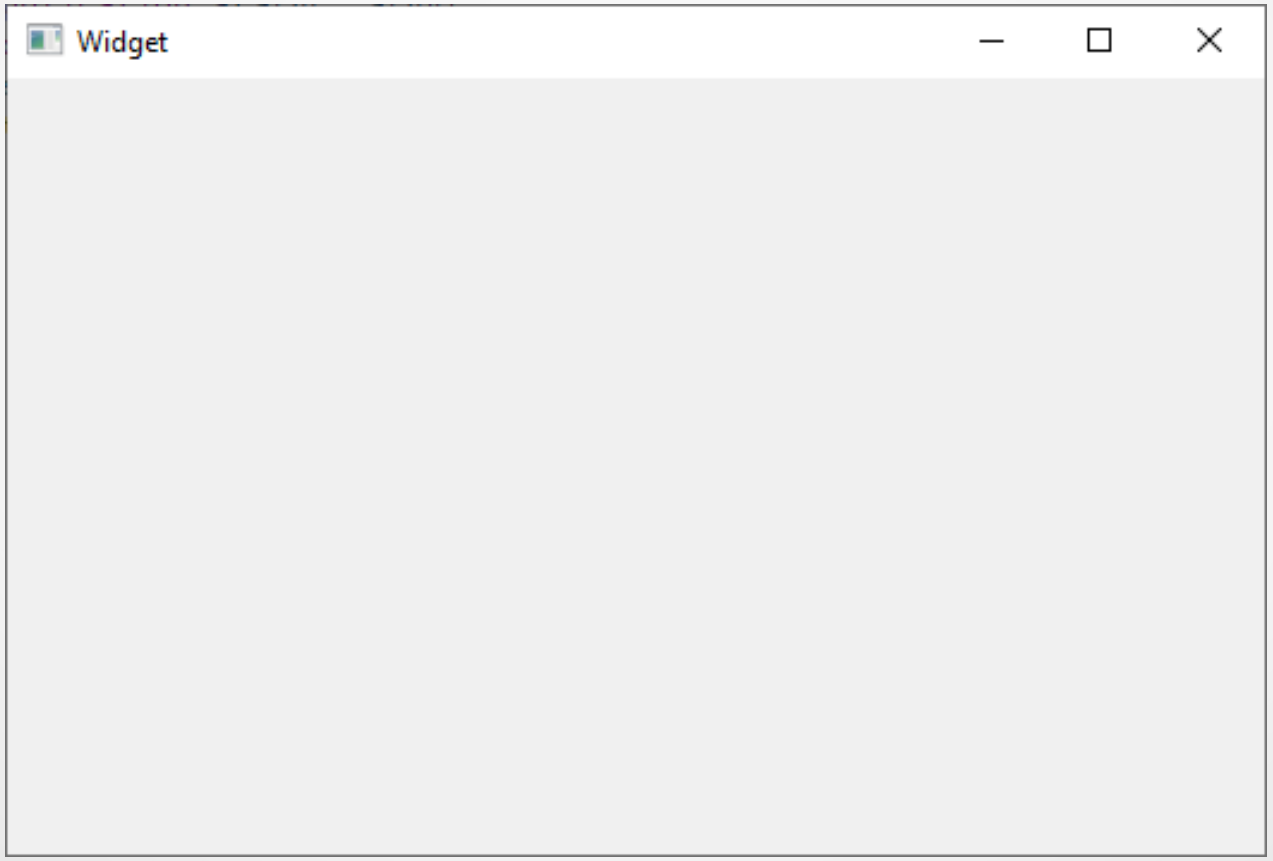
3. Створення проекту з GUI в Qt

Після чого новостворений проект буде відкрито в Qt Creator.



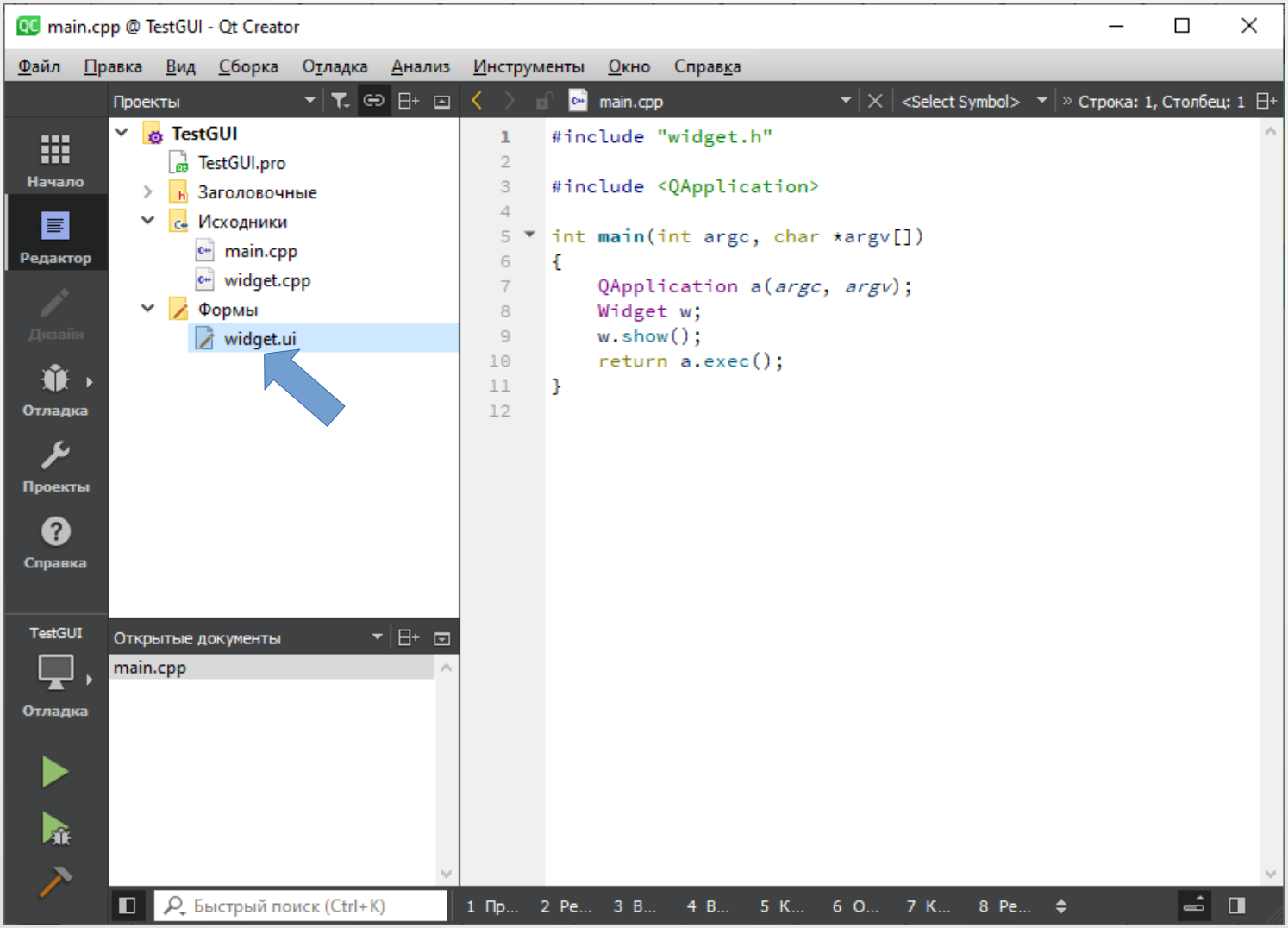
3. Створення проекту з GUI в Qt

Компіляція і запуск цієї програми повинні привести до наступного результату:



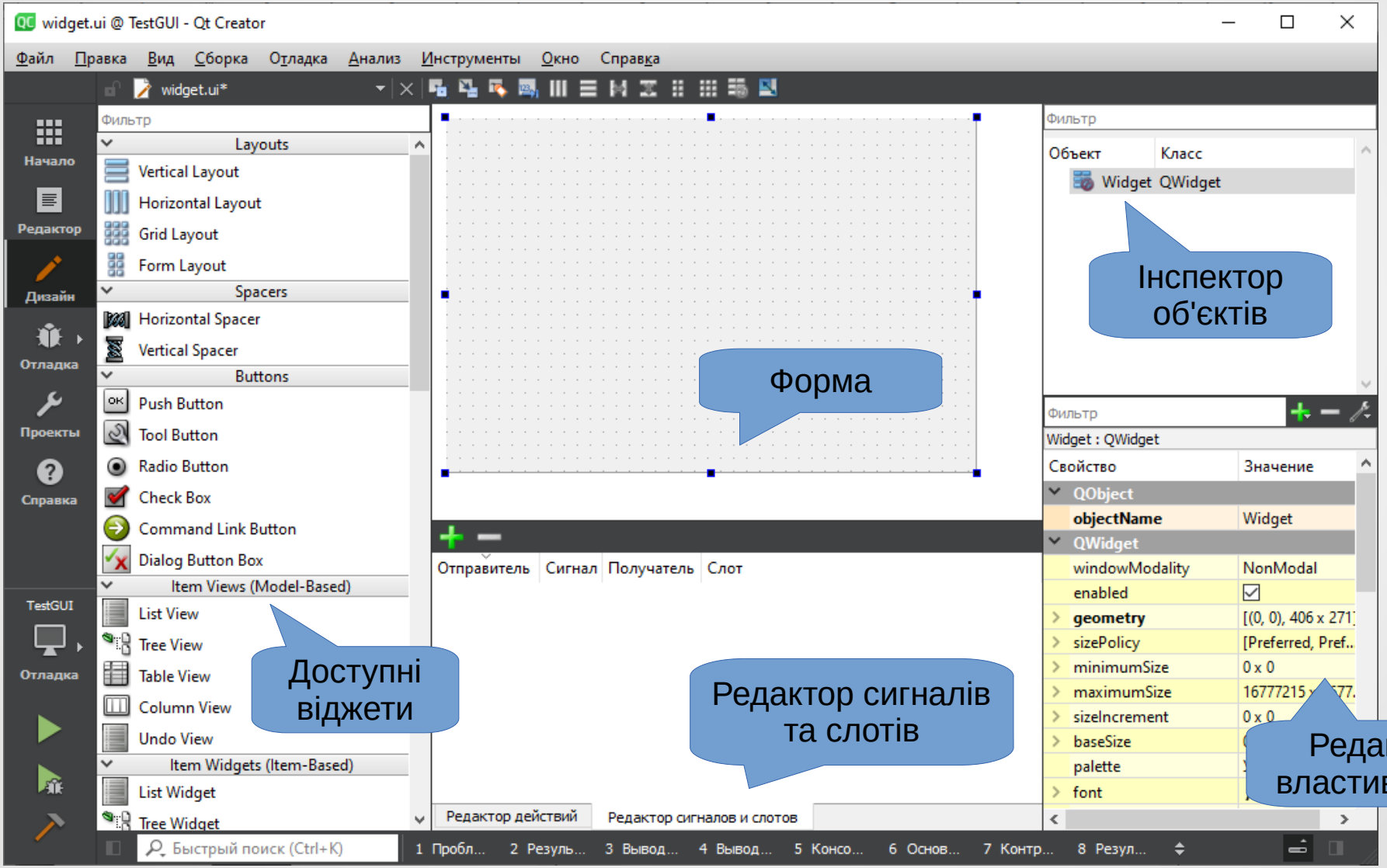
4. Редагування форми в Qt Designer

Для редагування форми її попередньо потрібно відкрити (двічі клікнувши у вікні “Проекты” на відповідному рядку):



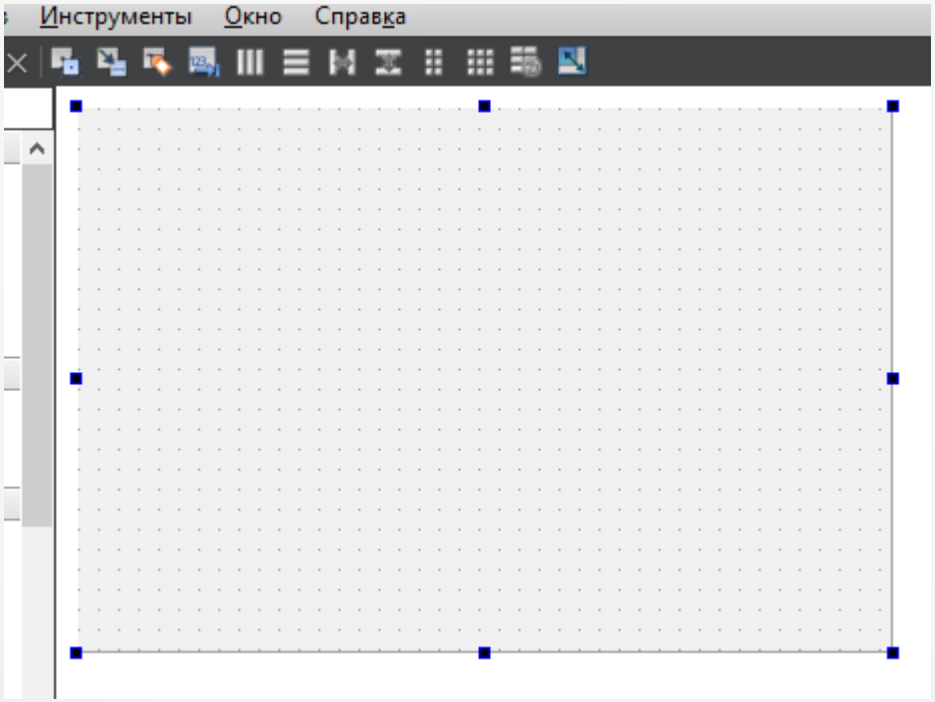
4. Редагування форми в Qt Designer

Після чого відповідна форма буде відкрита у Qt Designer:



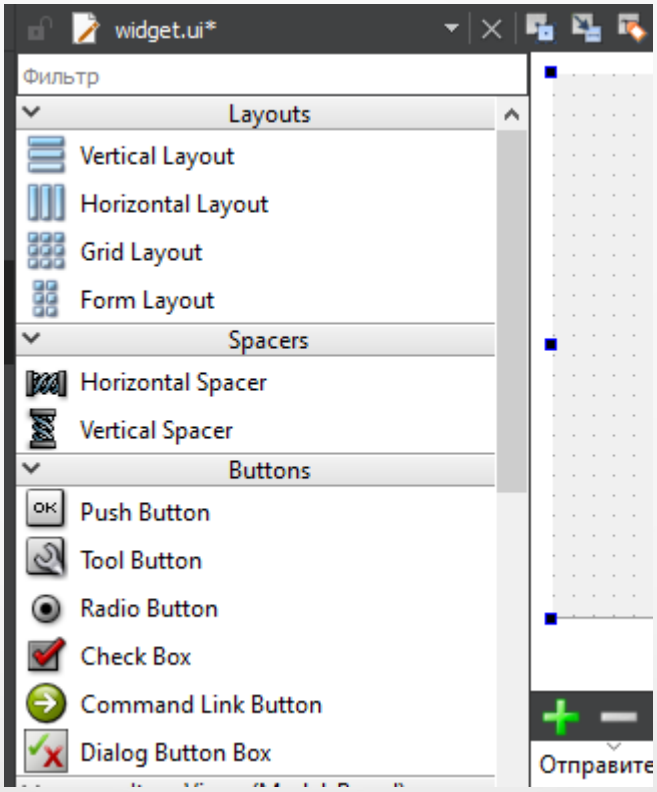
4. Редагування форми в Qt Designer

Форма – центральний елемент дизайну програми. Це прототип майбутнього вікна програми (головного або діалогового).



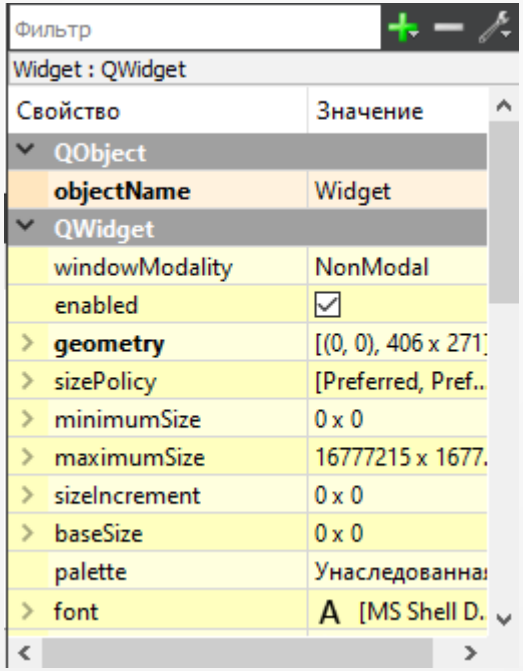
4. Редагування форми в Qt Designer

Вікно **доступних віджетів** (Widget Box) містить список наявних віджетів і об'єктів компонування, згрупованих в окремі категорії. Перетягуванням з цього вікна на форму потрібних користувачеві компонентів здійснюється додавання їх до інтерфейсу програми.



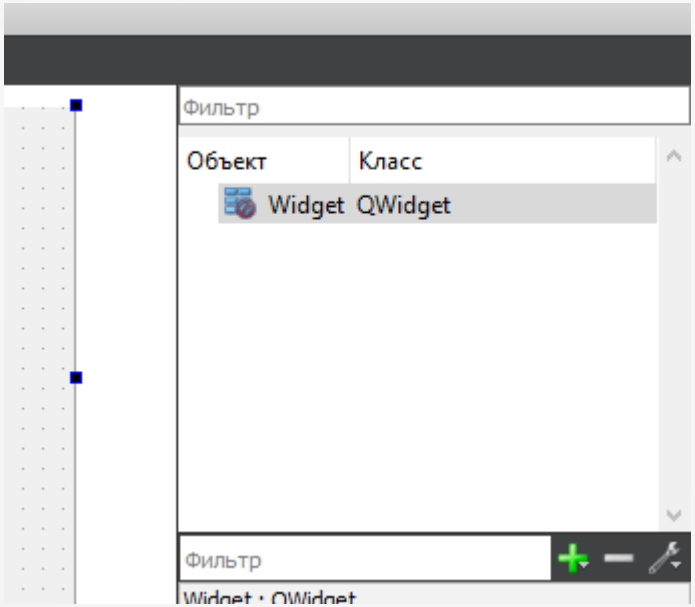
4. Редагування форми в Qt Designer

Редактор властивостей (Property Editor) – містить базові властивості поточного віджету. Наприклад, це можуть бути **колір фону**, **шрифт**, **максимальний** та **мінімальний розмір** віджету тощо.



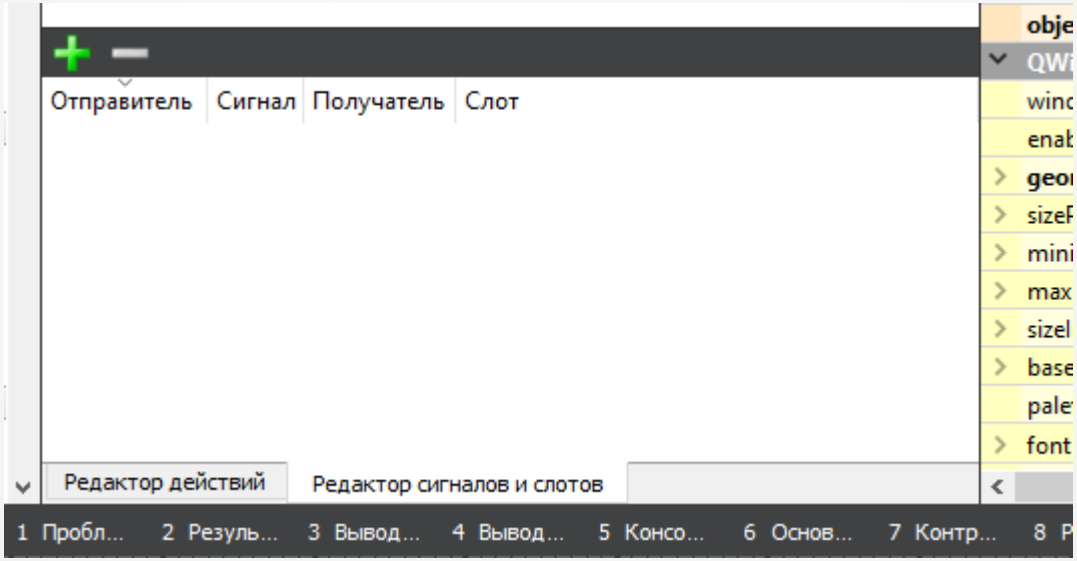
4. Редагування форми в Qt Designer

Інспектор об'єктів (Object Inspector) – містить список використовуваних віджетів. У цьому вікні їх можна вибирати для подальшого редагування.



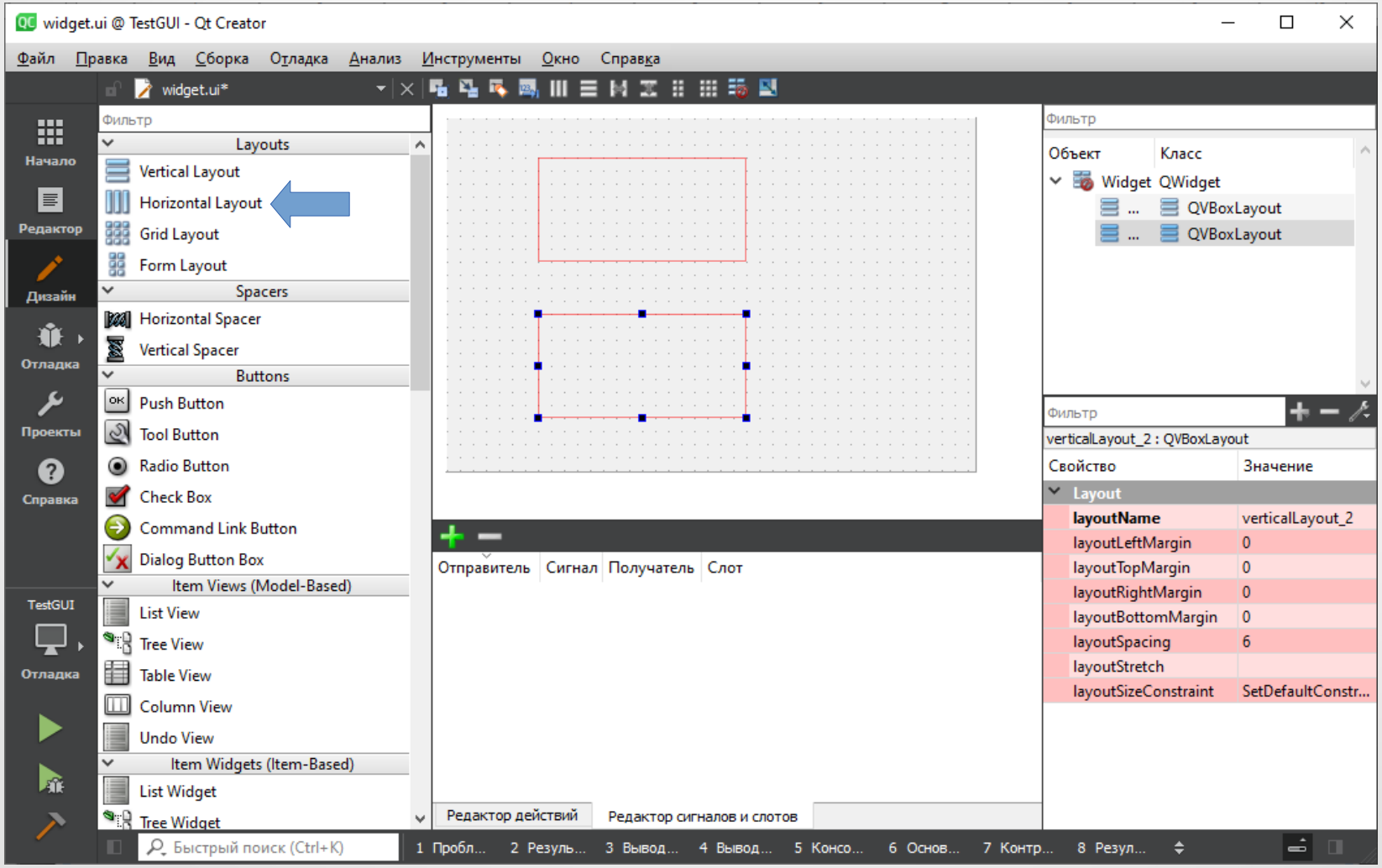
4. Редагування форми в Qt Designer

Редактор сигналів і слотів (Signal / Slot Editor) – вікно, призначене для редагування з’єднань сигналів (подій) зі слотами (функціями-обробникам).



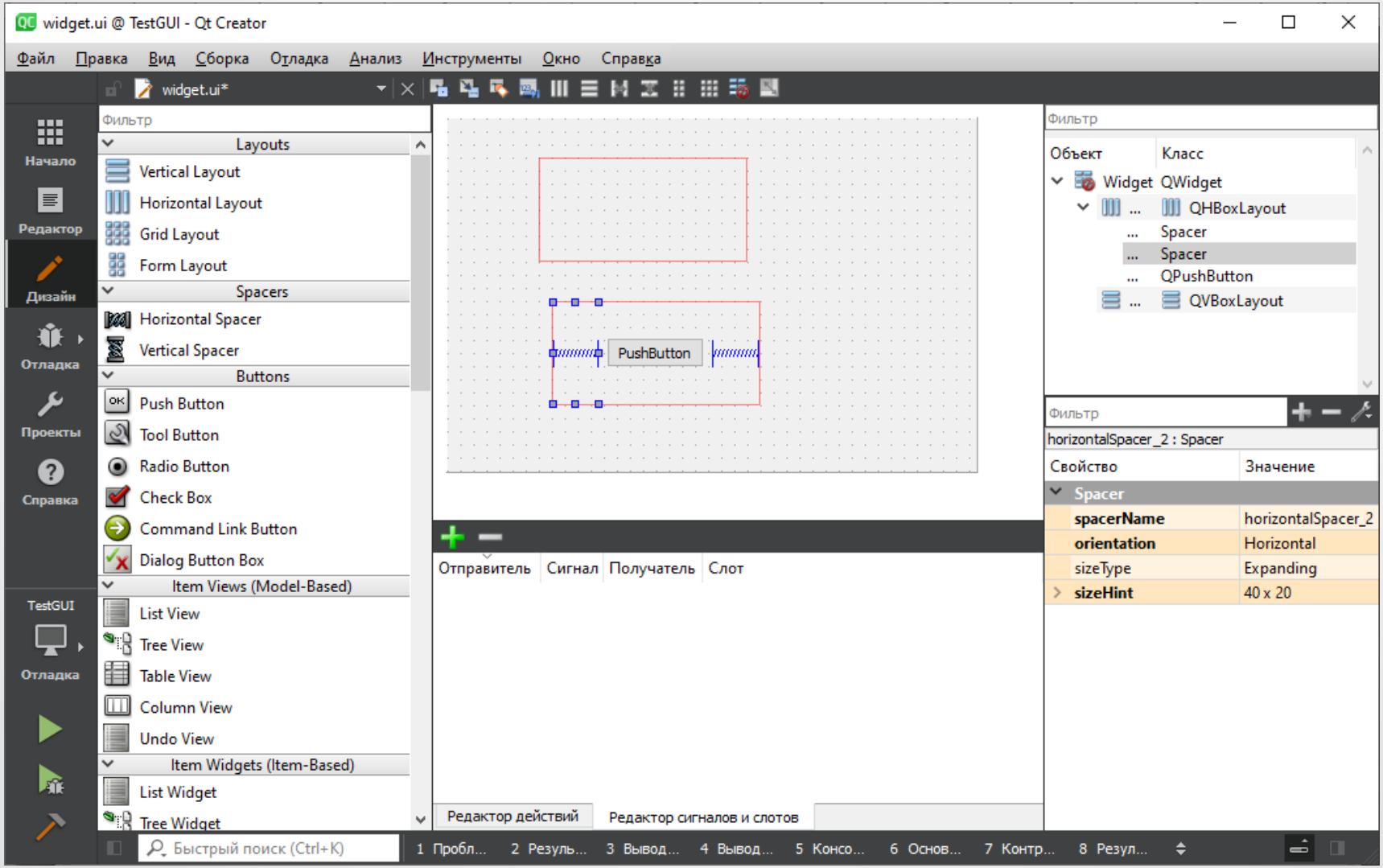
4. Редагування форми в Qt Designer

Для додавання нового віджету на форму його потрібно вибрати у вікні доступних віджетів і перетягнути на форму. Наприклад, перетягнемо на форму два горизонтальних компонувальника:



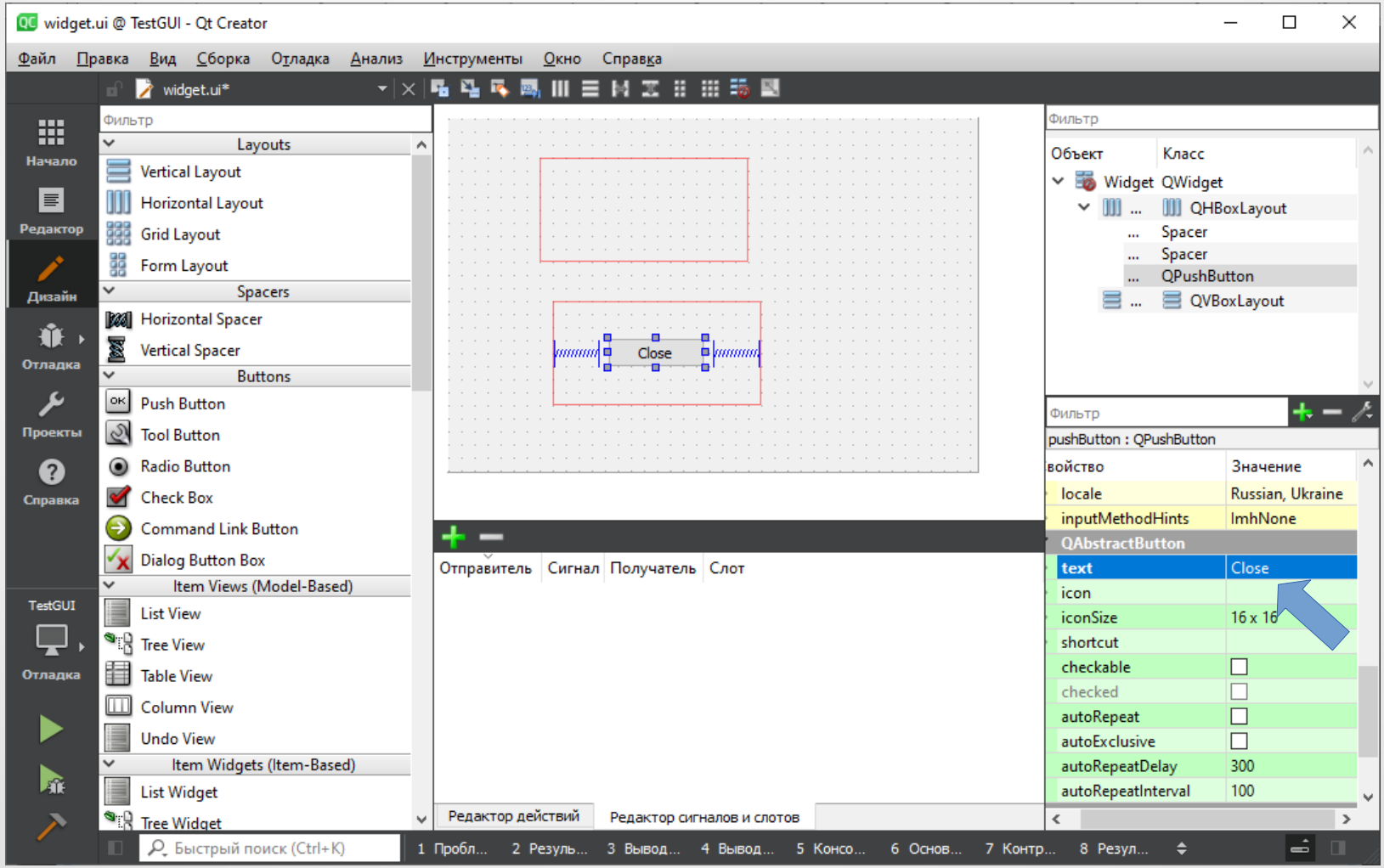
4. Редагування форми в Qt Designer

У нижній компонувальник помістимо кнопку (**Push Button**) і два горизонтальних заповнювача (**Horizontal Spacer**):



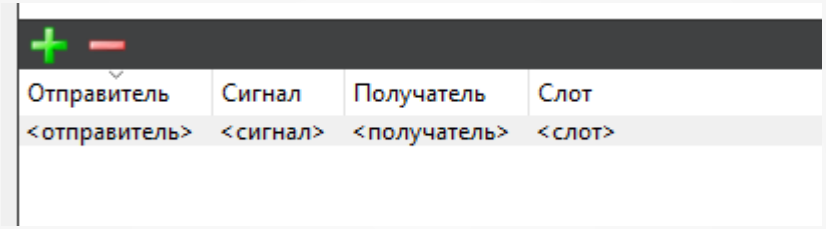
4. Редагування форми в Qt Designer

Клікнувши на кнопку, зробимо її активною. У вікні редактора властивостей знайдемо властивість **Text** (напис на кнопці) і замінимо стандартний текст на “Close”.

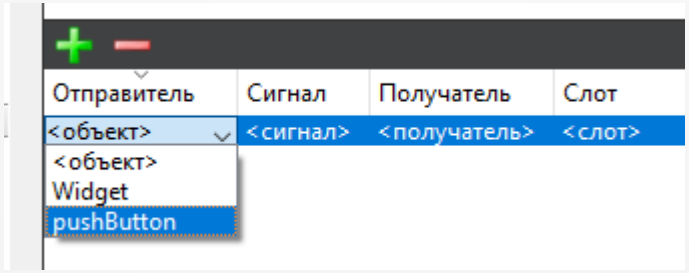


4. Редагування форми в Qt Designer

Налаштуємо цю кнопку таким чином, щоб при її натисканні форма (вікно) закривалася (**закриття головного вікна програми автоматично призводить до її завершення**). Для цього у вікні редактора сигналів і слотів натиснемо на кнопку “+”, після чого буде додано новий рядок налаштувань:

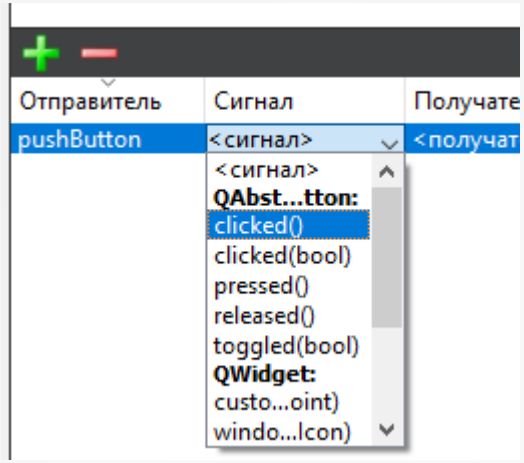


Змінимо значення поля “Отправитель”, вибравши зі списку значення “pushButton”.

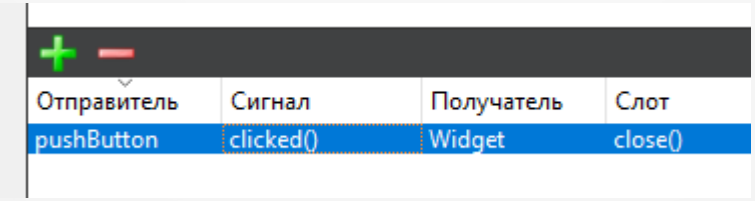


4. Редагування форми в Qt Designer

Далі для поля «Сигнал» оберемо значення ”clicked()”:



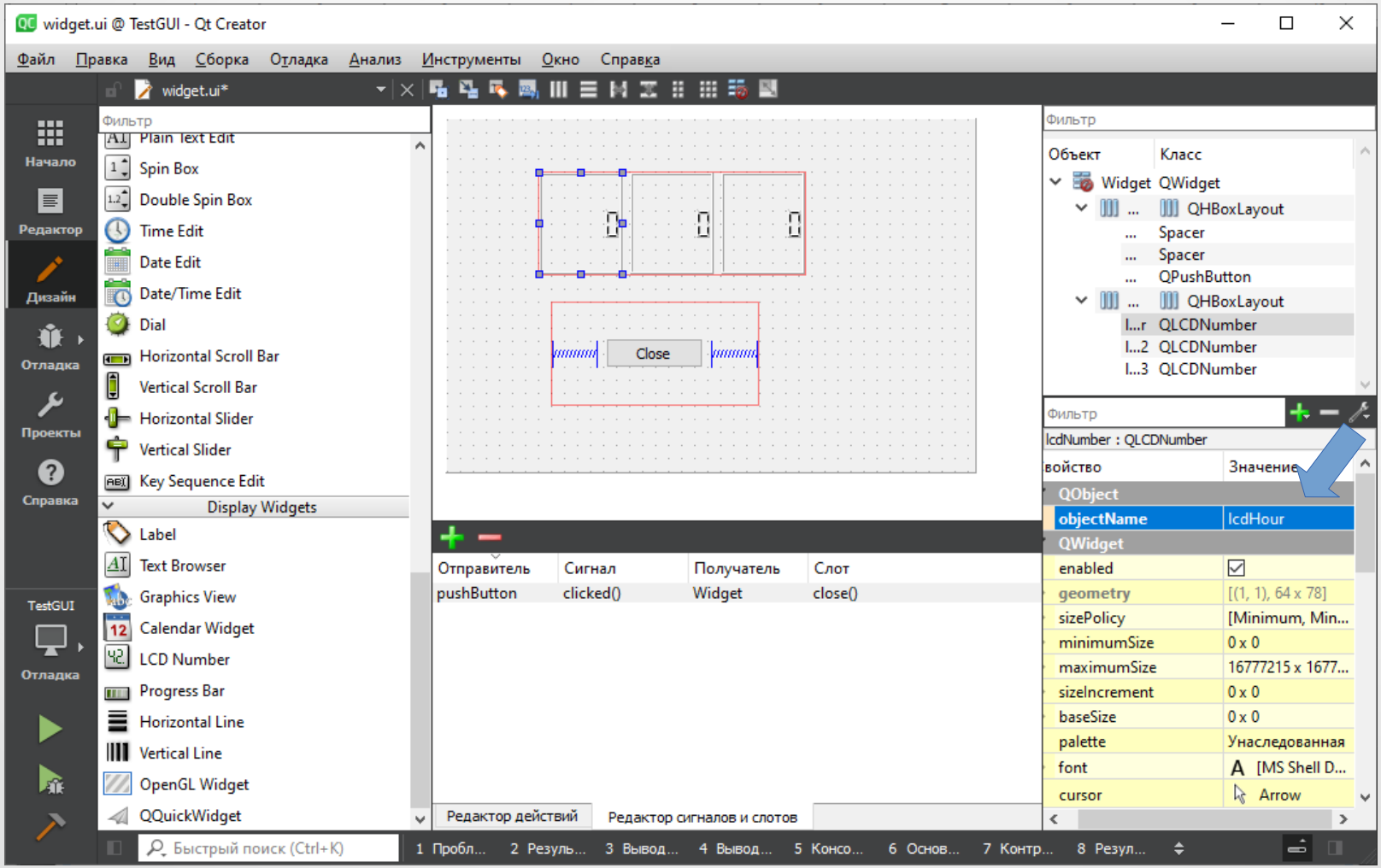
Аналогічним чином налаштуємо поля “Получатель” і “Слот”, обравши для них значення “Widget” і “close()”.



Таким чином, при натисканні на кнопку “pushButton” буде генеруватися сигнал “clicked()”, який буде викликати слот “close()” форми “Widget”, що призведе до закриття вікна.

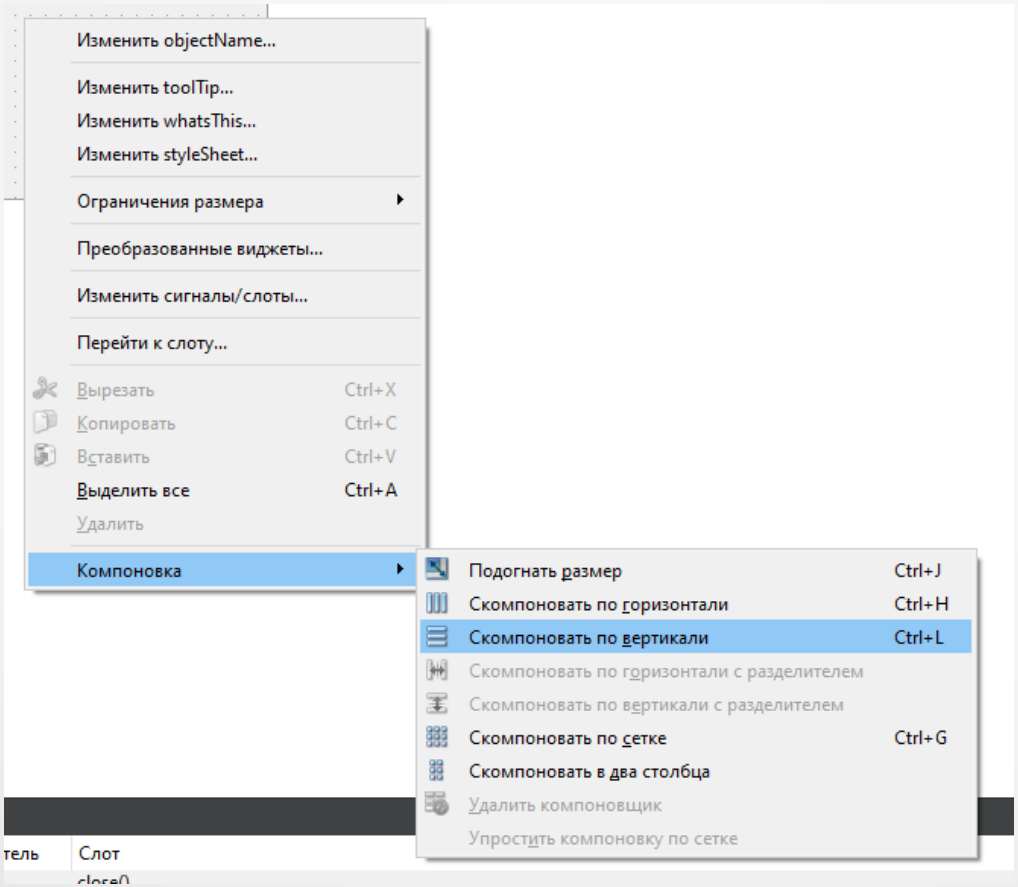
4. Редагування форми в Qt Designer

У верхній вертикальний компонувальник додамо три віджета “LCD Number”. За допомогою властивості “objectName” у вікні редактора властивостей перейменуємо ці об'єкти в “lcdHour”, “lcdMin” і “lcdSec”:



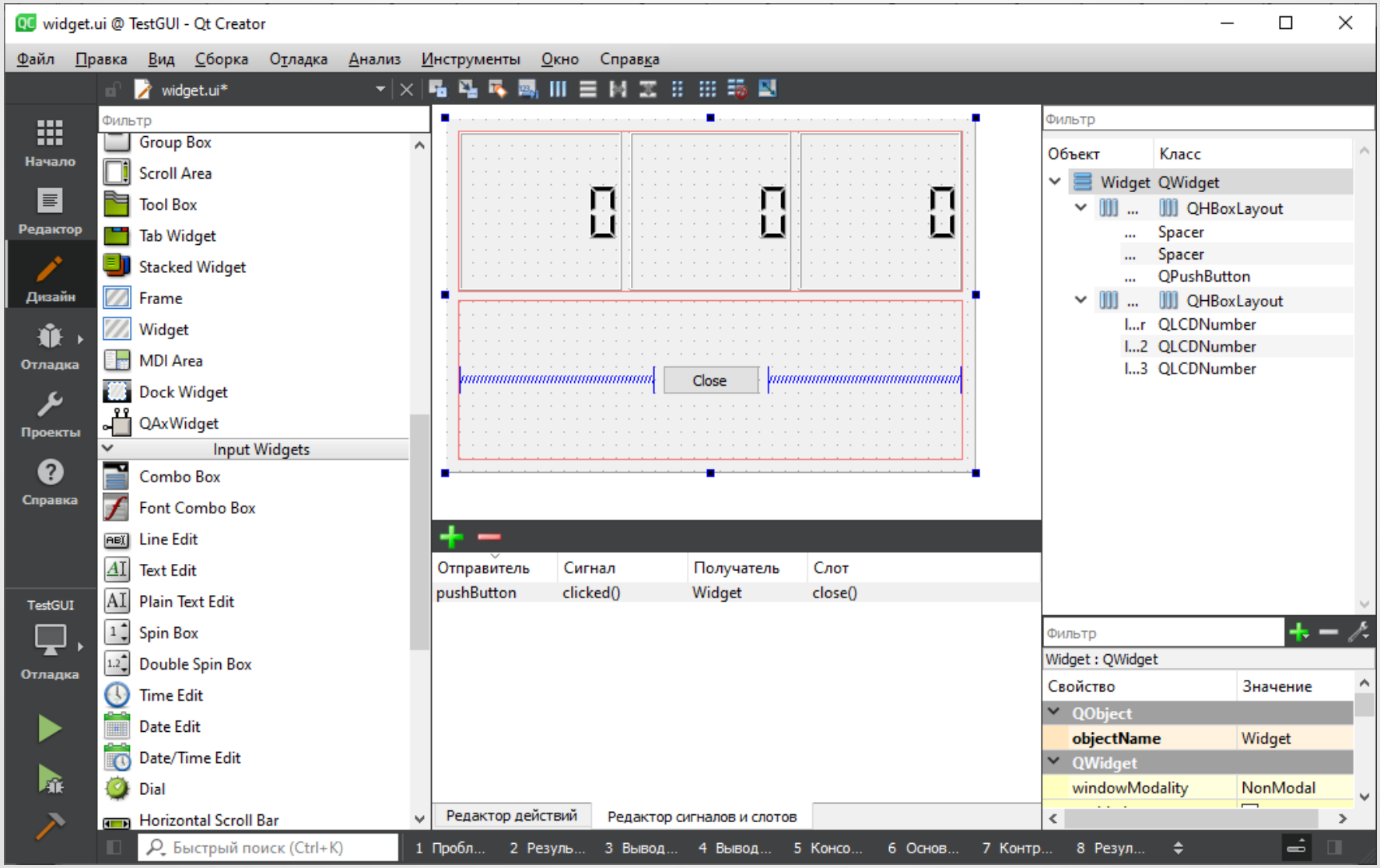
4. Редагування форми в Qt Designer

Кликнемо на формі **правою кнопкою мишки** і в контекстному меню виберемо команду «**Компоновка | Скомпоновать по вертикали**»:



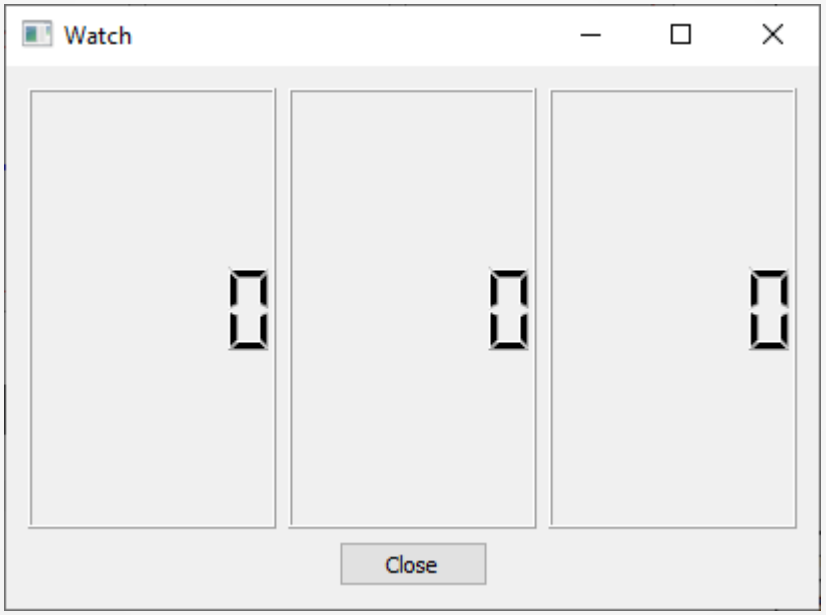
4. Редагування форми в Qt Designer

Після чого компоненти на формі вирівняються таким чином:



4. Редагування форми в Qt Designer

Змінимо заголовок форми (властивість “windowTitle” в редакторі властивостей) на “Watch” і запустимо програму. Отримаємо наступний результат:



4. Редагування форми в Qt Designer

Для завершення розробки годинника з цифровою індикацією часу залишилося реалізувати функціонал зміни значення віджетів, що показують поточне значення годин, хвилин і секунд, відповідними значеннями часу.

Заголовний файл widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>

QT_BEGIN_NAMESPACE
namespace Ui { class Widget; }
QT_END_NAMESPACE

class Widget : public QWidget
{
    Q_OBJECT
private:
    QTimer *timer;
private slots:
    // Зміна індикації часів
    void sloteUpdateWatch(void);

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

private:
    Ui::Widget *ui;
};
#endif // WIDGET_H
```

4. Редагування форми в Qt Designer

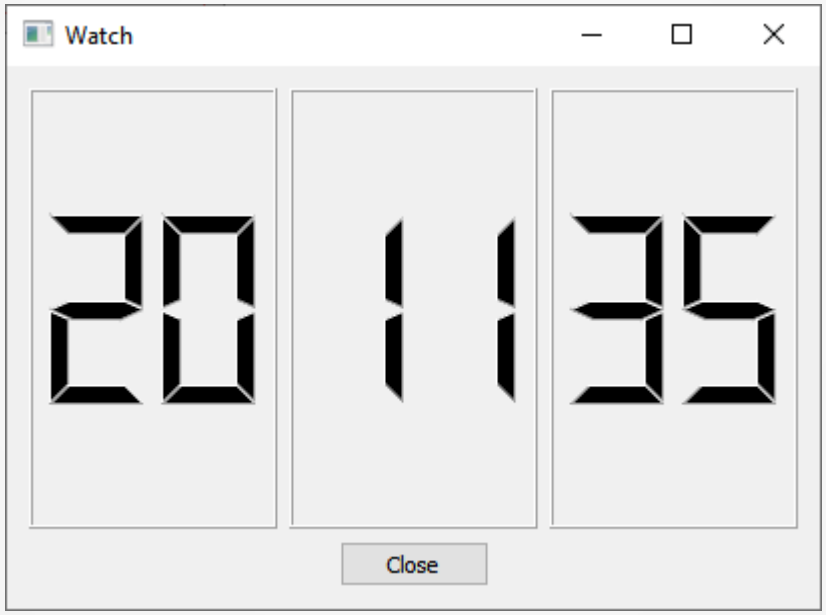
Файл widget.cpp

```
#include <QTime>
#include <QTimer>
#include "widget.h"
#include "ui_widget.h"

Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    timer = new QTimer();
    connect(timer, SIGNAL(timeout()), this,
                SLOT(sloteUpdateWatch()));
    timer->start(1000);
}

Widget::~Widget()
{
    delete timer;
    delete ui;
}
```

```
void Widget::sloteUpdateWatch(void)
{
    QTime time = QTime::currentTime();

    ui->lcdHour->display(time.hour());
    ui->lcdMin->display(time.minute());
    ui->lcdSec->display(time.second());
}
```

4. Редагування форми в Qt Designer

Компіляція та запуск цієї програми приведуть до наступного результату:



Примітка. В наведеному прикладі у об’єктів типу LCDNumber була замінена на “2” властивість “digitCount”, які містить кількість цифр, що відображуються.