

**Московский приборостроительный техникум**

**Практическая работа №2**

**Дисциплина:** ОП.04 «Основы алгоритмизации и программирования»

**Тема:** Разработка модуля регистрации с помощью Windows Forms на языке C#

**Специальность:** 09.02.07 Информационные системы и программирование

**Квалификация:** Разработчик веб и мультимедийных приложений

Разработал:  
преподаватель Колмыкова В.А..

Рассмотрено и одобрено на заседании ЦМК  
Профессиональных модулей 09.02.07-ВД

Протокол от \_\_\_\_\_ г. № \_\_\_\_\_

Председатель ЦМК \_\_\_\_\_ / М.С. Прищеп /

## **Цель работы**

Изучение и практическое применение элементов управления и контейнеров для разработки приложений типа Windows Forms на языке C#. Изучение способов реализации добавления данных в БД посредством приложения.

## **Задание**

Руководствуясь перечисленными ниже требованиями, выполнить действия по созданию проекта оконного приложения Windows Forms в интегрированной среде разработки MS Visual Studio. Проект будет содержать в себе множество модулей, выполняемые постепенно, в соответствии с заданием.

В качестве основы данной практической работы следует использовать результат выполнения практической работы №1 «Разработка модуля авторизации с помощью Windows Forms на языке C#» с сохранением указанных в ней требований к решению задачи. Предполагается, что указанная практическая работа №1 выполнена правильно и в полном объеме.

Данная практическая работа подразумевает создание формы регистрации пользователя в системе и разработка её полного функционала.

Квалификация: Разработчик веб и мультимедийных приложений

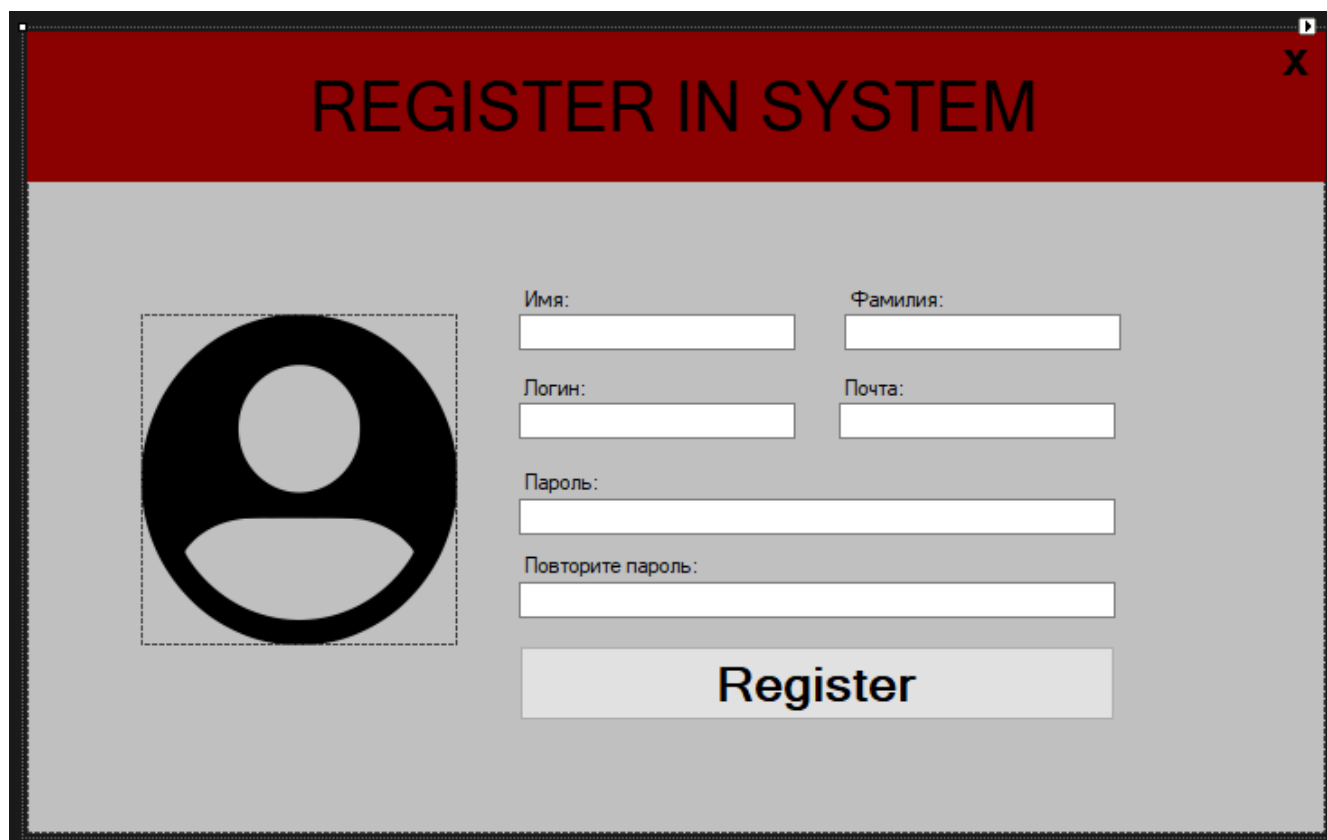
## Этапы выполнения:

### Этап 1. Создание дополнительной формы

На рисунке 1 представлена форма, которую необходимо повторить.

*«Стилизация формы, показанная на рисунке 1, не является обязательной частью. Задача состоит в том, чтобы у каждого была стилизована форма в соответствии со своей предметной областью. Это значит, что необходимо подбирать цвета и картинки, которые больше подойдут к той или иной предметной области».*

Также необходимо обратить внимание, что в зависимости от предметной области количество полей для ввода и их значений могут отличаться, от примера.



The image shows a web form titled "REGISTER IN SYSTEM" in a red header bar. The form is set against a light gray background. On the left, there is a circular profile picture placeholder with a dashed border. To the right of the placeholder are several input fields: "Имя:" (Name) and "Фамилия:" (Surname) are side-by-side; "Логин:" (Login) and "Почта:" (Email) are side-by-side; "Пароль:" (Password) is a single wide field; and "Повторите пароль:" (Repeat password) is another single wide field. At the bottom of the form is a large gray button labeled "Register". The entire form is enclosed in a window-like frame with a close button (X) in the top right corner.

Рисунок 1 – Форма регистрации

Квалификация: Разработчик веб и мультимедийных приложений

После создания формы, необходимо добавить на форму авторизации элемент label, который будет отвечать за переход на недавно созданную форму (рисунок 2).

Созданному элементу необходимо дать соответствующее название и кликнуть на него. Внутри нужно прописать создание переменной RF с функцией формы и вызвать ее (рисунок 3).



Рисунок 2 – Добавления элемента на форму авторизации

```
Ссылка: 1
private void openRegisterForm_Click(object sender, EventArgs e)
{
    RegForm RF = new RegForm();
    RF.ShowDialog();
}
```

Рисунок 3 – Код для открытия формы регистрации

Квалификация: Разработчик веб и мультимедийных приложений

## Этап 2. Создание нового класса

Необходимо в папке проекта создать новый класс `CheckTextClass`. Он будет отвечать за удаление ненужных отступов в полях ввода (рисунок 4).

1. Первым делом создается строковая переменная `TextBox` с модификатором `protected` (доступ к защищенному элементу может быть получен из соответствующего класса, а также экземплярами производных классов);
2. Далее создается публичная функция созданного класса с параметром строковой переменной `_TextBox`. Внутри, переменной `TextBox`, которая является текущей, присваивается та самая строковая переменная;
3. Создается публичная функция называемая `CheckSimbol()`, внутри которой текущей переменной `TextBox` присваивается выполнение замены (а именно: замена значений с пробелом на пустые);
4. Создается публичная строковая функция `returnTextBox()`, которая отвечает за возвращение переменной текущего поля ввода.

```
namespace bd21klientka
{
    Ссылка: 1
    internal class CheckTextClass
    {
        protected string TextBox;

        Ссылка: 0
        public CheckTextClass(string _TextBox)
        {
            this.TextBox = _TextBox;
        }

        Ссылка: 0
        public void CheckSimbol()
        {
            this.TextBox = this.TextBox.Replace(" ", "");
        }

        Ссылка: 0
        public string returnTextBox()
        {
            return this.TextBox;
        }
    }
}
```

Рисунок 4 – Код класса

Квалификация: Разработчик веб и мультимедийных приложений

Теперь можно перейти к форме регистрации и открыть событие клика на кнопку. На рисунке 5 представлен код:

1. Первым делом создается переменная nameBoxClass с недавно созданным классом, которой передается значение из поля ввода имени пользователя;
2. Далее выполняется функция CheckSimbol();
3. И последнее – это создание переменной nameUser и присваивание ей переменной nameBoxClass с функцией возвращение текущего поля ввода.

```
Ссылка: 1
private void registerButton_Click(object sender, EventArgs e)
{
    CheckTextClass nameBoxClass = new CheckTextClass(userFirstname.Text);
    nameBoxClass.CheckSimbol();
    string nameUser = nameBoxClass.returnTextBox();
}
```

Рисунок 5 – Применение класса к значению в поле ввода

Код, который был представлен выше, необходимо повторить на каждое поле ввода, которое есть на форме регистрации (кроме пароля).

```
Ссылка: 1
private void registerButton_Click(object sender, EventArgs e)
{
    CheckTextClass nameBoxClass = new CheckTextClass(userFirstname.Text);
    nameBoxClass.CheckSimbol();
    string nameUser = nameBoxClass.returnTextBox();

    CheckTextClass lastnameBoxClass = new CheckTextClass(userLastname.Text);
    lastnameBoxClass.CheckSimbol();
    string lastnameUser = lastnameBoxClass.returnTextBox();

    CheckTextClass emailBoxClass = new CheckTextClass(userEmail.Text);
    emailBoxClass.CheckSimbol();
    string emailUser = emailBoxClass.returnTextBox();

    CheckTextClass loginBoxClass = new CheckTextClass(userLogin.Text);
    loginBoxClass.CheckSimbol();
    string loginUser = loginBoxClass.returnTextBox();
}
```

Рисунок 6 – Добавление класса к остальным полям ввода

Квалификация: Разработчик веб и мультимедийных приложений

### Этап 3. Реализация регистрации пользователя

Теперь можно перейти к реализации регистрации пользователя:

1. Сначала нужно добавить условие: если пароль не пустой, тогда условие: если пароль равен подтвержденному паролю, тогда..., иначе вывод сообщения, что пароли не совпадают (рисунок 7)

```
CheckTextClass loginBoxClass = new CheckTextClass(userLogin.Text);
loginBoxClass.CheckSimbol();
string loginUser = loginBoxClass.returnTextBox();

if(userPassword.Text != "")
{
    if(userPassword.Text == userConfirmPassword.Text)
    {
    }
    else
    {
        MessageBox.Show("Пароли разные, повторите попытку снова");
    }
}
```

Рисунок 7 – Условие на проверку пароля и подтвержденного пароля

2. Внутри второго условия необходимо создать переменную db и cmdReg. Первая переменная содержит в себе функцию выполнения класса DB, а вторая запрос к БД. Запрос выглядит следующим образом: добавить в таблицу пользователей (перечисление атрибутов) значения (переменные, прошедшие проверку на пустые символы или переменные, содержащие текст из поля ввода). Запрос можно рассмотреть на рисунке 8

```
if(userPassword.Text == userConfirmPassword.Text)
{
    DB db = new DB();
    MySqlCommand cmdReg = new MySqlCommand($"INSERT INTO users (firstname_user, lastname_user, email_user, login_user, password_user)
    VALUES ({nameUser}, {lastnameUser}, {emailUser}, {loginUser}, {userConfirmPassword.Text});", db.GetConnection());
}
```

Рисунок 8 – Запрос к БД для добавления пользователя

3. Далее нужно открыть подключение и прописать условие: если переменная, содержащая в себе запрос с параметром, отвечающий за возврат значения, равна единице, тогда отправляется сообщение о том, что пользователь был создан и происходит закрытие формы, иначе вывод ошибки. После условия необходимо закрыть подключение.

Квалификация: Разработчик веб и мультимедийных приложений

Если говорить простыми словами, то если при выполнении запроса его данные передались в БД и выполнились, тогда создается пользователь

```
if(userPassword.Text == userConfirmPassword.Text)
{
    DB db = new DB();
    MySqlCommand cmdReg = new MySqlCommand($"INSERT INTO users (firstname_user, lastname_user, email_user, login_user, password_user)
    VALUES ('{nameUser}', '{lastnameUser}', '{emailUser}', '{loginUser}', '{userConfirmPassword.Text}');", db.GetConnection());

    db.openConnection();
    if (cmdReg.ExecuteNonQuery() == 1)
    {
        MessageBox.Show("Пользователь создан");
        Close();
    }
    else
    {
        MessageBox.Show("Ошибка: Не удалось создать пользователя, повторите позже");
    }
    db.closeConnection();
}
else
{
    MessageBox.Show("Пароли разные, повторите попытку снова");
}
```

Рисунок 9 – Выполнение запроса



Квалификация: Разработчик веб и мультимедийных приложений

#### Этап 4. Дополнение регистрации пользователя проверками

Регистрация пользователя реализована. Теперь можно приступить к созданию различных проверок для корректности работы приложения:

1. Первая проверка будет на уникальность логина, т.е. нужно сделать так, чтобы при попытке создания пользователя выходила ошибка, если введенный логин уже существует. Можно рассмотреть код, показанный на рисунке 10:

Создается переменная, содержащая класс подключения к БД, потом создается строковая переменная, в которой содержится введенное значение логина из поля ввода. Дальнейший код является уже знакомым, он использовался ранее. Запрос, который необходим в данном случае выглядит так: выбрать все из таблицы пользователя, где атрибут логина равен переменной, содержащей логин, введенный через поле ввода.

Дальше идет выполнение запроса и условие: если в таблице количество строк больше 0, тогда возвращение значения ложь, иначе возвращение значения истина.

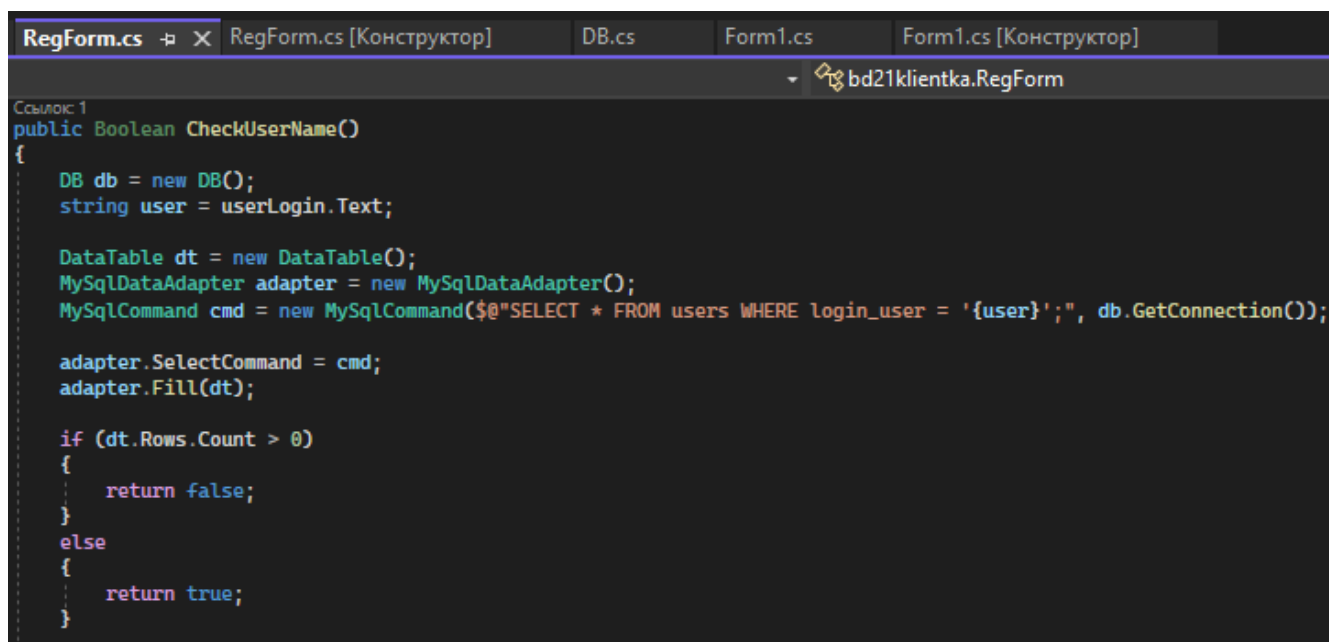


Рисунок 10 – Проверка на уникальность логина

2. Функция проверки логина готова, теперь ее необходимо использовать внутри кода регистрации. Добавить выполнение данной функции необходимо после проверки паролей и разместить её внутри условия: если функция выполняется,

Квалификация: Разработчик веб и мультимедийных приложений

тогда выполняется код регистрации, иначе вывод сообщения о том, что пользователь с таким логином уже существует.

```
if(userPassword.Text != "")
{
    if(userPassword.Text == userConfirmPassword.Text)
    {
        if (CheckUserName())
        {
            DB db = new DB();
            MySqlCommand cmdReg = new MySqlCommand($"INSERT INTO users (firstname_user, lastname_user, email_user, login_user, password_user)
            VALUES ('{nameUser}', '{lastnameUser}', '{emailUser}', '{loginUser}', '{userConfirmPassword.Text}');", db.GetConnection());

            db.openConnection();
            if (cmdReg.ExecuteNonQuery() == 1)
            {
                MessageBox.Show("Пользователь создан");
                Close();
            }
            else
            {
                MessageBox.Show("Ошибка: Не удалось создать пользователя, повторите позже");
            }
            db.closeConnection();
        }
        else
        {
            MessageBox.Show("Пользователь с таким логином уже существует!");
        }
    }
    else
    {
        MessageBox.Show("Пароли разные, повторите попытку снова");
    }
}
```

Рисунок 11 – Добавление первой проверки в код регистрации

3. Следующая проверка – проверка уникальности почты и правильно введенных символов. Код для реализации данной проверки практически идентичен с прошлым кодом.

Различия: в запросе идет проверка не логина, а почты; добавлены 2 переменные, которые находят наличие символов («@» и «.») в поле ввода почты; в условие добавлены проверки на наличие нужных символов.

```
Ссылка 1
public Boolean CheckUserEmail()
{
    DB db = new DB();

    DataTable dt2 = new DataTable();
    MySqlDataAdapter adapter2 = new MySqlDataAdapter();
    MySqlCommand cmd2 = new MySqlCommand($"SELECT * FROM users WHERE email_user = '{userEmail.Text}';", db.GetConnection());

    adapter2.SelectCommand = cmd2;
    adapter2.Fill(dt2);

    int atpos = userEmail.Text.IndexOf("@");
    int dotpos = userEmail.Text.LastIndexOf(".");

    if (dt2.Rows.Count > 0 || atpos < 1 || dotpos < 1)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Рисунок 12 – Проверка на уникальность почты

Квалификация: Разработчик веб и мультимедийных приложений

4. Созданную проверку также необходимо добавить к коду регистрации, как показано на рисунке 13.

```
if(userPassword.Text != "")
{
    if(userPassword.Text == userConfirmPassword.Text)
    {
        if (CheckUserName())
        {
            if (CheckUserEmail())
            {
                DB db = new DB();
                MySqlCommand cmdReg = new MySqlCommand($"INSERT INTO users (firstname_user, lastname_user, email_user, login_user, password_user)
                VALUES ({nameUser}, '{lastnameUser}', '{emailUser}', '{loginUser}', '{userConfirmPassword.Text}';", db.GetConnection());

                db.openConnection();
                if (cmdReg.ExecuteNonQuery() == 1)
                {
                    MessageBox.Show("Пользователь создан");
                    Close();
                }
                else
                {
                    MessageBox.Show("Ошибка: Не удалось создать пользователя, повторите позже");
                }
            }
            db.closeConnection();
        }
        else
        {
            MessageBox.Show("Пользователь с такой почтой уже существует или вы ввели почту неправильно!");
        }
    }
    else
    {
        MessageBox.Show("Пользователь с таким логином уже существует!");
    }
}
else
{
    MessageBox.Show("Пароли разные, повторите попытку снова");
}
```

Рисунок 13 – Добавление второй проверки в код регистрации

5. Последняя проверка, которая необходима – это проверка на пустые поля. На рисунке 14 представлен код данной проверки: создаются строковые переменные, которые содержат в себе значения, написанные в поля ввода, после, далее прописывается условие, если эти переменные пустые, тогда возвращается значение – ложь, иначе – истина.

```
Ссылка 1
public Boolean CheckTextBox()
{
    string nameBox = userFirstname.Text;
    string lastNameBox = userLastname.Text;
    string emailBox = userEmail.Text;
    string loginBox = userLogin.Text;

    if (nameBox == "" || lastNameBox == "" || emailBox == "" || loginBox == "")
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Рисунок 14 – Проверка на пустые поля

Квалификация: Разработчик веб и мультимедийных приложений

6. Также, как и предыдущие проверки, необходимо добавить данную проверку в код регистрации, как показано на рисунке 15.

```
if (userPassword.Text != "")
{
    if (userPassword.Text == userConfirmPassword.Text)
    {
        if (CheckTextBox())
        {
            if (CheckUserName())
            {
                if (CheckUserEmail())
                {
                    DB db = new DB();
                    MySqlCommand cmdReg = new MySqlCommand($"INSERT INTO users (firstname_user, lastname_user, email_user, login_user, password_user)
                                                                VALUES ('{nameUser}', '{lastnameUser}', '{emailUser}', '{loginUser}', '{userConfirmPassword.Text}');", db.GetConnection());

                    db.openConnection();
                    if (cmdReg.ExecuteNonQuery() == 1)
                    {
                        MessageBox.Show("Пользователь создан");
                        Close();
                    }
                    else
                    {
                        MessageBox.Show("Ошибка: Не удалось создать пользователя, повторите позже");
                    }
                    db.closeConnection();
                }
            }
            else
            {
                MessageBox.Show("Пользователь с такой почтой уже существует или вы ввели почту неправильно!");
            }
        }
        else
        {
            MessageBox.Show("Пользователь с таким логином уже существует!");
        }
    }
    else
    {
        MessageBox.Show("Не все поля заполнены");
    }
}
```

Рисунок 15 – Добавление третьей проверки в код регистрации

Квалификация: Разработчик веб и мультимедийных приложений

## **Составление отчета**

### **Оформление отчёта**

Для оформления отчёта допускается использование любых текстовых редакторов, в которых могут быть реализованы предъявляемые к оформлению отчёта требования, например, Microsoft Office, Open Office или подобные. Отчёт о выполнении работы оформляется в соответствии с методическими указаниями.

### **Содержание отчёта**

Отчёт о выполнении работы должен включать:

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Скриншоты, иллюстрирующие отображение всех форм проекта в режиме проектирования и в режиме выполнения программы.
5. Вывод.
6. Исходных код программы (файлы проекта с расширением \*.cs).