

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационной безопасности

Кафедра инфокоммуникационных технологий

**ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ  
ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ  
Часть 1**

**Лабораторная работа 1  
Введение в тип данных Класс.  
Создание приложения с использованием Qt**



**Минск 2022**

## Содержание

Лабораторная работа 1	
Введение в тип данных Класс. Создание Qt GUI в интерактивном режиме .....	3
Введение в тип данных Класс.....	3
Создание GUI в интерактивном режиме.....	7
Задание к лабораторной работе 1.....	11

# Лабораторная работа 1

## Введение в тип данных Класс.

### Создание Qt GUI в интерактивном режиме

Цель работы: Познакомиться со структурой объектно-ориентированной программы. Изучить основы создания GUI (Graphical User Interface) графического пользовательского интерфейса с использованием Qt.

### Введение в тип данных Класс

ООП – методология, основанная на представлении программ в виде совокупности объектов, которые между собой взаимодействуют, причем каждый объект является реализацией конкретного класса. То есть основным элементом ООП является класс.

В основу объектного подхода заложено понятие класса. *Класс* представляет собой описание, схему, модель реально существующего объекта. Класс является типом данных, определяемым пользователем. В результате объявления класса создается новый тип данных, в котором объединены элементы: поля и методы, обрабатывающие эти данные.

*Поле класса* – это элемент класса, описывающий данные.

*Метод класса* – это процедура или функция, включенная в описание класса. Метод класса вызывается конкретным экземпляром класса и привязан к описанию и структуре класса.



Поля и методы класса могут быть закрытыми или открытыми. К закрытым полям и методам класса можно обращаться только внутри класса. Открытые элементы класса доступны за его пределами. Как правило, открытая часть кода обеспечивает управляемое взаимодействие (интерфейс) с закрытыми элементами объекта.

Управляют доступностью элементов класса ключевые слова `private`, `public`, и `protected` (еще их называют: модификаторами доступа, атрибутами доступа, спецификаторами доступа), после которых следует знак двоеточия. В общем виде описание класса выглядит следующим образом.

```
class Имя_класса
{
private:
    /* список полей и методов для использования внутри класса */
public:
    /* список методов, доступных другим функциям и объектам программы */
protected:
    /* список средств, доступных для дружественных классов и наследников */
};
```

По умолчанию все содержимое класса является доступным для чтения и записи только для него самого. Для того чтобы разрешить доступ к данным класса извне, используют ключевое слово `public`. Все что находится после ключевого слова `public`, становятся доступными из всех частей программы.

```
class X
{
    int a, b; // Поля a и b будут иметь private по умолчанию
public:
    double a1, b1; // Поля a1 и b1 будут иметь доступ public
protected:
    char a2, b2; // Поля a2 и b2 будут иметь доступ protected
};
```

Закрытые данные класса размещаются после модификатора доступа `private`. Если отсутствует модификатор `public`, то все функции и переменные по умолчанию являются закрытыми.

Ключевое слово `protected` используется в механизме наследования.

Обычно `private` используют для полей класса, а `public` для его методов. Все действия с закрытыми полями класса реализуются через его методы.

```
class Sale // Объявление класса Sale. Имя класса Sale - становится именем типа, определенного пользователем (пользовательским типом)
{
private: // Все поля класса закрытые
    // Поля класса:
    int IDproduct;
    double price;
    int av_begin;
    int av_end;
public: // Все методы класса открытые
    // Методы класса:
    int total();
    int total_cost();
}; // Завершается объявление класса точкой с запятой
```

Каждый объявленный метод класса должен быть реализован.

*Реализация* – это описание действий метода. Реализация метода может быть определена в самом классе или за пределами класса.

```
class Sale // Объявление класса Sale. Имя класса Sale - становится именем типа, определенного пользователем (пользовательским типом)
{
private: // Все поля класса закрытые
    // Поля класса:
    int IDproduct;
    double price;
    int av_begin;
    int av_end;
public: // Все методы класса открытые
    // Методы класса:
    int total() // Реализация метода определена в классе
    {
        return av_begin - av_end;
    }
    int total_cost(); // Прототип метода
}; // Завершается объявление класса точкой с запятой

int Sale::total_cost() // Реализация метода total_cost вне класса
{
    return (av_begin - av_end) * price;
}
```

Определение реализации метода вне класса содержит имя класса и имя метода, разделенных с помощью оператора, состоящего из двух последовательных двоеточий `::` и называемого оператором разрешения области видимости или оператором привязки. `Sale::totalcost()` означает "Метод `totalcost` принадлежит классу `Sale`".

Класс – это всего лишь логическая абстракция, недоступная для прямого использования в программе. В языке C++ спецификация класса используется для создания объектов, то есть получения доступа к полям и методам класса необходимо создать экземпляр класса, называемый объектом. К одному классу может принадлежать одновременно несколько объектов, каждый из которых имеет уникальное имя. Объект характеризуется физическим существованием и является конкретным экземпляром класса.

Программа работает только с объектами, а классы нужны для задания их внутреннего устройства. Условно говоря, класс – это проектное описание модели (например, автомобиля), а объект – это сам физически существующий автомобиль, построенный по чертежу.

Объект нового типа можно создать так же, как и любую переменную базового типа.

имя\_класса имя\_объекта;

```

class Sale // Объявление класса Sale. Имя класса Sale - становится именем типа, определенного пользователем (пользовательским типом)
{
public: // Все поля и методы класса открыты
    // Поля класса:
    int IDproduct; double price; int av_begin; int av_end;
    // Методы класса:
    int total() // Реализация метода определена в классе
    {
        return av_begin - av_end;
    }
    int total_cost() // Реализация метода определена в классе
    {
        return (av_begin - av_end) * price;
    }
}; // Завершается объявление класса точкой с запятой
int main()
{
    Sale product; // Создание объекта product класса Sale
}

```

В этой строке определяется объект `product` класса (или типа) `Sale`.

В следующем примере создаются два объекта одного класса.

```

class Sale // Объявление класса Sale. Имя класса Sale - становится именем типа, определенного пользователем (пользовательским типом)
{
public: // Все поля и методы класса открыты
    // Поля класса:
    int IDproduct; double price; int av_begin; int av_end;
    // Методы класса:
    int total(){ ... }
    int total_cost(){ ... }
}; // Завершается объявление класса точкой с запятой
int main()
{
    Sale product1, product2; // Создание объектов product1 и product2 класса Sale
}

```

### Прямой способ обращения

После определения объекта класса для доступа к его элементам (как полям, так и методам) используется оператор `(.)`.

имя\_объекта.имя\_поля

имя\_объекта.имя\_метода

В языке C++ нельзя присвоить значение базовому типу данных, оно присваивается только переменной. То же относится и к классам.

```
Sale.IDproduct = 125; // неверно
```

Компилятор выдаст ошибку, поскольку сначала необходимо создать объект класса `Sale`, а потом его полю `IDproduct` присвоить значение 125.

```
Sale product;
product.IDproduct = 125; // верно
```

В программе ниже объявляется класс `Sale`, создается два объекта (`product1` и `product2`) типа `Sale`, полям объектов присваиваются значения и выводятся на экран.

```

#include <iostream>
using namespace std;
class Sale // Объявление класса Sale
{
public: // Открытые элементы класса
    // Поля класса:
    int IDproduct; double price; int av_begin; int av_end;
    // Методы класса:
    int total()
    {
        return av_begin - av_end;
    }
    int total_cost();
};
int Sale::total_cost() // Реализация метода total_cost вне класса
{
    return (av_begin - av_end) * price;
}
int main() {
    Sale product1; // Создание объекта product1 класса Sale
    // Полям объекта product1 присваиваются значения:
    product1.IDproduct = 125;
    product1.price = 120000; product1.av_begin = 20;
    product1.av_end = 3;
    Sale product2; // Создание объекта product2 класса Sale
    // Полям объекта product2 присваиваются значения:
    product2.IDproduct = 130;
    product2.price = 145600; product2.av_begin = 40;
    product2.av_end = 5;
    // Вывод информации о продуктах
    cout << "product1 " << product1.IDproduct << " " <<
    product1.price << " " << product1.av_begin << " " <<
    product1.av_end << " " << product1.total() << " " <<
    product1.total_cost() << endl;
    cout << "product2 " << product2.IDproduct << " " <<
    product2.price << " " << product2.av_begin << " " <<
    product2.av_end << " " << product2.total() << " " <<
    product2.total_cost();
}

```

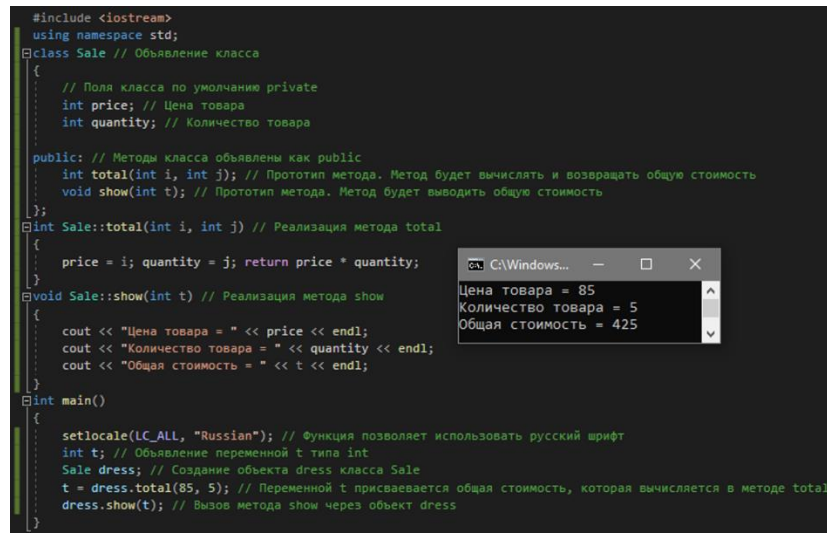
```

Консоль отладки Microso...
product1 125 120000 20 3 17 2040000
product2 130 145600 40 5 35 5096000

```

Как уже было сказано, поля и методы класса могут быть открытыми `public` и закрытыми `private`. По умолчанию все элементы класса являются закрытыми. К закрытым элементам могут обращаться только те методы, которые принадлежат этому классу.

Открытые элементы доступны для всех других методов программы. Согласно общей стратегии использования классов их поля следует оставлять закрытыми. Следовательно, чтобы передавать и возвращать значения закрытых переменных, необходимо создать открытые методы, известные как методы доступа. Применение методов доступа позволяет скрыть от пользователя подробности хранения данных в объектах, предоставляя в то же время методы их использования.



```
#include <iostream>
using namespace std;
class Sale // Объявление класса
{
    // Поля класса по умолчанию private
    int price; // Цена товара
    int quantity; // Количество товара

public: // Методы класса объявлены как public
    int total(int i, int j); // Прототип метода. Метод будет вычислять и возвращать общую стоимость
    void show(int t); // Прототип метода. Метод будет выводить общую стоимость
};

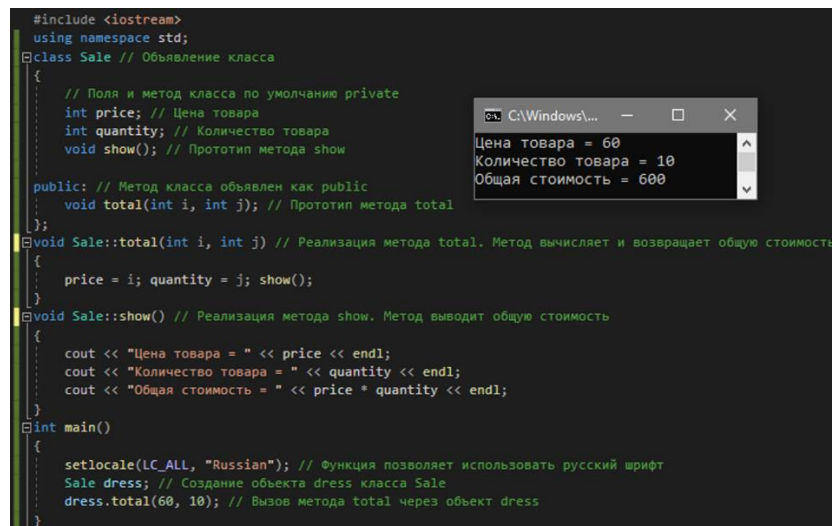
int Sale::total(int i, int j) // Реализация метода total
{
    price = i; quantity = j; return price * quantity;
}

void Sale::show(int t) // Реализация метода show
{
    cout << "Цена товара = " << price << endl;
    cout << "Количество товара = " << quantity << endl;
    cout << "Общая стоимость = " << t << endl;
}

int main()
{
    setlocale(LC_ALL, "Russian"); // Функция позволяет использовать русский шрифт
    int t; // Объявление переменной t типа int
    Sale dress; // Создание объекта dress класса Sale
    t = dress.total(85, 5); // Переменной t присваивается общая стоимость, которая вычисляется в методе total
    dress.show(t); // Вызов метода show через объект dress
}
```

Цена товара = 85  
Количество товара = 5  
Общая стоимость = 425

В примере ниже метод `show` класса `Sale` уже закрытый и вызывается в методе `total`.



```
#include <iostream>
using namespace std;
class Sale // Объявление класса
{
    // Поля и метод класса по умолчанию private
    int price; // Цена товара
    int quantity; // Количество товара
    void show(); // Прототип метода show

public: // Метод класса объявлен как public
    void total(int i, int j); // Прототип метода total
};

void Sale::total(int i, int j) // Реализация метода total. Метод вычисляет и возвращает общую стоимость
{
    price = i; quantity = j; show();
}

void Sale::show() // Реализация метода show. Метод выводит общую стоимость
{
    cout << "Цена товара = " << price << endl;
    cout << "Количество товара = " << quantity << endl;
    cout << "Общая стоимость = " << price * quantity << endl;
}

int main()
{
    setlocale(LC_ALL, "Russian"); // Функция позволяет использовать русский шрифт
    Sale dress; // Создание объекта dress класса Sale
    dress.total(60, 10); // Вызов метода total через объект dress
}
```

Цена товара = 60  
Количество товара = 10  
Общая стоимость = 600

### Косвенный способ обращения

Для объектов могут создаваться указатели, как и для переменных базовых типов. Значением указателя является адрес первой ячейки области памяти, выделенной под объект.

имя\_класса \*имя\_указателя;

Чтобы получить адрес памяти, по которому записан объект, перед именем объекта указывают оператор `&`.

имя\_указателя = &имя\_объекта;

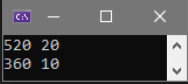
При обращении к конкретному элементу объекта посредством указателя на объект нужно использовать оператор-стрелку (`->`) (ввести знак "минус", а затем знак "больше"). После указателя через стрелку указывается имя поля или метода.

имя\_указателя->имя\_поля;

имя\_указателя->имя\_метода;

В примере ниже демонстрируется создание и использование указателей на объекты.

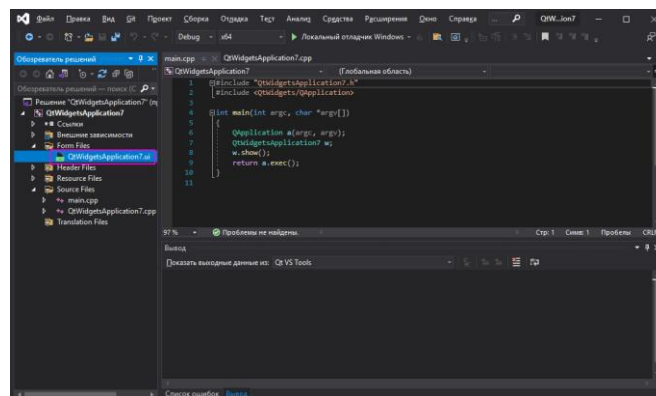
```
#include<iostream>
using namespace std;
class Sale // Объявление класса
{
    // Поля и методы класса public
public:
    int price; // Цена товара
    int quantity; // Количество товара
    void total(int i, int j); // Прототип метода total
};
void Sale::total(int i, int j) // Реализация метода total
{
    price = i; quantity = j;
}
int main() {
    Sale product; // Создание объекта product класса Sale
    Sale *p_product; // Создание указателя p_product на объект класса Sale
    product.total(520, 20); // Вызов метода total через объект product
    cout << product.price << " " << product.quantity << endl;
    p_product = &product; // Присвоение указателю p_product адреса объекта product
    p_product->total(360, 10); // Вызов метода total через указатель p_product
    cout << p_product->price << " " << p_product->quantity << endl;
}
```



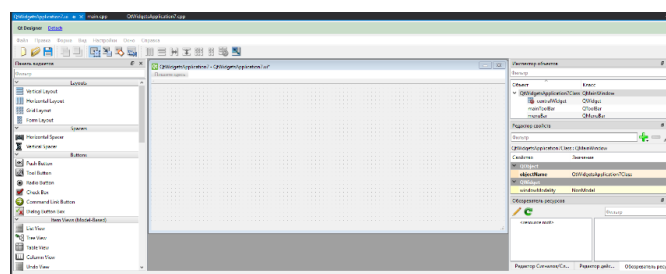
## Создание GUI в интерактивном режиме

### IDE Visual Studio

Для создания GUI в интерактивном режиме, необходимо в окне **Обозреватель решений** выбрать файл `.ui`.



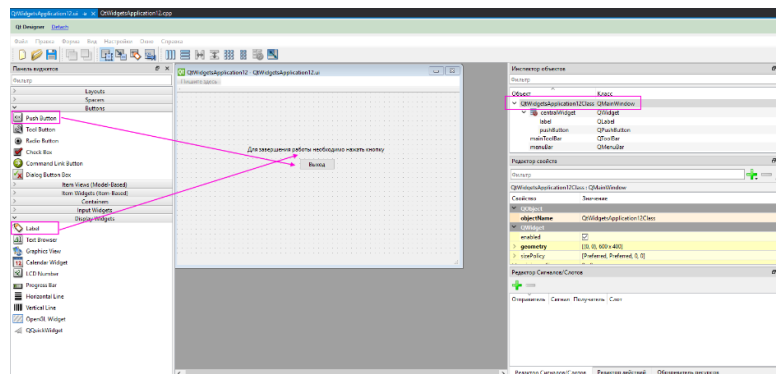
Откроется редактор **Qt Designer**. **Qt Designer** – инструмент для проектирования и создания графических пользовательских интерфейсов (GUI) из компонентов Qt.



В результате создается заготовка окна `QtWidgetsApplication...`, где `QtWidgetsApplication...` – имя объекта класса `QMainWindow` по умолчанию.



Далее с **Панели виджетов** перетаскиваем на заготовку окна виджеты **Push Button** и **Label** меняем надписи (используя двойной клик на элементе) и размер (используя движение мыши).

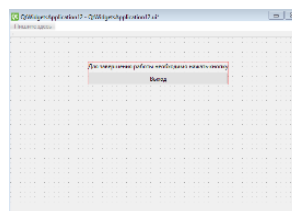


Сохраняем форму **Файл→Сохранить**.

Также, как и при создании интерфейса вручную для управления размещением виджетов на форме Qt Designer использует компоновщики. Для добавления компоновщика виджеты должны быть выделены, что можно сделать, кликая мышкой по виджетам при нажатой клавише **Ctrl**.

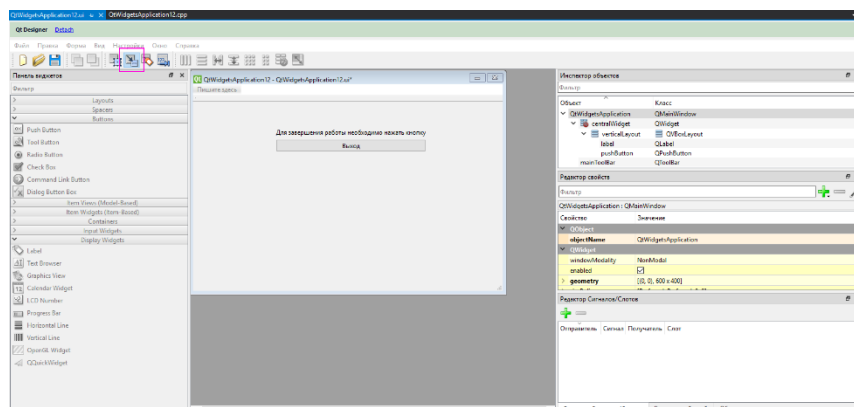
Добавление компоновщиков и связывание с ними виджетов осуществляется выбором пунктов меню **Форма**:

- Скомпоновать по горизонтали;
- Скомпоновать по вертикали;
- Скомпоновать по горизонтали с разделителем;
- Скомпоновать по вертикали с разделителем;
- Скомпоновать по сетке;
- Скомпоновать в два столбца.



Кроме визуального конструирования вида окон, Qt Designer позволяет связать заранее предусмотренные сигналы виджетов с такой же заранее предусмотренной реакцией других виджетов на эти сигналы.

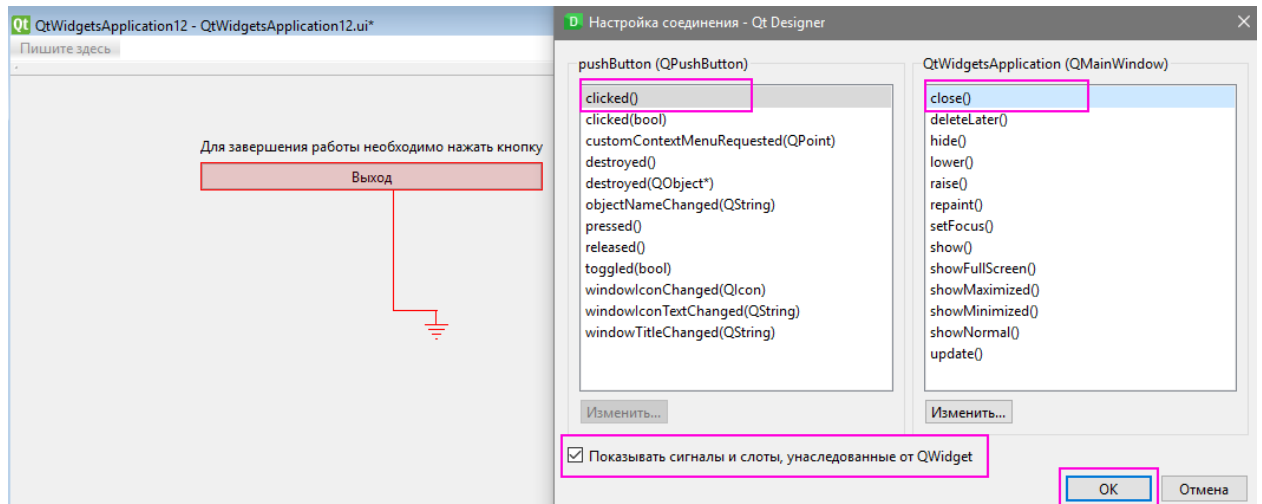
Для этого необходимо переключиться из режима **Изменение виджетов** в режим **Изменение сигналов/слотов**.



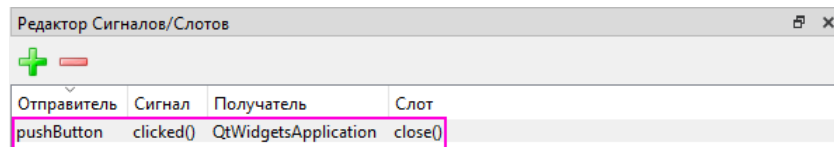
Выбор виджетов, между сигналом и слотом которых устанавливается связь, выполняется визуально: кликаем левой клавишей мышки по виджету кнопки и, не отпуская левую клавишу, переносим указатель мышки на свободное место



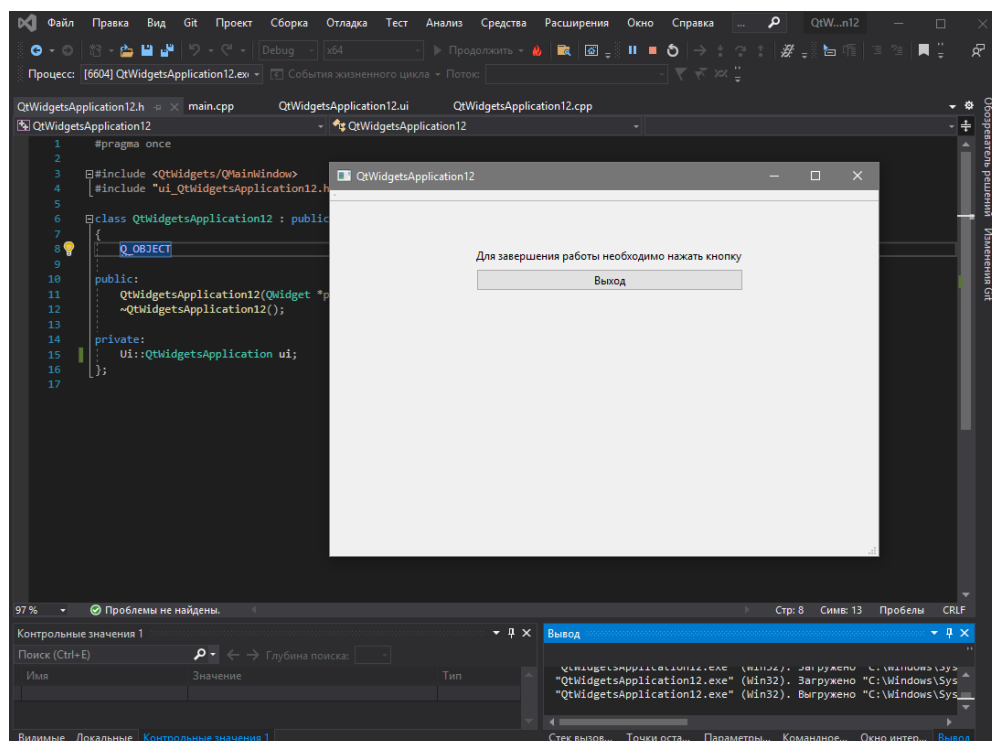
окна. Выполняем настройки соединения сигнала `clicked()` кнопки **Выход** со слотом `close()` через окно настроек.



В окне **Редактор Сигналов/Слотов** (окно расположено справа внизу редактора Qt Designer) отражается результат соединения сигнала и слота.

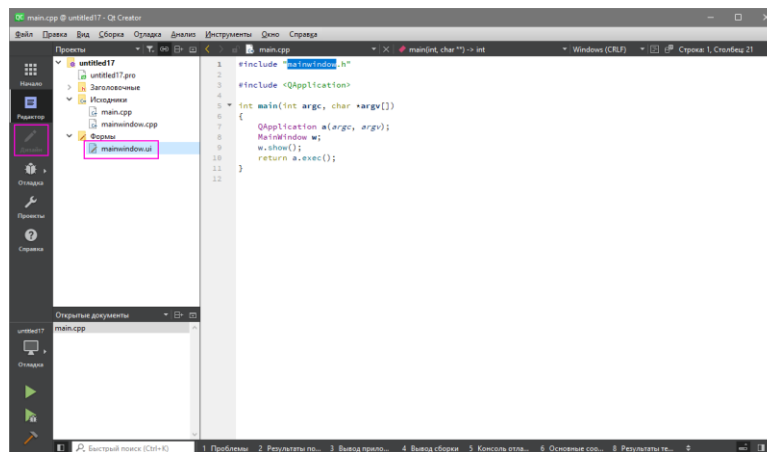


Сохраняем результаты работы в редакторе Qt Designer (Файл→Сохранить), возвращаемся в редактор Visual Studio и запускаем приложение. Нажимаем кнопку **Выход**, окно приложения закрывается.

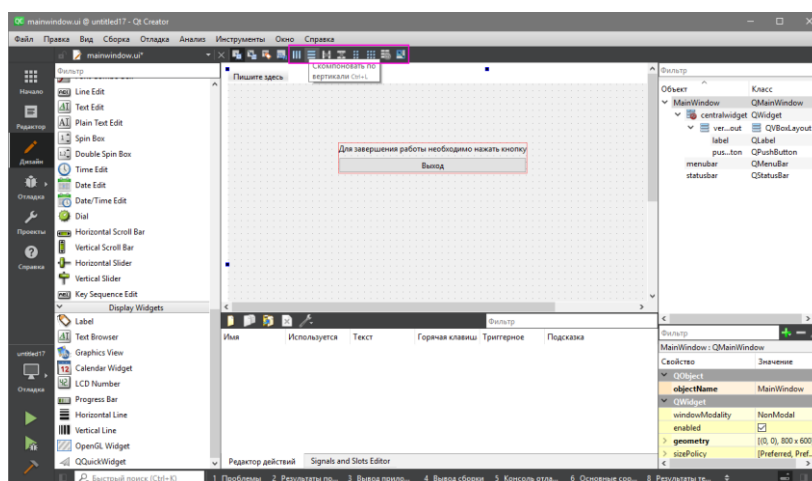


## IDE Qt Creator

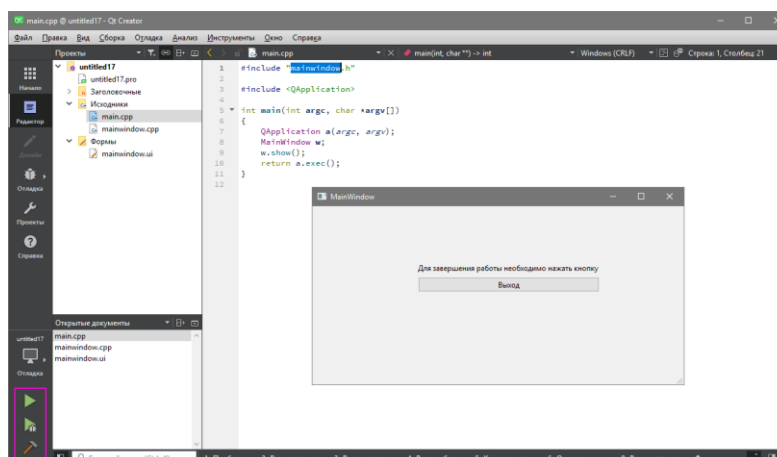
Для создания формы необходимо выбрать файл `.ui` либо переключится в режим **Дизайн**.



Добавление компоновщиков и связывание с ними виджетов осуществляется выбором пунктов на панели сверху.



Все остальные действия идентичны, как и в предыдущем подразделе. После создания кнопки и установки сигналов и слотов выполняем сборку и запуск.



Нажимаем кнопку **Выход**, окно приложения закрывается.

## Задание к лабораторной работе 1

Для выполнения лабораторной работы необходимо установить и настроить Visual Studio и инструменты Qt. Задание к лабораторной работе 1 состоит из нескольких задач. Задачи выполняются по вариантам (например, Вариант 1 – номера по списку в группе: 1, 9, 17, 26; Вариант 2 – номера по списку в группе: 2, 10, 18, 27 и т.д.).

### Вариант 1



#### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса). Каждая строчка кода должна содержать комментарий.

Вы свободный фрилансер и на днях к вам обратилась компания "BestGameFromEarth", они просят вас помочь им. Вам необходимо создать класс `Player`, в котором будет храниться `hp`, `magic_power`, `stamina` (все поля типа `int`) и написать метод, который будет выводить крутость игрока. Крутость игрока рассчитывается по формуле

$$(\text{hp} \times \text{magic\_power}) / \text{stamina} + (\text{hp} \times \text{stamina}) / \text{magic\_power} - (\text{stamina} \times \text{magic\_power}) / \text{hp}.$$

Не забывайте про деление на ноль. В этом случае должно быть выведено сообщение об ошибке с указанием на поле, которое было равно нулю).

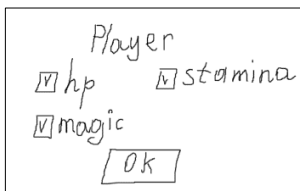
Вы успешно выполнили заказ и уже думаете куда поехать отдыхать на заработанные деньги, да вот только заказчик вспомнил, что персонаж еще имеет физический и магический урон и просит срочно добавить эти поля и вывести значения этого урона у персонажа с бонусом, получаемым от `hp`, `magic_power` и `stamina`. Заварив еще одну чашечку кофе, вы приступили к работе. Вам необходимо добавить в ранее написанный класс еще два поля (тип `double`): `magic_damage`, `physical_damage` и добавить еще один метод, который будет выводить магический и физический урон персонажа по формулам

$$\begin{aligned} &(\text{magic\_damage} \times \text{magic\_power}) - \text{stamina}, \\ &(\text{physical\_damage} \times \text{stamina} + \text{hp}) - \text{magic\_power}. \end{aligned}$$

#### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Находясь в "чемоданном настроении", вы получили письмо от заказчика с просьбой в срочном порядке разработать GUI интерфейс с одной функциональной кнопкой. К письму был прикреплен быстрый набросок от руки. "Дурость" – подумали вы и пошли выполнять просьбу. Вам необходимо создать GUI приложение с текстовой меткой, тремя чекбоксами и функциональной кнопкой `Ok`.



## Вариант 2



### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса). Каждая строка кода должна содержать комментарий.

Добро пожаловать в Научно-практический центр исследований "Explore all you want". Вы младший научный сотрудник и сегодня вы изучаете плесень. Да, младшим научным сотрудником еще мало чего доверяют, но вы докажете свой профессионализм. Цель вашего исследования определить количество плесени через определенный промежуток времени. Вам известны: количество начальной плесени, ее прирост в день и количество дней, которое она будет размножаться с такой скоростью. Для того, чтобы не пачкать бумагу на расчеты и тем самым сохранить природу, создайте программу с классом `Mildew` в котором будут три поля (типа `int`): `speed`, `day`, `percentage` и напишите метод, который вернет количество плесени через промежуток времени, введенный в поле `day`.

Вы успешно написали программу и вычислив предполагаемое количество плесени через `n` дней, со спокойной душой ушли в отпуск, но вернувшись, вы обнаружили, что количество плесени не совпадает с тем значением, которое прогнозировала программа. Ошибку в программе вы исключили сразу (вы же лично ее писали! В ней не может быть ошибок!). Выпив кофейку, вы поняли в чем дело. Каждый день помимо прибавления плесени, шло и отмирание старой. Вычислив количество плесени, которое отмирает в день, вы готовы внести правки в программу. Необходимо добавить поле (тип `int`): `death_speed`, и в методе, который должен выводить количество плесени через промежуток времени необходимо учесть эту переменную.

### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

За проявленный энтузиазм вы получили похвалу от руководства. Программу захотели использовать и ваши коллеги. Средний возраст сотрудников центра 66 лет, поэтому необходим дружественный интерфейс. Вам необходимо создать GUI приложение с текстовой меткой и переключателями, скомпонованными по сетке: `Плесень` (переключатели: `прибавление плесени`, `отмирание плесени`) и функциональной кнопкой `Выход`.

### Вариант 3



#### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса). Каждая строка кода должна содержать комментарий.

Сегодня вы заступили на должность главврача больницы "Healthy like a bull". И первое что бы вы хотели узнать – это процент выздоровевших больных и симулянтов от общего числа пациентов. Чтобы все отделения могли быстро выдать вам правильный результат, не запутавшись в этих процентах и делениях, создайте класс с тремя полями (типа `int`): `convalescents`, `sick`, `simulators` и метод, который вернет количество выздоровевших, больных и симулянтов в % от общего числа пациентов. На всякий случай, общее количество пациентов

$(convalescents + sick + simulators)$ .

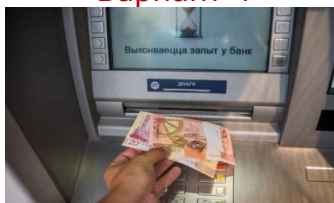
Вы получили статистику пациентов, а что насчет персонала? Сколько людей на вас вообще работает? Это необходимо срочно выяснить, да еще и в процентном отношении. Необходимо добавить три поля (тип `int`): `doctor`, `nurse`, `medical_orderly` и метод, который выведет количество персонала каждого типа в процентах от общего количества сотрудников.

#### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Вы захотели, чтобы вся информация о пациентах и сотрудниках больницы предоставлялась в более понятном виде. Для этого вам необходимо создать GUI приложение с двумя текстовыми метками и переключателями, скомпонованными по вертикали: Пациенты (переключатели: выздоровевшие, больные, симулянты), Персонал (переключатели: доктора, медсестры, санитары), и функциональной кнопкой Выход.

## Вариант 4



### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса). Каждая строчка кода должна содержать комментарий.

В банке "We are the most honest" где вы проходите стажировку хотят поставить новый автомат для оплаты каких-то услуг. Механизм автомата элементарен. Клиент вставляет свою карточку, выбирает услугу, которую он хочет оплатить и если у него хватает денег на карточке, то услуга оплачивается. Но есть нюанс. За свои услуги автомат списывает комиссию с карточки в размере 5 % от цены услуги (необходимо внимательно читать правила пользования автоматом, которые приклеены позади него). Вам необходимо создать класс `Machine` в котором будет три поля (два поля типа `int`, одно поле типа `double`): `money`, `price`, `residue`. Пользователь вводит значения для полей `money` и `price` (поле `residue` изначально равно нулю), поле `residue` заполняется при выполнении метода, который должен высчитать остаток после оплаты услуги (не забываем про списание автоматом 5 %). Если оплата прошла успешно, метод выводит об этом сообщение, в случае нехватки денег, выводит сообщение о неудаче.

Вашему начальнику очень нравится работа автомата для оплаты каких-то услуг, теперь он хочет добавить туда новые опции. Теперь, пользователь должен иметь возможность добавить себе денег на карту взяв их в долг у банка, а потом в течении месяца вернуть. Вот только, начальник попросил оставить функцию изъятия 5 % автоматом за выполнение операции. То есть, если пользователь возьмет 100 рублей у банка в долг, то автомат снимет комиссию 5 % и перечислит пользователю 95 рублей, а вернуть пользователь должен будет 100 рублей. Со словами "Это не баг, это фича" вы пошли выполнять просьбу начальства. Необходимо добавить поле (тип `int`): `debt` и метод, который увеличит значение `money` на `debt` с вычетом 5 % комиссии и выведет старое значение `money`, значение `money` после пополнения и сумму долга перед банком.

### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Дошло дело и до разработки пользовательского интерфейса автомата. "Начнем с малого" – сказал начальник и поставил задачу реализовать запрос "Желаете ли Вы продолжить?". Вам необходимо создать GUI приложение с текстовым полем и диалоговыми кнопками `Да/Нет`. Нажатие на кнопку `Нет` должно привести к закрытию приложения.

## Вариант 5



### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса).

С сегодняшнего дня в ваше распоряжение переходит полицейский участок "Liberty". И первое что вы хотели бы узнать – это процент простых, средних и сложных вызовов от общего их числа. Чтобы сотрудники быстрее смогли дать вам эту важную информацию, создайте класс `Call` в котором будет храниться три поля (тип `int`): `easy`, `middling`, `hard` и метод, который выведет количество простых, средних и тяжелых вызовов в % от общего их числа. На всякий случай, общее количество вызовов

(`easy+middling+hard`).

Вы получили статистику вызовов, а что насчет персонала? Сколько людей на вас вообще работает? Это необходимо срочно выяснить, да еще и в процентном соотношении. Необходимо добавить три поля (тип `int`): `policeman`, `special_forces`, `detective` и метод, который выведет количество персонала каждого типа в процентах от общего количества сотрудников.

### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Вы захотели, чтобы вся информация о сотрудниках участка предоставлялась в более понятном виде. Для этого вам необходимо создать GUI приложение с текстовой меткой и чекбоксами: `Сотрудники` (чекбоксы: `полицейские`, `спецназовцы`, `детективы`), и функциональной кнопкой `Выход`. Компоновку элементов осуществить по вертикали.



## Вариант 6



### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса).

Сегодня вы заступаете на должность мэра славного города "Racun City". Вам срочно необходимо узнать уровень развития своего города. Так как город очень быстро растет, чтобы не производить сложные расчеты каждый раз после появления чего-то нового, вы записались на курсы "ООП за один день" и теперь можете создать программу, которая сделает это за вас. Вам необходимо создать класс `City` в котором будут три поля: `residential_buildings`, `industrial_building`, `unique_buildings` и метод, который будет выводить уровень развития города на основе этих полей. Воспользуйтесь формулой для расчета уровня развития города

$$((\text{residential\_buildings} \times \text{industrial\_building}) - \text{residential\_buildings}) \times \text{unique\_buildings} / \text{industrial\_building}.$$

Не забывайте про деление на ноль. В этом случае должно быть выведено сообщение, что уровень развития города равен нулю.

Посмотрев на карту города, вы обнаружили, что у вас есть еще парки и магазины, а это тоже необходимо учитывать при расчете уровня развития города. Необходимо добавить два поля (тип `int`): `park`, `store`. Новая формула для расчета уровня развития города

$$((\text{residential\_buildings} \times \text{industrial\_building} \times \text{park} + \text{store}) - \text{residential\_buildings}) \times \\ \times (\text{unique\_buildings} + \text{park} + \text{store}) / \text{industrial\_building}.$$

### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Вы захотели, чтобы вся информация о городе предоставлялась в более понятном виде. Для этого вам необходимо создать GUI приложение с текстовыми метками, скомпонованными по вертикали: Постройки (переключатели: жилые дома, промышленные здания), Торговые и развлекательные объекты (переключатели: парки, магазины), и функциональной кнопкой Выход.

## Вариант 7



### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса).

Открыта новая фабрика по производству носков "Warm feet" и вы ее новый директор. Уже прошел месяц со старта фабрики и вам очень хочется узнать сколько же вы заработали. Напишите класс `Socks` в котором будет три поля (тип `int`): `produced`, `sales`, `price` и метод, который выведет прибыль от продаж и количество непроданного товара.

Прибыль от продажи носков вас приятно удивила, и вы решили расширяться. Долго думая, что еще можно производить, вы пришли к мысли, что гольфы – это просто длинные носки и ваша фабрика способна их тоже производить. Необходимо добавить три поля (тип `int`): `stockings_produced`, `stockings_sales`, `stockings_price` и метод, который выведет прибыль от продаж гольф и их непроданное количество.

### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Вы захотели, чтобы вся информация о фабрике предоставлялась в более понятном виде. Для этого вам необходимо создать GUI приложение с текстовыми метками и переключателями, скомпонованными по вертикали: Носки (переключатели: количество произведенных, количество проданных), Гольфы (переключатели: количество произведенных, количество проданных), и функциональной кнопкой Выход.

## Вариант 8



### Задача 1

Необходимо написать объектно-ориентированную программу, использующую прямой и косвенный способы обращения к методам. Пользовательский класс должен содержать необходимые поля, метод установки их начальных значений, метод просмотра их текущего состояния, метод, решающий поставленную задачу (реализацию этого метода определить вне класса).

Вы устроились администратором в отель "Flophouse". Это очень большой отель, вы даже не знаете сколько в нем этажей, вам лишь известно, что на последнем этаже комнат может быть меньше, чем на всех остальных этажах, так же вам известно общее количество комнат в отеле и количество комнат на каждом этаже (кроме последнего). Вам необходимо узнать, сколько же этажей в отеле, которым вы руководите (лифт сломан, а подниматься и считать вам лень). Вам необходимо написать класс `Hotel` в котором будет два поля (тип `int`): `number_of_rooms_on_the_floor`, `number_of_rooms` и метод, который выведет количество комнат на последнем этаже и количество этажей в вашем отеле.

Вы смогли посчитать количество этажей в вашем отеле. Теперь необходимо посчитать сколько прибыли приносит ваш отель. Узнав цену за одну комнату и количество жильцов, вы приступили к работе. Необходимо добавить два поля (тип `int`): `price`, `tenant` и метод, который выведете прибыль отеля.

### Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Вы захотели, чтобы вся информация об отеле предоставлялась в более понятном виде. Для этого вам необходимо создать GUI приложение с текстовыми метками переключателями, скомпонованными по сетке: Этаж (переключатели: 10, 12, 13), Количество комнат в номере (переключатели: 1, 2, 3), и функциональной кнопкой Ok.