

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ГЕНЕРАТОР QR-КОДА

Студент


(подпись, дата)

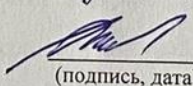
Ильин Сергей Русланович
(ФИО полностью)

Руководитель


(подпись, дата)

Карамышева Н.С.
(фамилия, инициалы)

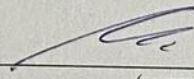
Нормоконтролёр


(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. № 14)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

14.06.2023г. № 14)

Секретарь ГЭК



Е.И. Гурин
(подпись)

(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 26 » 04 20 23 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Ильин Сергей Русланович гр. 19ВВП1 факультета вычислительной техники.

2. Тема работы: «Генератор QR-кода»

Тема утверждена приказом ПГУ № 357/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы Карамышева Н.С.

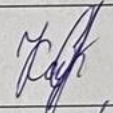
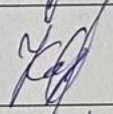
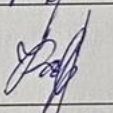
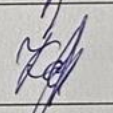
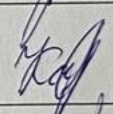
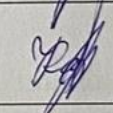
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Генератор QR-кода» на операционной системе *Windows*, позволяющее генерировать QR-код и хранить внутри QR-кода данные разных типов, а также имеющее возможность сохранять QR-код в виде файла.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить аналоги существующих генераторов QR-кода и произвести их анализ, анализ архитектуры QR-кода, сравнение существующих языков программирования, наиболее подходящих для решения задачи.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023 – 30.04.2023	
2.	Анализ требований	10	01.05.2023 – 02.05.2023	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2023 – 07.05.2023	
4.	Описание алгоритмов	40	08.05.2023 – 11.05.2023	
5.	Разработка приложения «Генератор QR-кода»	150	12.05.2023 – 29.05.2022	
6.	Оформление пояснительной записки	24	30.05.2023 – 31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

Руководитель бакалаврской работы

(подпись, дата)

Задание к исполнению принял студент

(подпись, дата)

Работу к защите допустить
Декан факультета

13.06.2023

(подпись, дата)

Реферат

Бакалаврская работа содержит 143 листа, 32 рисунок, 6 источников, 3 приложения

QR-КОД, АРИХИТЕКТУРА QR-КОДА, СОЗДАНИЕ QR-КОДА, ПОДКЛЮЧЕНИЕ К СЕТИ WI-FI С ПОМОЩЬЮ QR-КОДА, QR-КОД ВИЗИТКА, C++, C#, WIN FORMS, VISUAL STUDIO

Цель работы – разработать приложение на операционной системе Windows, позволяющее генерировать QR-код и хранить внутри QR-кода данные разных типов, а также имеющее возможность сохранять QR-код в виде файла.

Объектами разработки являются:

- нативная библиотека динамической компоновки для создания байтов данных;
- компонент графического интерфейса для отображения QR-кода в пользовательском интерфейсе;
- оконное приложение на основе WinForms для задания параметров генерации QR-кода непосредственно пользователем.

В результате проделанной работы разработано приложение, полностью соответствующее поставленной задаче, а именно: разработана нативная библиотека динамической компоновки на языке программирования C++, которая генерирует байты данных QR-кода, разработан компонент графического интерфейса на основе «PictureBox», который рисует QR-код по заданным параметрам и байтам данных и отображает их на экране, разработано оконное приложение с помощью WinForms, которое позволяет пользователю эффективно и удобно создавать и настраивать QR-код.

					ПГУ1.090301.08.001 ПЗ							
Изм.	Лист	№ докум.	Подпись	Дата	Генератор QR-кода Пояснительная записка			Лит.	Лист	Листов		
Раз раб..		Ильин С.Р.								44		
Провер.		Карамышева Н.С.						ФВТ, каф. ВТ гр. 19ВВП2				
		Бычков А.С.										
Н. Контр.												
Утверд.		Митрохин М.А.										

Содержание

Перечень сокращений и обозначений.....	7
Введение	8
1 Генератор QR-кода.....	9
1.1 Анализ решения поставленной задачи.....	9
1.1.1 Типы QR-кодов	9
1.1.2 Где используют QR-коды.....	9
1.1.3 Преимущества использования QR-кодов	10
1.1.4 Недостатки использования QR-кодов	11
1.2 Анализ <i>web-сервисов</i> для генерации QR-кодов	12
1.2.1 Сервис <i>qrcoder.ru</i>	12
1.2.2 Сервис <i>code-qr.ru</i>	13
1.2.3 Сервис <i>qr-online.ru</i>	14
1.3 Анализ алгоритма генерации QR-кода.....	15
1.3.1 Кодирование данных.....	16
1.3.2 Добавление служебной информации	17
1.3.3 Разделение информации на блоки.....	20
1.3.4 Создание байтов коррекции.....	22
1.3.5 Объединение блоков	25
1.3.6 Размещение информации на QR-коде	26
1.4 Постановка задачи.....	33
1.4.1 Анализ архитектуры генератора QR-кода.....	33
1.5 Выбор языка программирования.....	34
1.6 Разработка приложения	35
1.6.1 Архитектура.....	35
1.6.2 Разработка графического интерфейса	36
1.6.3 Разработка моделей.....	38
1.6.4 Описание доступных функций	49
1.7 Описание программы.....	57
1.7.1 Нативная <i>DLL</i>	57
1.7.2 Элемент пользовательского интерфейса <i>QrBox</i>	58

1.7.3 Пользовательский интерфейс	59
2 Безопасность жизнедеятельности	60
2.1 Характеристика потенциальных опасностей и вредностей	60
2.2 Санитарно-гигиеническая характеристика производства.....	61
2.3 Электробезопасность	63
2.4 Пожарная безопасность	63
Заключение.....	65
Список использованных источников	66
Приложение А – <i>UML</i> –диаграммы	67
Приложение Б– Листинг.....	72
Приложение В– Презентация	129

Перечень сокращений и обозначений

В настоящей работе применяются следующие сокращения и обозначения

ГОСТ - государственный общесоюзный стандарт

ИС - информационная система

ПК - персональный компьютер

ПЭВМ - персональная электронно-вычислительная машина

РФ - российская федерация

СанПиН - санитарно-эпидемиологические правила и нормы

ЭВМ - электронно-вычислительная машина

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

DLL (Dynamic Link Library) – библиотека динамической компоновки

E-mail (Electronic mail) – технология и служба по пересылке и получения электронных сообщений между пользователями компьютерной сети

IDE (Integrated Development Environment) – интегрированная среда разработки

QR (Quick Response) – технология быстрого отклика, используемая в QR-коде

SMS (Short Message Service) – технология приёма и передачи коротких текстовых сообщений с помощью сотового телефона

UML (Unified Modeling Language) – унифицированный язык моделирования

Wi-Fi (Wireless Fidelity) – технология беспроводной локальной сети

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

Введение

QR-код — это двумерный тип штрих-кода, который легко считывается цифровыми устройствами и хранит информацию в виде ряда пикселей в квадратной сетке, которая выглядит как черно-белый узор. QR-коды считываются в горизонтальном и вертикальном направлении, в отличие от штрих-кодов. Это позволяет увеличить объём содержащейся в нём информации. Когда QR-код отсканирован, пользователь может немедленно получить доступ к этим данным.

В настоящее время QR-коды используются во многих отраслях промышленности, как для хранения данных (ссылка на веб-сайт, номер телефона или текст), так и для обеспечения доступа к *Wi-Fi*, отправки и получения платежной информации и других целей.

В 2020 году многие люди впервые столкнулись с QR-кодами. Сначала они были введены для контроля системы самоизоляции и передвижения, а затем их стали использовать для передачи вакцинированных людей в различные учреждения. По данным агентства *Bitly*, за полтора года пандемии количество считываний QR-кодов по всему миру увеличилось на 750%.

Во время последнего карантина, который начался 28 октября и продолжался до 7 ноября, во многих регионах России больным коронавирусом был разрешен доступ в культурные и развлекательные учреждения только с помощью вакцинации и QR-кодов[1].

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

1 Генератор QR-кода

1.1 Анализ решения поставленной задачи

QR— это двумерный тип штрих-кода, который легко считывается цифровым устройством и хранит данные в виде серии пикселей в квадратной сетке, которая внешне выглядит как черно-белый узор. QR-код, в отличие от штрих-кода, читается в двух направлениях — по горизонтали и по вертикали. Это позволяет хранить в нем больше данных. При сканировании QR-кода пользователь получает доступ к этим данным мгновенно [1].

1.1.1 Типы QR-кодов

QR-коды бывают:

- статические — используется для информации, которая не может быть отредактирована после создания кода. Подходит для стационарных операций: кодирования настроек Wi-Fi, идентификации сотрудников, регистрации событий и передачи данных;

- динамические — информация обновляется в любое время. Компании выбирают эти коды для сбора статистики, комментариев и маркетинговых мероприятий.

1.1.2 Где используют QR-коды

Сфера использования кода не ограничена. Некоторые примеры:

- маркетинговые задачи — рекламные мероприятия, другая информация о продукте, цифровые визитные карточки;

- приём платежей по QR-коду — можно заменить обычный кассовый аппарат и совершайте покупки;

- контроль товаров в производственных и инспекционных органах;

- контроль сотрудников — доступ к личным местам или файлам, идентификация персонала;

- передвижение людей — передвижение во время внутренней пандемии и во время транзита;

- навигация — самостоятельные экскурсии для туристов;
- образовательные цели — контроль посещаемости с помощью QR-кода или доступа к обучающим материалам;
- переход к дополненной и виртуальной реальности. Пользователь считывает код и переходит на страницу с демонстрацией товара в действии.

1.1.3 Преимущества использования QR-кодов

Для сканирования не требуется никакого специального оборудования, требуется только приложение на смартфоне. Вы можете сделать снимки, а затем прочесть QR-код бесплатно. После сканирования пользователь получит ресурсы без ошибок, и ему не придется вручную вводить адрес сайта и другие данные - это ускоряет доступ к информации и покупкам.

Клиентам не нужны наличные или карта для оплаты QR-кодов. Вам не нужно вводить данные учетной записи /карты или номер телефона для перевода — риск ошибок меньше. Получить платеж с помощью QR-кода просто: если код статичен, пользователь сразу получает платежные реквизиты. Используя динамические QR-коды, вы можете персонализировать платежи и предоставлять пользователям ссылку для оплаты на определенную сумму. Этот метод ускорит и упростит платежи, а также защитит от фишинговых сайтов и ошибки.

Купоны на скидку, другая информация о товаре и ссылки на конкурсы — все это размещено в QR-коде. С помощью QR-кода они приглашают вас на мероприятие или сообщают вам программу мероприятия, а также регистрируют гостей на мероприятии.

С помощью QR-кодов маркетологи могут отслеживать местоположения, которые привлекательны для пользователей в торговом зале. В местах, где выполняется больше сканирований, QR-код обычно замечается именно там. Это рабочий инструмент для оценки эффективности наружной рекламы и отслеживания пути клиентов.

Во время пандемии рестораны начали составлять цифровые меню, чтобы уменьшить количество контактов с поверхностями. Но этот метод по-прежнему удобен даже в отсутствие пандемии.

QR-коды также заменяют печатную рекламу. Это особенно выгодно, когда печатаются листовки для краткосрочных рекламных акций. Код вместо офлайн-материалов упрощает маркетинг и снижает затраты компании

Если в офлайн-магазине мало места, QR-код отобразит весь ассортимент без необходимости поиска и распечатки каталога на веб-сайте. Тогда компании не нужно арендовать помещение большего размера для размещения всего товара.

QR-код выполнен в черно-белом цвете и не требует цветной печати. Чтение под разными углами - возможность размещать QR-коды в любом пространстве: баннерах, бумажных стаканчиках, печатных листовках, автомобилях и т.д. Даже если поверхность повреждена на 30%, она не пострадает - сканер все равно распознает код и подсчитывает информацию.

1.1.4 Недостатки использования QR-кодов

Не все понимают, как использовать QR-коды. Прежде всего, компаниям придется провести исследование и выяснить, насколько развита целевая аудитория и возникнут ли у пользователей трудности.

Новички часто сталкиваются с сомнениями. Вдруг клиенты решат, что вы хотите снять деньги с их счетов таким "хитрым" способом. Цифровая грамотность не везде находится на высоком уровне, поэтому такой риск существует. Недоверчивым пользователям следует объяснить, почему вы перешли на новую технологию, как проверять QR-коды и почему эта технология безопасна. Иногда это бывает нелегко.

Даже для продвинутых и хорошо обученных пользователей QR-коды не всегда удобны. Если нет приложения для сканирования, им придется потратить время и скачать его. Или пользователь забудет код от мобильного приложения для совершения платежа. Оставив один способ оплаты или получения информации с помощью QR-кода, вы рискуете потерять клиентов.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

В 2021 году 90% людей в мире уже имели доступ к быстрому Интернету, а мобильный трафик в 2022 году достиг миллионов терабайт. Однако эти системы не всегда работают стабильно, и отсутствие интернета в неожиданный момент - реальная и знакомая ситуация для каждого. Интернет не работает из-за отсутствия средств на счете, из-за неудачного местоположения или просто из-за проблем с оператором мобильной связи или поставщиком, если подключение осуществляется через *Wi-Fi*. Прочитать QR-коды для оплаты товаров без доступа к Интернету невозможно, поэтому здесь выигрывает возможность оплаты через платежную систему *GooglePay/ApplePay*, где интернет не требуется.

1.2 Анализ *web-сервисов* для генерации QR-кодов

Для генерации QR-кодов существует множество решений. Например, для генерации можно использовать уже готовые *web-сервисы*. В настоящее время существует множество различных бесплатных *online-сервисов*, которые предоставляют возможность создания QR-кодов.

В процессе работы были протестированы следующие онлайн генераторы QR-кодов:

- *qrcoder.ru*;
- *code-qr.ru*;
- *qr-online.ru*.

1.2.1 Сервис *qrcoder.ru*

Данный сервис имеет простой в использовании интерфейс (рисунки 1). Также можно получить постоянную ссылку на QR-код.

С помощью этого сервиса можно сгенерировать QR-код со следующей информацией:

- текст;
- адрес сайта;
- визитка;
- SMS-сообщение.

ГЕНЕРАТОР QR КОДОВ

закодировать: **любой текст** [ссылку на сайт](#) [визитную карточку](#) [sms-сообщение](#)

введите текст для кодирования:

размер: ☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐ 6

создать код (ctrl+enter)

ИСПОЛЬЗОВАНИЕ:

1. Возьмите мобильный телефон с камерой,
2. Запустите программу для сканирования кода,
3. Наведите объектив камеры на код,
4. Получите информацию!

ПРИМЕНЕНИЕ:

В качестве применения qr-кодов можно назвать: размещение их изображений в интернете, нанесение на визитные карточки, футболки, рекламные вывески и многое другое.

ЧТО ТАКОЕ QR-КОД:

QR код «QR - Quick Response - Быстрый Отклик» — это двухмерный штрихкод (бар-код), предоставляющий информацию для быстрого ее распознавания с помощью камеры на мобильном телефоне.

При помощи QR-кода можно закодировать любую информацию, например: текст, номер телефона, ссылку на сайт или визитную карточку.

ВАШ QR-КОД:



Постоянная ссылка на изображение:

<http://qrcoder.ru/code/?%7B8&48.0>

HTML-код для вставки в блог:

` correction_; }
private set { } }
public byte Version { get => version_; private set { } }
public byte[] QrCodeData { get; set; }
public Image Logo { get; set; } = null;
public bool ModuleSizeAuto { get; set; } = true;
public int ModuleSize { get => moduleSize_; set { moduleSize_ = value; } }
ModuleSizeAuto = false; }
public Color ModuleColor { get => moduleColor_; set => moduleColor_ = value; }
public Color BackgroundColor { get => backgroundColor_; set => backgroundColor_ = value; }
public bool SelectBestMask { get; set; } = false;

public int SetVersion(byte version)
{
 int result = 0;
 result = QrCodeDllCore.QrCodeSetVersion(QrCode, version);
 if (result != 0)
 return result;

 version_ = version;
 return result;
}

public int SetCorrectionLevel(QrCodeCorrectionLevel correctionLevel)
{
 int result = 0;
 result = QrCodeDllCore.QrCodeSetCorrectionLevel(QrCode, correctionLevel);
 if (result != 0)
 return result;

 correction_ = correctionLevel;
 return result;
}

public Bitmap GetImage(){...}
virtual void SetText(string text)
{
 byte[] textBytes = Encoding.UTF8.GetBytes(text);
 if (QrCodeDllCore.QrCodeGenerate(QrCode, textBytes, (short)(textBytes.Length)) != 0)
 {
 MessageBox.Show("Слишком много данных для данной версии и уровня коррекции");
 return;
 }
}

```

```

 ushort sizeQrCodeData =
 QrCodeDllCore.QrCodeGetDataSize(QrCode);

 QrCodeData = newbyte[sizeQrCodeData];
 QrCodeDllCore.QrCodeRead(QrCode, 0, QrCodeData,
 sizeQrCodeData);

 version_ = (byte)(QrCodeDllCore.QrCodeGetVersion(QrCode) -
 1);
 correction_ = QrCodeDllCore.QrCodeGetCorrectionLevel(QrCode);
 }

 protected override void OnPaint(PaintEventArgs pe)
 {
 base.OnPaint(pe);
 pe.Graphics.Clear(Color.White);

 Bitmap qr = this.GetImage();
 pe.Graphics.DrawImage(qr, new Point(0, 0));
 }

 ...
}

```

Описание методов и полей данного класса:

- *Dispose* - метод для освобождения ресурсов;
- *sizesLogo* - массив для хранения максимальных размеров логотипа;
- *AlciliatingPoint* - массив, содержащий координаты выравнивающих узоров для каждой версии QR-кода;
- *data* - массив, в который заносятся готовые данные, для отрисовки QR-кода;
- *QrCodeSize\_* - поле для хранения размера стороны QR-кода;
- *moduleSize\_* - поле для хранения размера модуля;
- *moduleColor\_* - поле для хранения цвета модуля;
- *backgroundColor\_* - поле для хранения цвета фона;
- *bmp\_* - класс «*Bitmap*» на котором рисуется QR-код;
- *QrCode* - указатель на нативный объект класса «*QrCode*»;
- *version\_* - поле с текущей версией QR-кода;
- *correction\_* - поле с текущим уровнем коррекции QR-кода;
- *CorrectionLevel* - свойство для получения и задания уровня коррекции QR-кода;

- *Version* - свойство для получения и задания версии QR-кода;
- *QrCodeData* - свойство для получения и задания байт данных QR-кода;
- *Logo* - свойство для получения и задания логотипа QR-кода;
- *ModuleSizeAuto* - свойство для задания или отключения автоматического определения размеров модулей;
- *ModuleSize* - свойство для задания или получения размера модулей;
- *ModuleColor* - свойство для задания или получения цвета модулей;
- *BackgroundColor* - свойство для задания или получения цвета фона;
- *SelectBestMask* - свойство для задания или отключения функции определения лучшей маски для QR-кода
- *SetVersion* - метод для установления версии;
- *SetCorrectionLevel* - метод для установления уровня коррекции;
- *GetImage* - метод для получения изображения QR-кода;
- *SetText* - метод для установления строки, которую необходимо записать в QR-код;
- *OnPaint* - переопределенный метод для перерисовки QR-кода, вызывается при создании элемента пользовательского интерфейса, также при вызове метода базового класса «*Refresh*».

#### 1.6.4 Описание доступных функций

Приложение предполагает реализацию таких функций, как:

- генерация QR-кода для текстовой информации;
- генерация QR-кода для ссылок;
- генерация QR-кода для SMS;
- генерация QR-кода для Wi-Fi;
- генерация QR-кода для электронной почты;
- генерация QR-кода для визиток;
- изменение версии QR-кода;
- изменение уровня коррекции QR-кода;
- добавление логотипа на QR-код;
- сохранение QR-кода в файл.

При запуске приложения нас встречает пользовательский интерфейс, позволяющий удобно генерировать и настраивать QR-код.

Для создания для текста нужно выбрать в меню слева вкладку «Текст». После этого появится поле для ввода текстовой информации. QR-код генерируется после нажатия на кнопку «Сгенерировать QR-код» (рисунок 19).

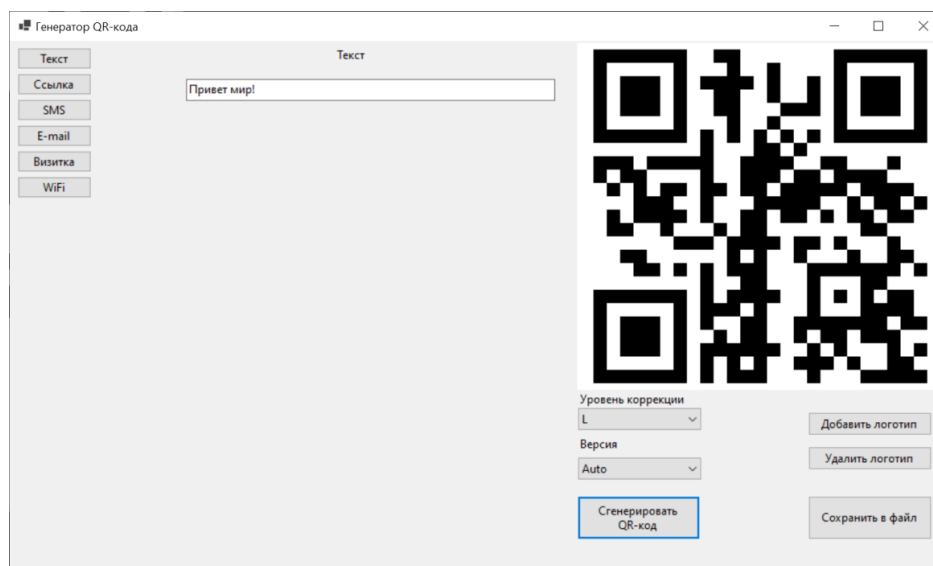


Рисунок 19 - Генерация текста

Для создания для ссылки нужно выбрать в меню слева вкладку «Ссылка». После этого появится поле для ввода ссылки. Можно вводить сразу название сайта без *http://www.* (рисунок 20).

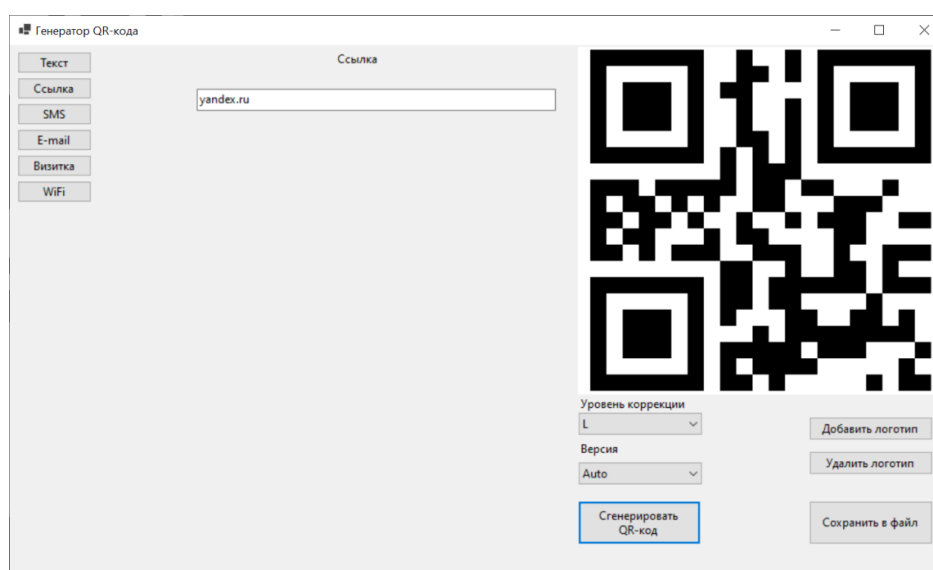


Рисунок 20 - Генерация ссылки

Для генерации QR-кода с SMS сообщением нужно выбрать в меню слева вкладку «SMS». После этого отобразятся два текстовых поля для ввода. В первое текстовое поле необходимо ввести номер телефона, на который будет отправлено SMSсообщение. Во втором поле вводится само сообщение для отправки (рисунок 21).

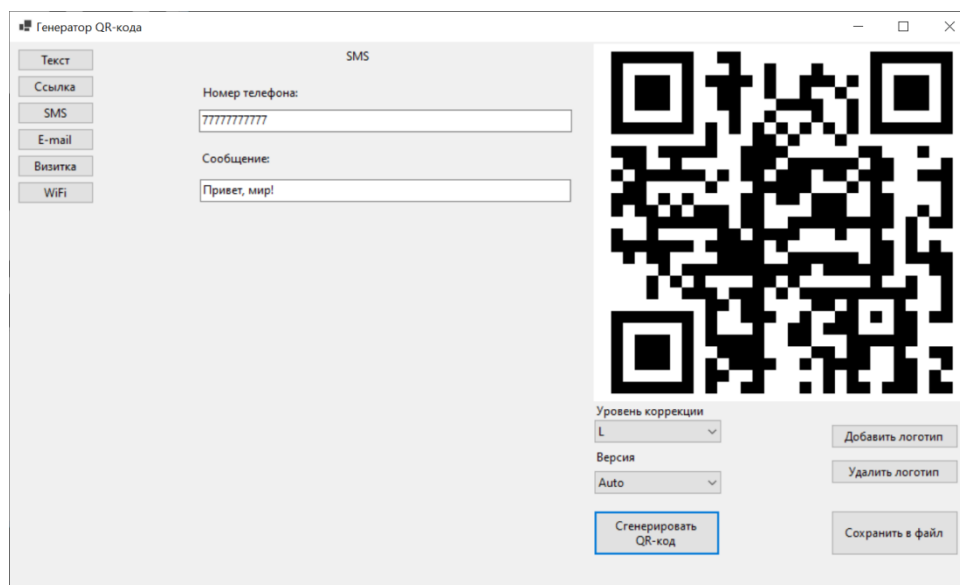


Рисунок 21 - Генерация SMS сообщения

Для генерации QR-кода с сообщением для электронной почты нужно выбрать в меню слева вкладку «E-mail». После этого отобразятся три текстовых поля для ввода. В первое текстовое поле необходимо ввести адрес электронной почты, на который будет отправлено сообщение. Во втором поле вводится тема сообщения для отправки. В третьем поле вводится само сообщение (рисунок 22).

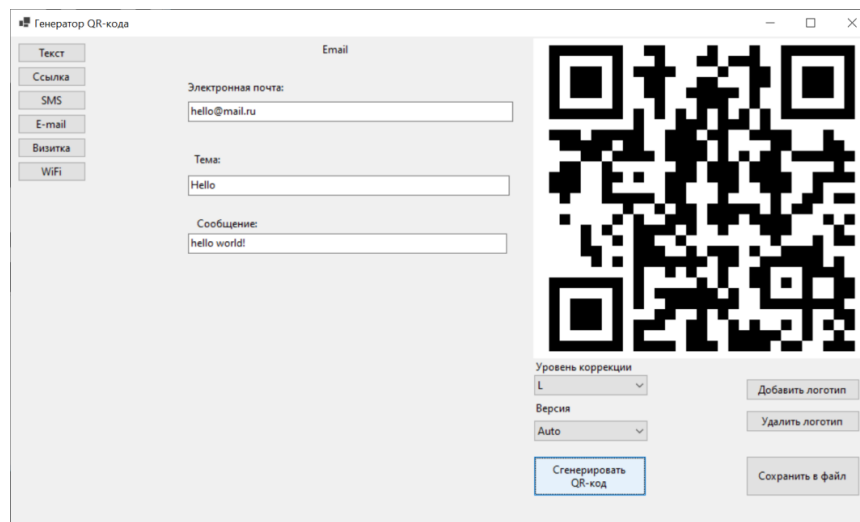


Рисунок 22 - Генерация сообщения на *e-mail*

Для генерации QR-кода с визиткой нужно выбрать в меню слева вкладку «Визитка». После этого отобразится форма для заполнения, включающая в себя: ФИО, электронную почту, веб сайт, номера телефона городского и мобильного, факс, информацию о компании, в которой работает человек, а именно название компании и занимаемая должность, информация о местоположении, а именно страна, область, город, улица, дом, почтовый индекс. Не обязательно заполнять все поля, достаточно заполнить хотя бы одно из них, чтобы создавалась визитка (рисунок 23).

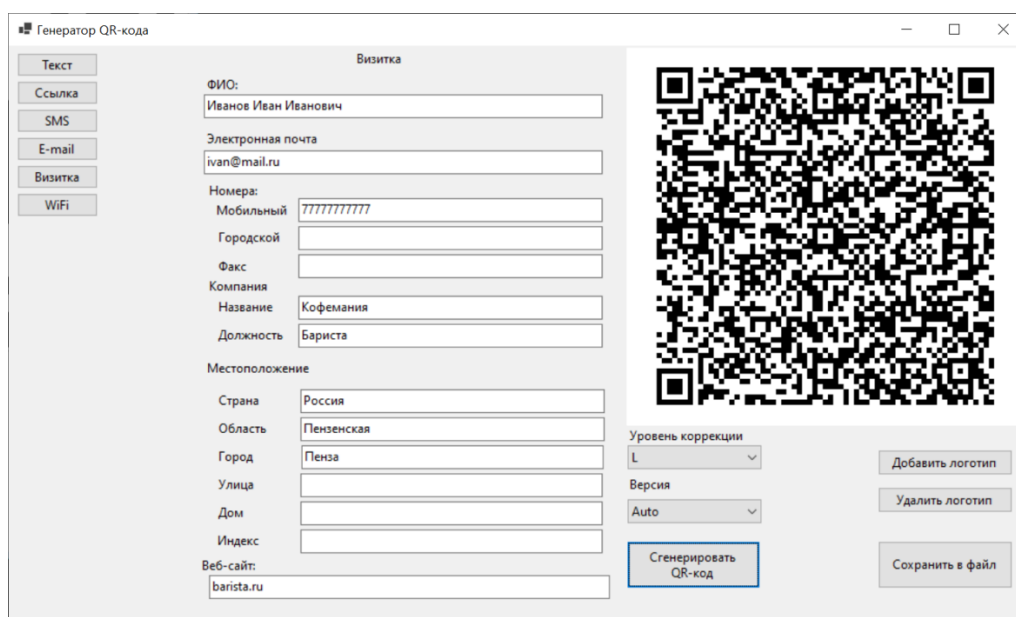


Рисунок 23 - Генерация визитки

Чтобы сгенерировать QR-код, для быстрого подключения к сети *Wi-Fi*, необходимо выбрать в меню слева вкладку «*Wi-Fi*». После этого отобразится форма для заполнения. В ней необходимо заполнить поля «Название сети» и «Пароль», а также выбрать один из методов шифрования. Так же имеется возможность подключаться к скрытой сети, для этого нужно отметить галочкой пункт «Скрытая». При необходимости есть возможность скрывать пароль при вводе, для этого нужно поставить галочку в графе «Скрывать пароль» (рисунок 24).

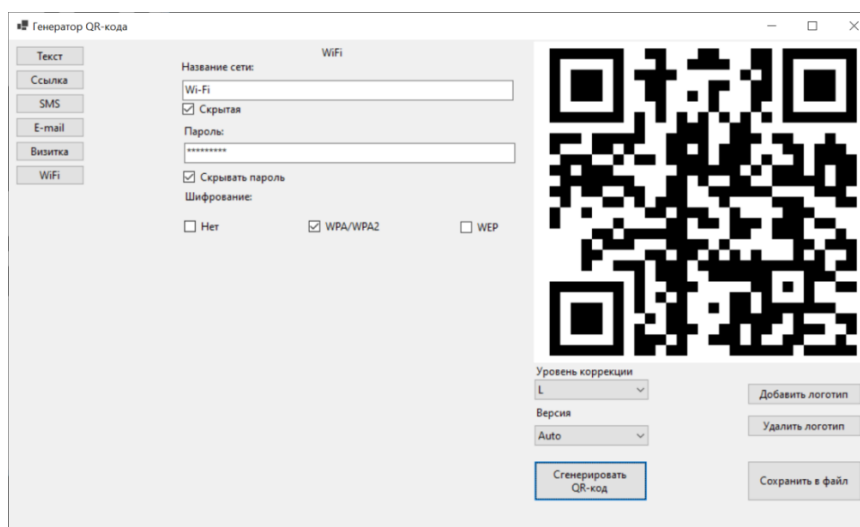


Рисунок 24 - Генерация QR-кода для подключения к *Wi-Fi*

Приложение имеет функцию изменения версии QR-кода. От версии зависит размер самого QR-кода и количество данных возможное записать в него. Для изменения версии нужно открыть вкладку «Версия» в правом нижнем углу и выбрать нужную вам версию (рисунок 25). Если количество введенных вами данных слишком велико для данной версии, то QR-код не сгенерируется, а на экране появится предупреждение (рисунок 26).

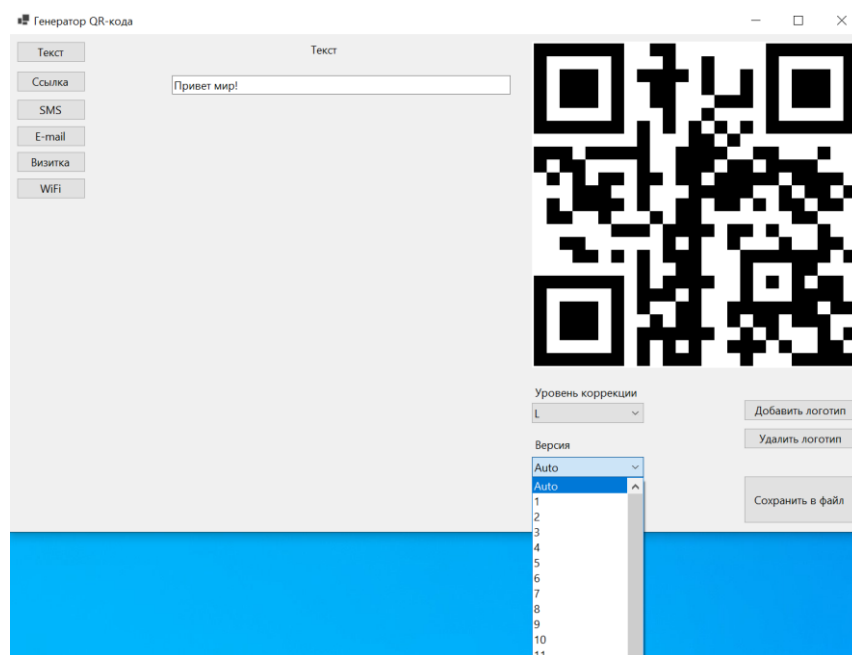


Рисунок 25 - Выбор версии

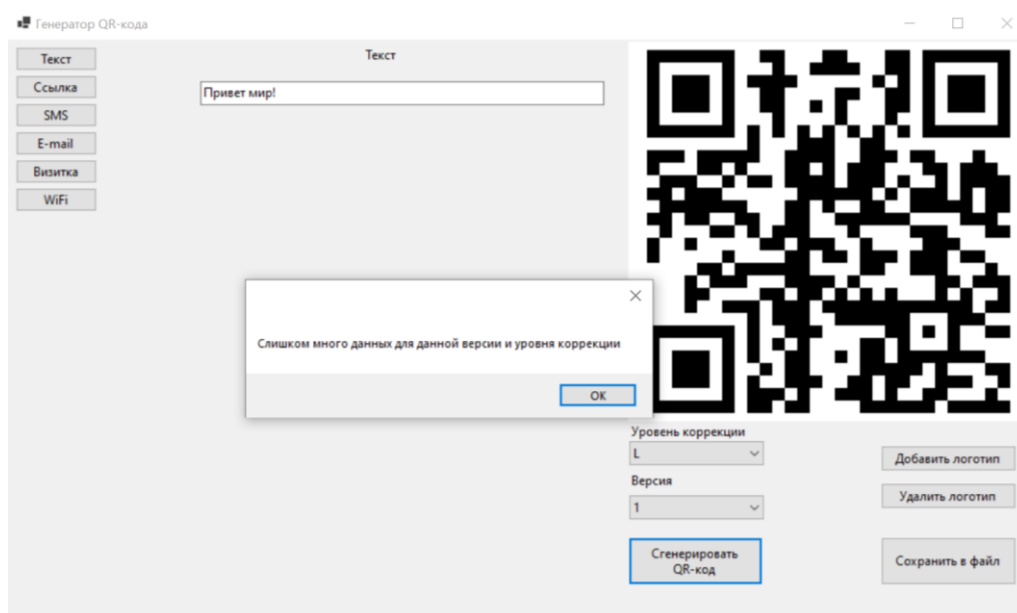


Рисунок 26 - Сообщение об ошибке

Так же имеется возможность изменения уровня коррекции QR-кода. От уровня коррекции зависит то на сколько QR-код может быть повреждён. Для изменения уровня коррекции нужно открыть вкладку «Уровень коррекции» в правом нижнем углу и выбрать нужный вам уровень коррекции (рисунок 27). Изменение уровня коррекции так же влияет на количество данных, которые можно записать в QR-код. Если количество введённых вами данных слишком

велико для данного уровня коррекции, то QR-код не сгенерируется, а на экране появится предупреждение.

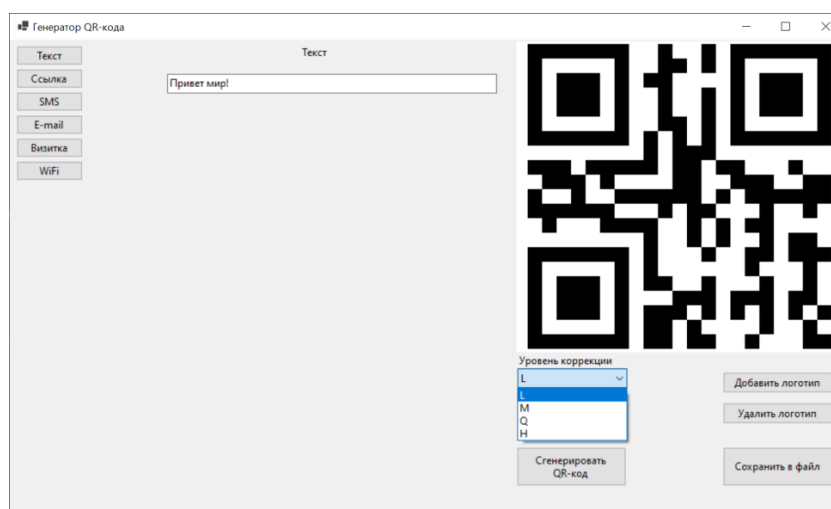


Рисунок 27 - Выбор уровня коррекции

На QR-коде можно разместить логотип. Для этого нужно нажать на кнопку «Добавить логотип», после этого откроется проводник, в котором можно выбрать нужную вам картинку для логотипа (рисунок 28). При выборе логотипа уровень коррекции изменяется на самый высокий «H», при этом вкладка изменения уровня коррекции блокируется (рисунок 29).

Для удаления логотипа с QR-кода, необходимо нажать на кнопку «Удалить логотип». После этого логотип удаляется, а вкладка выбора уровня коррекции снова становится активной.

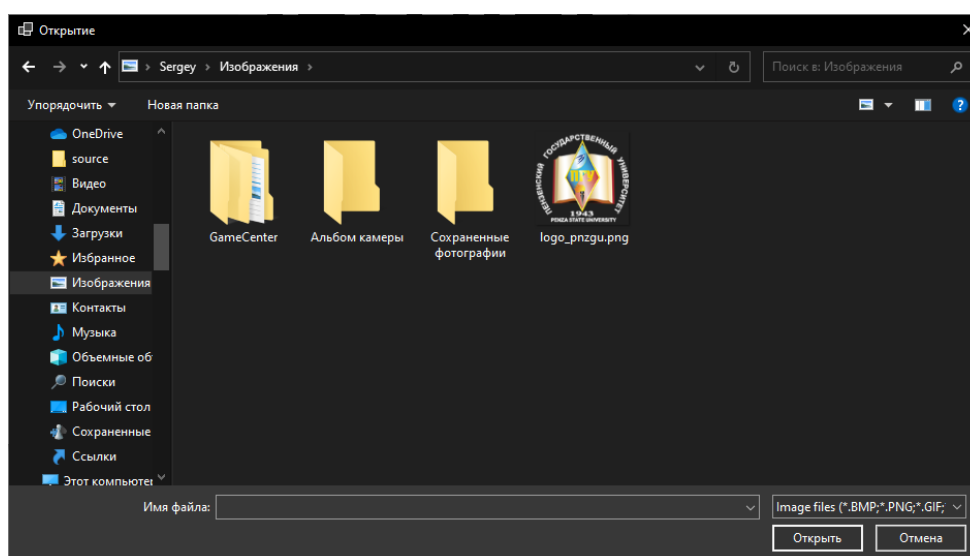


Рисунок 28 - Выбор логотипа

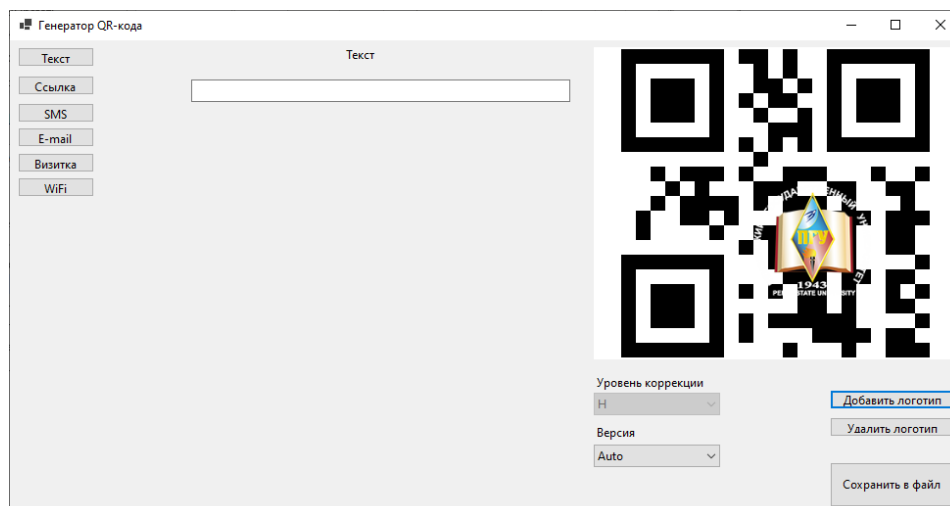


Рисунок 29 - QR-код с логотипом

Сгенерированный QR-код можно сохранить в файл, для этого необходимо нажать на кнопку «Сохранить в файл». После этого открывается проводник, в котором можно выбрать место для сохранения изображения, а также его формат (рисунок 30). После этих действий изображение сохранится по пути, который вы выбрали (рисунок 31).

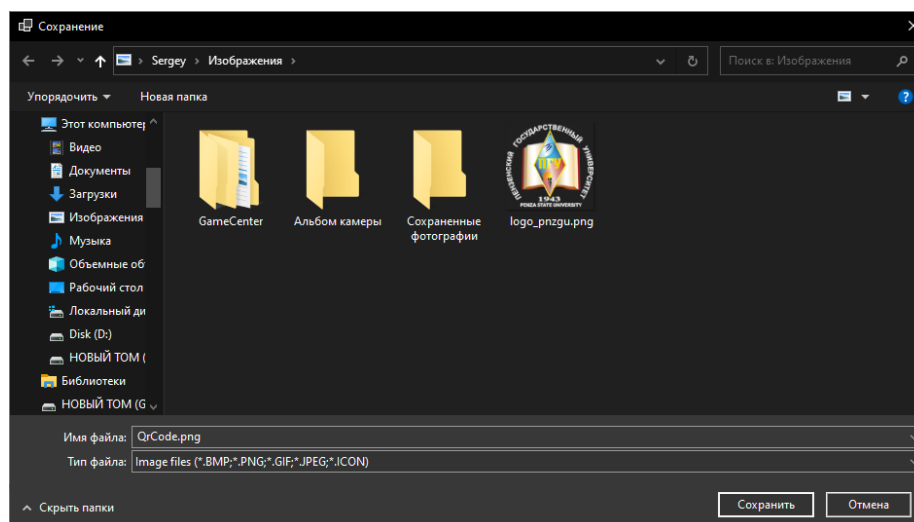


Рисунок 30 - Выбор папки для сохранения

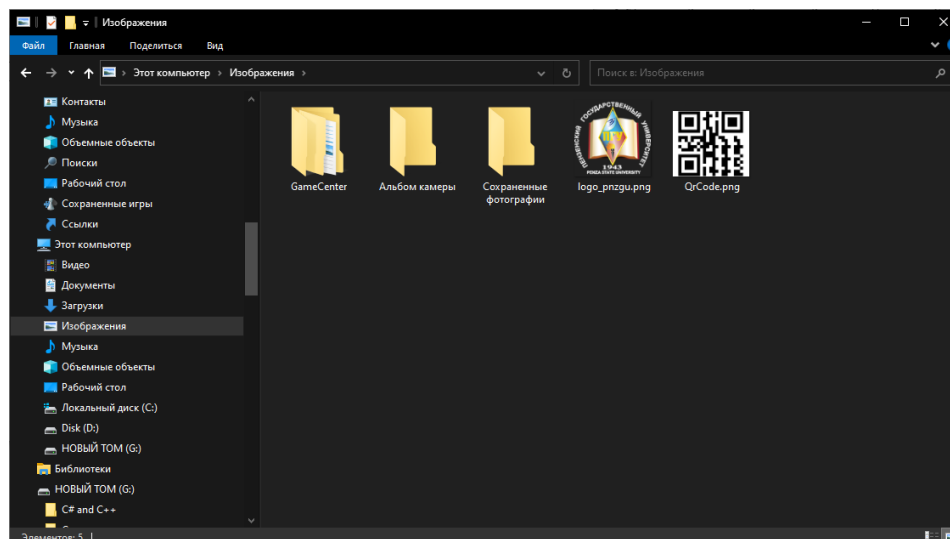


Рисунок 31 - Сохраненное изображение QR-кода

## 1.7 Описание программы

### 1.7.1 Нативная DLL

Нативная *DLL* написана на языке программирования *C++*. Она предназначена для генерации байтов данных. Язык *C++* был выбран из-за того, что при генерации байтов данных используется множество операций с битами. Для того чтобы все процессы генерации байтов данных были быстрыми, было принято решение использовать язык программирования.

Так как это библиотека динамической компоновки, ее можно будет использовать для других проектов и на других языках программирования.

Нативная *DLL* состоит из следующих файлов:

- *CorrectionLevel.h*;
- *QrCodeInfo.h*;
- *QrCodeInfo.cpp*;
- *QrCode.h*;
- *QrCode.cpp*;
- *QrCodeDllExports.h*;
- *QrCodeDllExports.cpp*.

Файл *CorrectionLevel.h* содержит в себе перечисление для удобного определения уровня коррекции QR-кода.

Файлы *QrCodeInfo.h* и *QrCodeInfo.cpp* содержат в себе структуру «*QrCodeInfo*» для удобного хранения информации о QR-коде, а именно максимальный размер данных, которые можно записать в QR-код, количество блоков данных и количество байтов коррекции. Так же в файле *QrCodeInfo.h* хранится матрица (*g\_qrCodeInfo*) изначально проинициализированных объектов *QrCodeInfo*.

Файлы *QrCode.h* и *QrCode.cpp* содержат в себе класс «*QrCode*», который генерирует байты данных. Этот объект содержит в себе методы для установки и получения текущей версии QR-кода (*GetVersion*, *SetVersion*). Если установить версию 0, то версия будет определяться автоматически, после генерации байтов данных. Так же присутствуют аналогичные методы для изменения и получения уровня коррекции (*SetCorrectionLevel*, *GetCorrectionLevel*). Для генерации байтов данных необходимо использовать метод «*Generate*». После вызова этого метода можно будет считать данные с помощью метода «*Read*», а также узнать количество байтов данных с помощью метода «*Size*».

Файлы *QrCodeDllExports.h* и *QrCodeDllExports.cpp* содержат в себе экспортируемые функции, которые являются обёрткой над объектом «*QrCode*».

### 1.7.2 Элемент пользовательского интерфейса *QrBox*.

Элемент пользовательского интерфейса *QrBox* представляет из себя компонент графического интерфейса (рисунок 32), написанный на языке программирования *C#* использующий *.NET WinForms*. Этот элемент пользовательского интерфейса состоит из следующих файлов:

- *QrCodeDllCore.cs*;
- *QrBox.cs*.

Файл *QrCodeDllCore.cs* содержит все классы «*QrCodeDllCore*», который импортирует методы из нативной *DLL*.

Файл *QrBox.cs* содержит в себе объект «*QrBox*», который является элементом пользовательского интерфейса, а именно QR-кодом, и отображает

QR-код на форме. Объект является наследуется от стандартного контрола WinForms «PictureBox» и имеет все его свойства и методы. Для того, чтобы отобразить QR-код на форме, необходимо вызвать методы объекта QrBox–SetText. Этот метод принимает строку, которую необходимо записать в QR-код. Для установки версии QR-кода нужно использовать свойство «Version», внутри себя свойство обращается к нативной DLL. Аналогично работает свойство для установки уровня коррекции «CorrectionLevel». Чтобы добавить логотип на QR-код нужно воспользоваться свойством «Logo», которое принимает объекты типа Image. Так же есть возможность изменять размеры и цвет модулей на QR-коде используя свойства «ModuleSize» и «ModuleColor» соответственно.

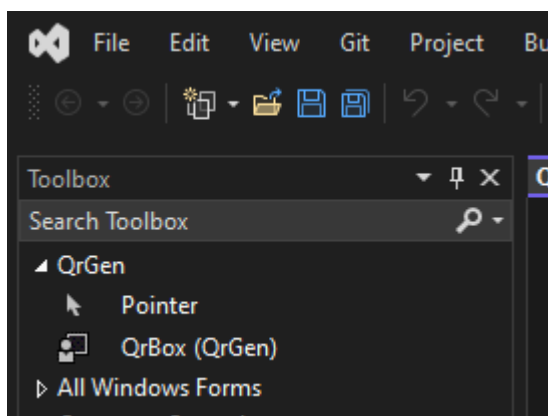


Рисунок 32 - Отображение QrBox в панели инструментов

### 1.7.3 Пользовательский интерфейс

Пользовательский интерфейс содержит в себе форму WinForms «QrGen», на которой располагаются все элементы пользовательского интерфейса.

В файле QrGen.cs содержатся методы для обработки событий элементов управления.

## 2 Безопасность жизнедеятельности

Реализация любого проекта, будь то полноценный производственный процесс или работа над конкретной задачей, требует не только экономических затрат и умственных или физических усилий, но и обеспечения труда в безопасных для здоровья работников условиях. Для того, чтобы предоставить равные условия для работы, была разработана система правил и мер, которая легла в основу целой науки, основной целью которой стали систематизация знаний и обучение квалифицированных работников охране труда. Безопасность жизнедеятельности представляет собой свод правил, важнейшей целью которых является разработка и обновление мер по предупреждению производственного травматизма и заболеваемости.

В случае работы на ЭВМ, основными факторами, влияющими на работников, будут особенности организации рабочего пространства, удобство локации, а также самого места работы, условиями внутреннего расположения человека в помещении (освещение рабочего места, уровень шума, качеством техники и др.), общий характер работы.

В данной главе будут рассмотрены опасности, которые могут возникнуть в процессе реализации данного проекта, пожарные и санитарные нормы, применяемые к помещению, где должен реализовываться проект, санитарно-гигиеническая характеристика места разработки проекта, а также основные требования электробезопасности.

### 2.1 Характеристика потенциальных опасностей и вредностей

Для того, чтобы определить основные потенциальные опасности, присутствующие при разработке данного проекта, нужно определить, в каких условиях может производиться данный проект. Данные опасности можно определить, проведя анализ рисков, составленный на основе практического опыта. Так, среди главных опасностей для здоровья человека согласно ГОСТу

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <b>ПГУ1.090301.08.001 ПЗ</b> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                              | 60   |

12.0.003-04 были определены факторы, часть из которых может особенно иметь влияние на исполнителя. Среди них:

- повышенная или пониженная температура/влажность окружающей среды, повышенный уровень шума;
- повышенное напряжение в электросети;
- повышенный уровень электромагнитных излучений - электромагнитное и электростатическое поля, акустический шум, изменение ионного состава воздуха и параметров микроклимата в помещении также могут влиять на процесс работы;
- недостаточная освещенность рабочего места и зрительное напряжение;
- монотонность трудового процесса и нервно-эмоциональные перегрузки - нервно-эмоциональное напряжение при работе за персональным компьютером возникает вследствие дефицита времени, большого объема и плотности информации;
- длительное нахождение в одном положении, зачастую неправильном - повышенные статические и динамические нагрузки у пользователей ПК приводят к жалобам на боли в спине, шейном отделе позвоночника и руках. Из всех недомоганий, обусловленных работой на компьютерах, чаще встречаются те, которые связаны с использованием клавиатуры и неправильном положении при сидячей работе.

## 2.2 Санитарно-гигиеническая характеристика производства

Одними из важнейших требований при организации рабочего места сотрудника, независимо от его должности, являются обеспечение безопасных, подходящих под конкретного человека, условий для работы, а также недопущение возникновения заболеваний и несчастных случаев на рабочем месте. Общие требования к организации рабочих мест регулируются Трудовым кодексом, санитарно-эпидемиологическими правилами и нормативами (СанПиН), а также другими правовыми документами.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 61   |

В настоящее время действуют СанПиН 2.2.4.335916 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах», утвержденные Постановлением Главного государственного санитарного врача РФ от 21.06.2016 № 81 (предназначены для граждан, состоящих в трудовых отношениях, индивидуальных предпринимателей и юридических лиц). Этим документом установлены санитарно-эпидемиологические требования к физическим факторам на рабочих местах и источникам этих физических факторов, а также требования к организации контроля, методам измерения физических факторов на рабочих местах и мерам профилактики вредного воздействия физических факторов на здоровье работающих.

В отношении офисных работников действуют также следующие санитарные нормы и правила:

- при работе с компьютером с плоским монитором рабочее место должно иметь площадь не менее 4,5 кв.м;
- при размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами должно быть не менее 2м; размещать рабочие столы следует таким образом, чтобы естественный свет падал на дисплей преимущественно слева;
- экран видеомонитора должен находиться на расстоянии 600 — 700мм от глаз пользователя, но не ближе 500мм с учетом размеров алфавитно-цифровых знаков и символов;
- рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также по расстоянию спинки от переднего края сиденья. При этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию. При выборе типа рабочего стула (кресла) следует учитывать рост пользователя, характер и продолжительность работы с ПЭВМ. Поверхности сиденья, спинки и других элементов стула (кресла) должны быть полумягкими, с нескользящим, слабо электризующимся и

воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений;

– для сидячих рабочих мест оптимальными параметрами микроклимата на рабочем месте являются: температура воздуха в холодный период года — от 22 до 24 °С, в теплый период года — от 23 до 25 °С; относительная влажность воздуха на постоянных рабочих местах — 40 — 60%, скорость движения воздуха — 0,1м/с.

## 2.3 Электробезопасность

При рассмотрении вопроса обеспечения электробезопасности разработчика необходимо выделить три основных фактора:

- электроустановки рабочего места программиста;
- вспомогательное электрооборудование;
- окружающая среда помещения.

Помещения, оборудованные вычислительной техникой, как правило, относятся к категории помещений без повышенной опасности так как:

- относительная влажность воздуха не превышает 75%;
- нет токопроводящей пыли и токопроводящих полов;
- температура не превышает длительное время 30 °С;
- отсутствует возможность одновременного прикосновения человека с имеющими соединение с землей металлическими конструкциями.

Таким образом, для предотвращения электротравматизма пользователя, необходимо соблюдать требования безопасности, как при работе с обычной бытовой техникой.

## 2.4 Пожарная безопасность

Меры пожарной безопасности к помещениям, которые по своему назначению можно отнести к офисам или кабинетам, определяют следующие нормы, правила, изложенные как в законодательной, так и в нормативно-

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <b>ПГУ1.090301.08.001 ПЗ</b> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                              | 63   |

технической базе, например, в Федеральном законе номер 69 «О пожарной безопасности».

Помещение, в котором ведется разработка программного обеспечения, классифицируется как офисное. К причинам возникновения пожара электрического характера относятся короткие замыкания, перегрузки, большие переходные сопротивления, искрение и электрические дуги, статическое электричество. Основным фактором, от которого зависит пожарная безопасность помещения, является автоматическое средство обнаружения пожаров. В помещении офиса должны быть также установлены: пожарные извещатели автоматической сигнализации, а также иметься воздушно-пенные, воздушно-эмульсионные, порошковые огнетушители.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 64   |

## Заключение

В ходе реализации данной работы было изучено внутренне устройство QR-кодов и назначение его основных элементов, а также устройство разных типов данных, которые можно закодировать в QR-код.

Так же были получены навыки разработки элементов графического интерфейса *WinForms*, разработки алгоритмов кодирования информации. Были получены навыки создания библиотек динамической компоновки и импортирование их методов.

Во время выполнения данной работы были разработаны: алгоритмы кодирования информации, алгоритмы генерации байт данных QR-кода, алгоритмы создания разных типов данных для QR-кода, методы импортирования функций из библиотеки динамической компоновки, алгоритмы отрисовки QR-кода. Так же было спроектировано оконное приложение для создания QR-кода.

В дальнейшем программу можно усовершенствовать, добавив новые кодировки данных, таких как числовая, символьно-числовая и кандзи. Так же возможно добавить новые способы отрисовки модулей QR-кода.

## Список использованных источников

1. *Elibrary.ru.* [Электронный ресурс]. URL: <https://www.elibrary.ru/item.asp?id=48441757> (дата обращения 23.04.2023)
2. *Habr.ru.* [Электронный ресурс]. URL: <https://habr.com/ru/articles/172525/> (дата обращения 23.04.2023)
3. Герберт, Шилдт С++. Базовый курс / Шилдт Герберт. - М.: Диалектика / Вильямс, 2022. - 564 с.
4. *Richard L. Weeks .NET Windows Forms Custom Controls; Sams - M., 2002. - 250 с.*
5. Смоленцев, Н. К. *MATLAB. Программирование на Visual C#, Borland JBuilder, VBA (+ CD-ROM)* / Н.К. Смоленцев. - М.: ДМК Пресс, 2018. - 456 с.
6. Лотка, Р. *C# и CSLA.NET Framework. Разработка бизнес-объектов* / Р. Лотка. - М.: Диалектика / Вильямс, 2017. - 555 с.

# ПриложениеА

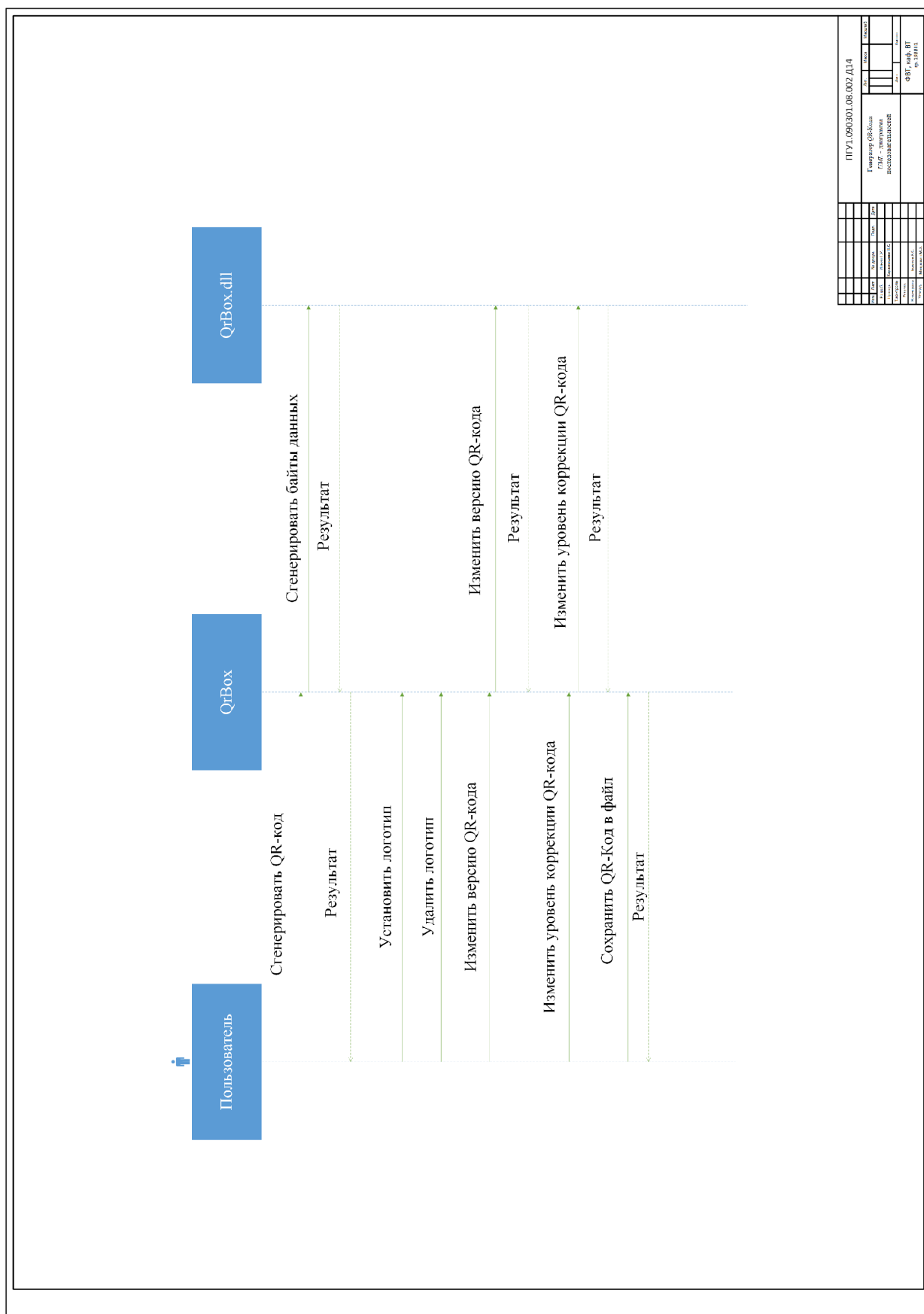
(Обязательное)

Генератор QR-кода

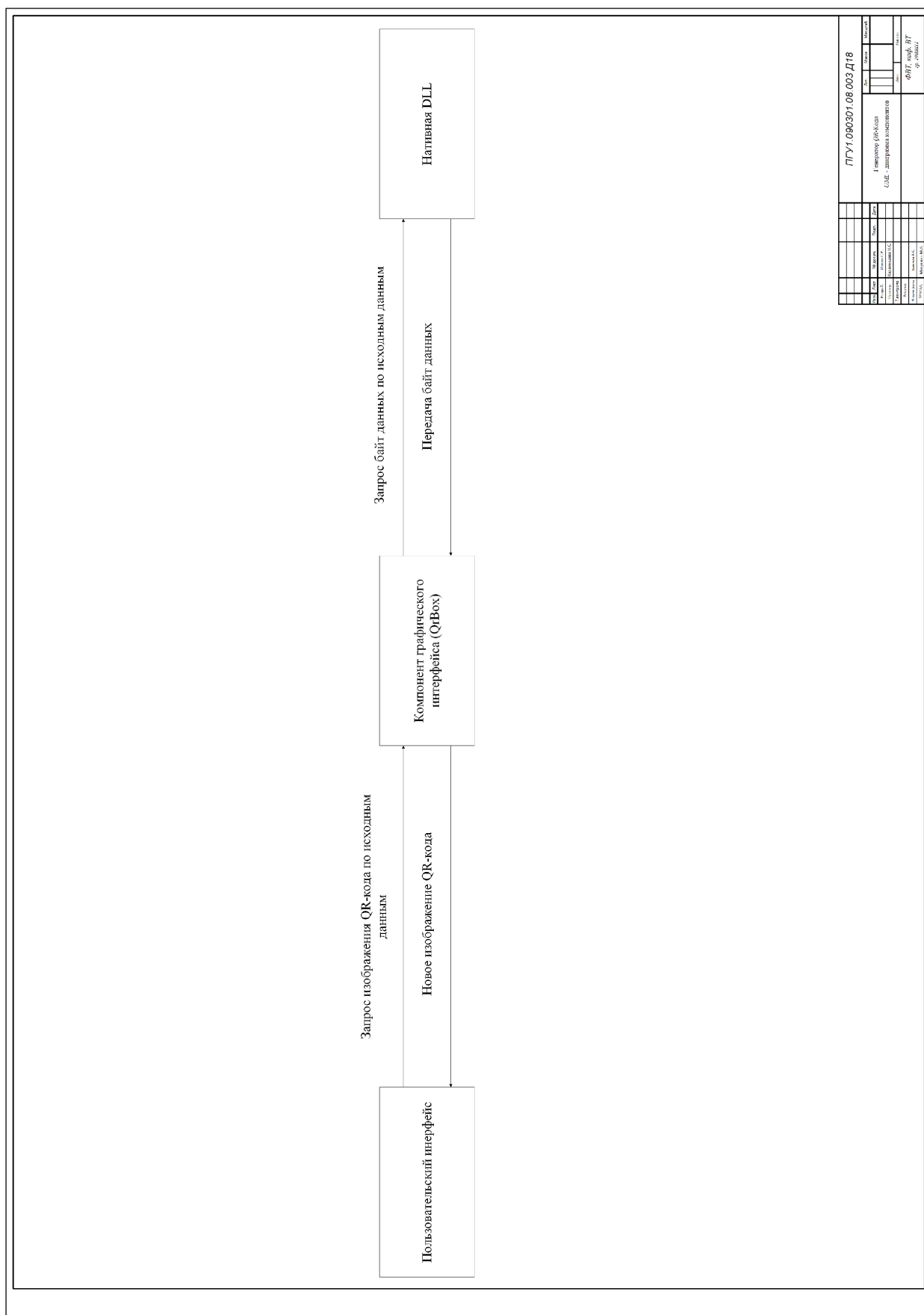
*UML*-диаграммы

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
|      |      |          |         |      |                       | 67   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

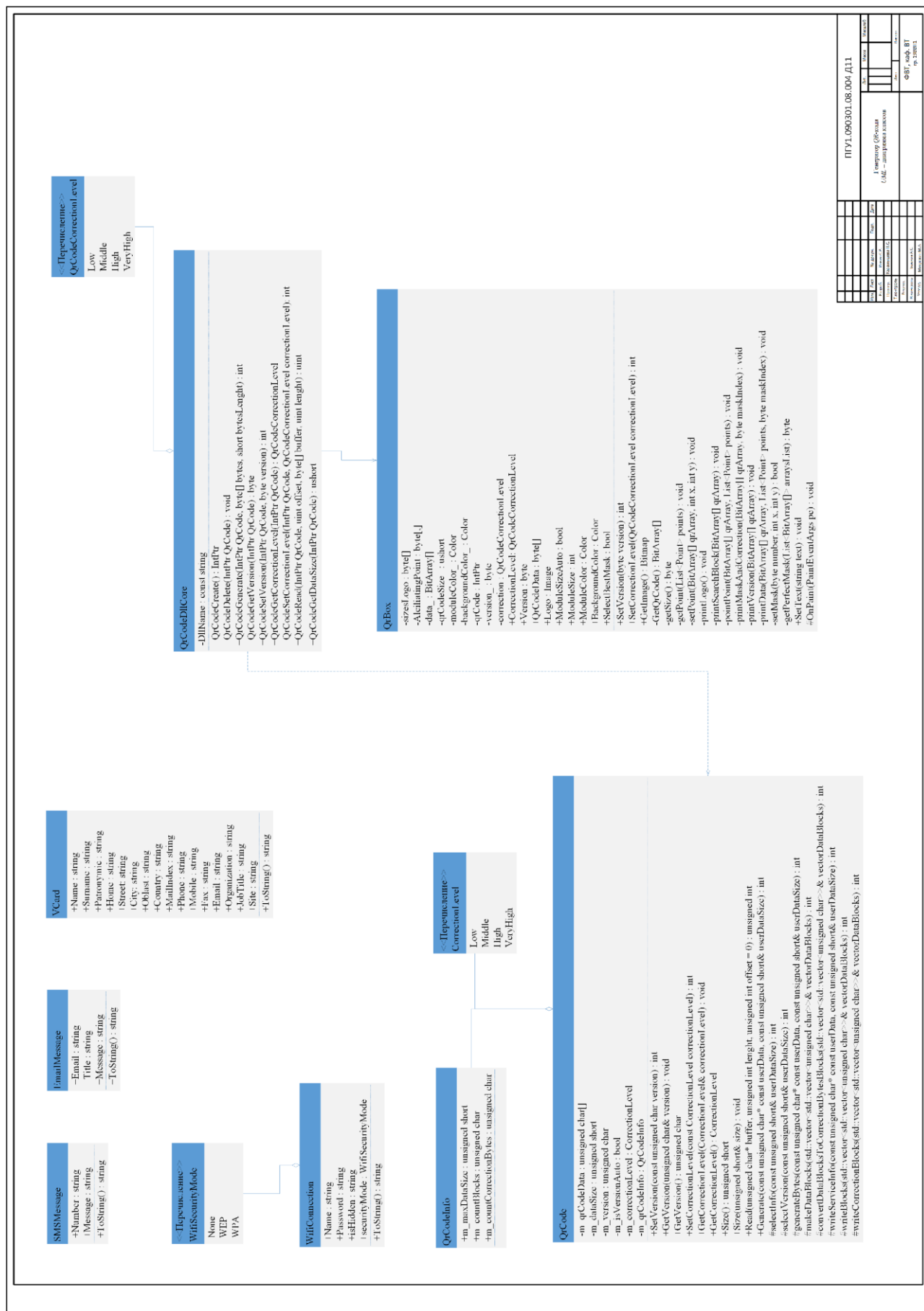
## Приложение А.1 – UML-Диаграмма последовательностей



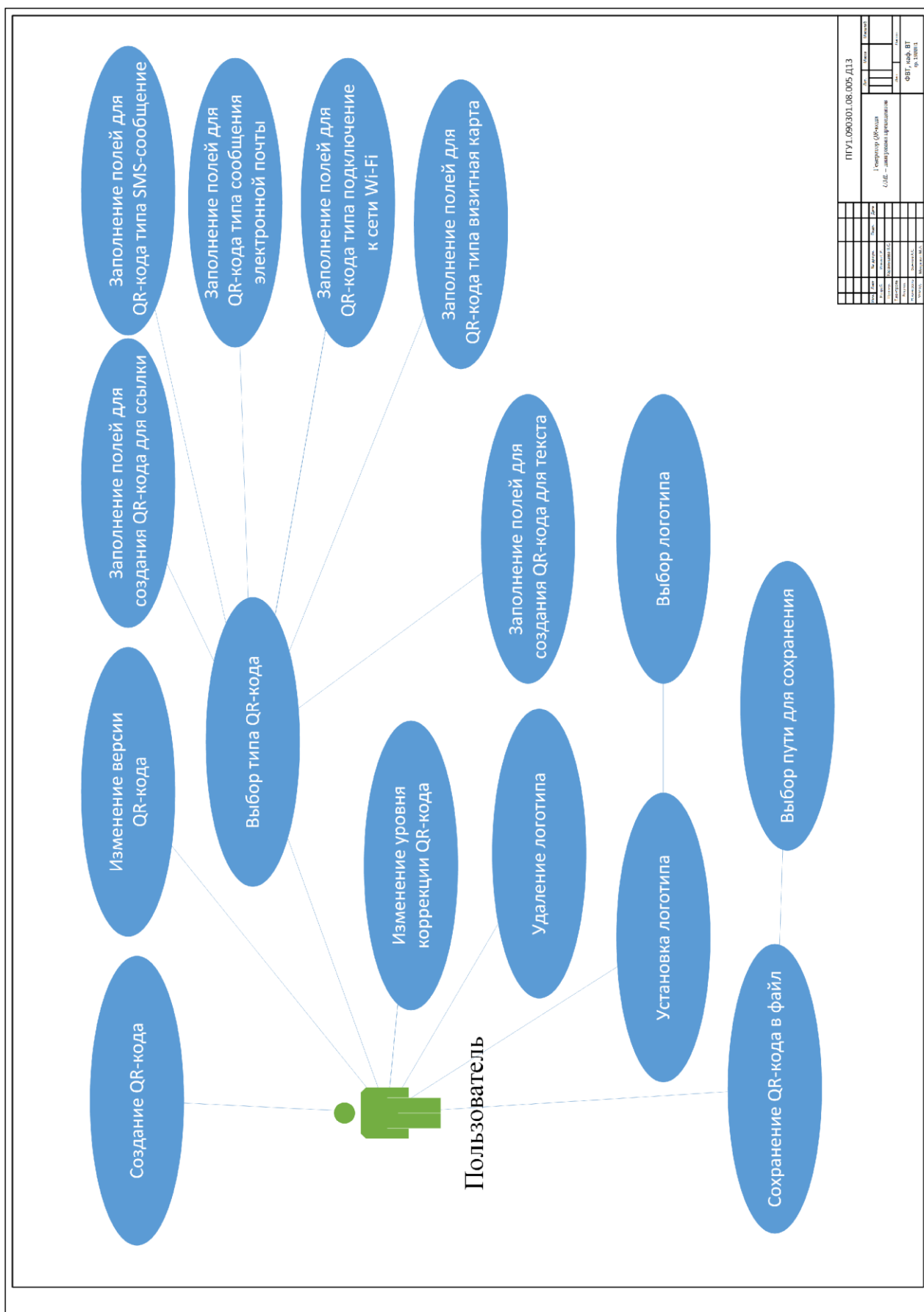
## Приложение А.2 – UML-Диаграмма компонентов



## Приложение А.3 – UML-Диаграмма классов



## Приложение А.4 – *UML*-Диаграмма прецедентов



# Приложение Б

(Обязательное)

Генератор QR-кода

Листинг

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
|      |      |          |         |      |                       | 72   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## Файл «CorrectionLevel.h»

```
#pragma once
enum class CorrectionLevel : unsigned char {
 Low,
 Middle,
 High,
 VeryHigh
}
```

## Файл «QrCodeInfo.h»

```
#pragma once
struct QrCodeInfo
{
 QrCodeInfo() noexcept;
 QrCodeInfo(const unsigned short maxDataSize,
 const unsigned char countBlocks,
 const unsigned char countCorrectionBytes) noexcept;

 explicit QrCodeInfo(const QrCodeInfo&) noexcept;
 explicit QrCodeInfo(QrCodeInfo&&) noexcept;

 QrCodeInfo& operator =(const QrCodeInfo&) noexcept;
 QrCodeInfo& operator =(QrCodeInfo&&) noexcept;

 virtual ~QrCodeInfo();

 unsigned short m_maxDataSize;
 unsigned char m_countBlocks;
 unsigned char m_countCorrectionBytes;
};

static const QrCodeInfog_qrCodeInfo[4][40]{
{
{19, 1, 7 }, {34, 1, 10 }, {55, 1, 15 }, {80, 1, 20 }, {108, 1,
26 },
{136, 2, 18 }, {156, 2, 20 }, {194, 2, 24 }, {232, 2, 30 }, {274, 4,
18 },
{324, 4, 20 }, {370, 4, 24 }, {428, 4, 26 }, {461, 4, 30 }, {523, 6,
22 },
{589, 6, 24 }, {647, 6, 28 }, {721, 6, 30 }, {795, 7, 28 }, {861, 8,
28 },
{932, 8, 28 }, {1006, 9, 28 }, {1094, 9, 30 }, {1174, 10, 30 }, {1276, 12,
26 },
{1370, 12, 28 }, {1468, 12, 30 }, {1531, 13, 30 }, {1631, 14, 30 }, {1735, 15,
30 },
{1843, 16, 30 }, {1955, 17, 30 }, {2071, 18, 30 }, {2191, 19, 30 }, {2306, 19,
30 },
{2434, 20, 30 }, {2566, 21, 30 }, {2702, 22, 30 }, {2812, 24, 30 }, {2956, 25,
30 }
},
{
{16, 1, 10 }, {28, 1, 16 }, {44, 1, 26 }, {64, 2, 18 }, {86, 2, 24
},

```

```

{108, 4, 16 }, {124, 4, 18 }, {154, 4, 22 }, {182, 5, 22 }, {216, 5,
26 },
{254, 5, 30 }, {290, 8, 22 }, {334, 9, 22 }, {365, 9, 24 }, {415, 10,
24 },
{453, 10, 28 }, {507, 11, 28 }, {563, 13, 26 }, {627, 14, 26 }, {669, 16,
26 },
{714, 17, 26 }, {782, 17, 28 }, {860, 18, 28 }, {914, 20, 28 }, {1000, 21,
28 },
{1062, 23, 28 }, {1128, 25, 28 }, {1193, 26, 28 }, {1267, 28, 28 }, {1373, 29,
28 },
{1455, 31, 28 }, {1541, 33, 28 }, {1631, 35, 28 }, {1725, 37, 28 }, {1812, 38,
28 },
{1914, 40, 28 }, {1992, 43, 28 }, {2102, 45, 28 }, {2216, 47, 28 }, {2334, 49,
28 }
},
{
{13, 1, 13 }, {22, 1, 22 }, {34, 2, 18 }, {48, 2, 26 }, {62, 4, 18
},
{76, 4, 24 }, {88, 6, 18 }, {110, 6, 22 }, {132, 8, 20 }, {154, 8,
24 },
{180, 8, 28 }, {206, 10, 26 }, {244, 12, 24 }, {261, 16, 20 }, {295, 12,
30 },
{325, 17, 24 }, {367, 16, 28 }, {397, 18, 28 }, {445, 21, 26 }, {485, 20,
30 },
{512, 23, 28 }, {568, 23, 30 }, {614, 25, 30 }, {664, 27, 30 }, {718, 29,
30 },
{754, 34, 28 }, {808, 34, 30 }, {871, 35, 30 }, {911, 38, 30 }, {985, 40,
30 },
{1033, 43, 30 }, {1115, 45, 30 }, {1171, 48, 30 }, {1231, 51, 30 }, {1286,
53, 30 },
{1354, 56, 30 }, {1426, 59, 30 }, {1502, 62, 30 }, {1582, 65, 30 }, {1666, 68,
30 }
},
{
{9, 1, 17 }, {16, 1, 28 }, {26, 2, 22 }, {36, 4, 16 }, {46, 4, 22
},
{60, 4, 28 }, {66, 5, 26 }, {86, 6, 26 }, {100, 8, 24 }, {122, 8,
28 },
{140, 11, 24 }, {158, 11, 28 }, {180, 16, 22 }, {197, 16, 24 }, {223, 18,
24 },
{253, 16, 30 }, {283, 19, 28 }, {313, 21, 28 }, {341, 25, 26 }, {385, 25,
28 },
{406, 25, 30 }, {442, 34, 24 }, {464, 30, 30 }, {514, 32, 30 }, {538, 35,
30 },
{596, 37, 30 }, {628, 40, 30 }, {661, 42, 30 }, {701, 45, 30 }, {745, 48,
30 },
{793, 51, 30 }, {845, 54, 30 }, {901, 57, 30 }, {961, 60, 30 }, {986, 63,
30 },
{1054, 66, 30 }, {1096, 70, 30 }, {1142, 74, 30 }, {1222, 77, 30 }, {1276, 81,
30 }
}
};

```

Файл «QrCodeInfo.cpp»

```
#include "pch.h"
```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 74   |

```

#include "QrCodeInfo.h"

QrCodeInfo::~QrCodeInfo() {}

QrCodeInfo::QrCodeInfo() noexcept
 : m_countBlocks()
 , m_countCorrectionBytes()
 , m_maxDataSize()
{}

QrCodeInfo::QrCodeInfo(const unsigned short maxDataSize,
 const unsigned char countBlocks,
 const unsigned char countCorrectionBytes) noexcept
 : m_maxDataSize(maxDataSize)
 , m_countBlocks(countBlocks)
 , m_countCorrectionBytes(countCorrectionBytes)
{}

QrCodeInfo::QrCodeInfo(const QrCodeInfo& other) noexcept
{
 m_countBlocks = other.m_countBlocks;
 m_countCorrectionBytes = other.m_countCorrectionBytes;
 m_maxDataSize = other.m_maxDataSize;
}

QrCodeInfo::QrCodeInfo(QrCodeInfo&& other) noexcept
{
 m_countBlocks = other.m_countBlocks;
 m_countCorrectionBytes = other.m_countCorrectionBytes;
 m_maxDataSize = other.m_maxDataSize;
 ::ZeroMemory(&other, sizeof(QrCodeInfo));
}

QrCodeInfo&QrCodeInfo::operator =(const QrCodeInfo& other)noexcept
{
 if (this != &other)
 {
 m_countBlocks = other.m_countBlocks;
 m_countCorrectionBytes = other.m_countCorrectionBytes;
 m_maxDataSize = other.m_maxDataSize;
 }
 return *this;
}

QrCodeInfo&QrCodeInfo::operator =(QrCodeInfo&& other) noexcept
{
 if (this != &other)
 {
 m_countBlocks = other.m_countBlocks;
 m_countCorrectionBytes = other.m_countCorrectionBytes;
 m_maxDataSize = other.m_maxDataSize;

 ::ZeroMemory(&other, sizeof(QrCodeInfo));
 }
 return *this;
}

```

# Файл «QrCode.h»

```
#pragma once

#include "CorrectionLevel.h"
#include "QrCodeInfo.h"

#include <vector>

#define QRCODE_MAXSIZE 3706

#define MAX_QRCODE_VERSION 40
#define MIN_QRCODE_VERSION 0
#define AUTO_QRCODE_VERSION 0

class QrCode
{
public:
 QrCode() noexcept;
 explicit QrCode(const QrCode& other) noexcept;
 explicit QrCode(QrCode&& other) noexcept;

 QrCode&
 operator =(const QrCode& other) noexcept;

 QrCode&
 operator =(QrCode&& other) noexcept;

 virtual ~QrCode();

 int
 SetVersion(const unsigned char version);

 void
 GetVersion(unsigned char& version);
 unsigned char
 GetVersion();

 int
 SetCorrectionLevel(const CorrectionLevel correctionLevel);

 void
 GetCorrectionLevel(CorrectionLevel& correctionLevel);
 CorrectionLevel
 GetCorrectionLevel();

 unsigned short
 Size();
 void
 Size(unsigned short& size);

 unsigned int
 Read(unsigned char* buffer, unsigned int lenght, unsigned int offset =
0);

 int
```

```

 Generate(const unsigned char* const userData, const unsigned
short&userDataSize);

protected:
 int
 selectInfo(const unsigned short&userDataSize);
 int
 selectVersion(const unsigned short&userDataSize);

 int
 generateBytes(const unsigned char* const userData, const unsigned
short&userDataSize);
 int
 makeDataBlocks(std::vector<std::vector<unsigned
char>>&vectorDataBlocks);
 int
 convertDataBlocksToCorrectionBytesBlocks(std::vector<std::vector<unsigne
d char>>&vectorDataBlocks);

 int
 writeServiceInfo(const unsigned char* const userData, const unsigned
short&userDataSize);
 int
 writeBlocks(std::vector<std::vector<unsigned char>>&vectorDataBlocks);
 int
 writeCorrectionBlocks(std::vector<std::vector<unsigned
char>>&vectorDataBlocks);

private:
 unsigned char m_qrCodeData[QRCODE_MAXSIZE];
 unsigned short m_dataSize;
 unsigned char m_version;
 bool m_isVersionAuto;
 CorrectionLevel m_correctionLevel;
 QrCodeInfo m_qrCodeInfo;
};

unsigned char* GetMultinomial(unsigned char countBytes);

const unsigned char gauleField[256]
{ 1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116, 232, 205, 135, 19, 38,
76, 152, 45, 90, 180, 117, 234, 201, 143, 3, 6, 12, 24, 48, 96, 192,
157, 39, 78, 156, 37, 74, 148, 53, 106, 212, 181, 119, 238, 193, 159, 35,
70, 140, 5, 10, 20, 40, 80, 160, 93, 186, 105, 210, 185, 111, 222, 161,
95, 190, 97, 194, 153, 47, 94, 188, 101, 202, 137, 15, 30, 60, 120, 240,
253, 231, 211, 187, 107, 214, 177, 127, 254, 225, 223, 163, 91, 182, 113, 226,
217, 175, 67, 134, 17, 34, 68, 136, 13, 26, 52, 104, 208, 189, 103, 206,
129, 31, 62, 124, 248, 237, 199, 147, 59, 118, 236, 197, 151, 51, 102, 204,
133, 23, 46, 92, 184, 109, 218, 169, 79, 158, 33, 66, 132, 21, 42, 84,
168, 77, 154, 41, 82, 164, 85, 170, 73, 146, 57, 114, 228, 213, 183, 115,
230, 209, 191, 99, 198, 145, 63, 126, 252, 229, 215, 179, 123, 246, 241, 255,
227, 219, 171, 75, 150, 49, 98, 196, 149, 55, 110, 220, 165, 87, 174, 65,
130, 25, 50, 100, 200, 141, 7, 14, 28, 56, 112, 224, 221, 167, 83, 166,
81, 162, 89, 178, 121, 242, 249, 239, 195, 155, 43, 86, 172, 69, 138, 9,
18, 36, 72, 144, 61, 122, 244, 245, 247, 243, 251, 235, 203, 139, 11, 22,
44, 88, 176, 125, 250, 233, 207, 131, 27, 54, 108, 216, 173, 71, 142, 1 };

```

```

const unsigned char reverseGauleField[256]
{ 0, 0, 1, 25, 2, 50, 26, 198, 3, 223, 51, 238, 27, 104, 199, 75,
4, 100, 224, 14, 52, 141, 239, 129, 28, 193, 105, 248, 200, 8, 76, 113,
5, 138, 101, 47, 225, 36, 15, 33, 53, 147, 142, 218, 240, 18, 130, 69,
29, 181, 194, 125, 106, 39, 249, 185, 201, 154, 9, 120, 77, 228, 114, 166,
6, 191, 139, 98, 102, 221, 48, 253, 226, 152, 37, 179, 16, 145, 34, 136,
54, 208, 148, 206, 143, 150, 219, 189, 241, 210, 19, 92, 131, 56, 70, 64,
30, 66, 182, 163, 195, 72, 126, 110, 107, 58, 40, 84, 250, 133, 186, 61,
202, 94, 155, 159, 10, 21, 121, 43, 78, 212, 229, 172, 115, 243, 167, 87,
7, 112, 192, 247, 140, 128, 99, 13, 103, 74, 222, 237, 49, 197, 254, 24,
227, 165, 153, 119, 38, 184, 180, 124, 17, 68, 146, 217, 35, 32, 137, 46,
55, 63, 209, 91, 149, 188, 207, 205, 144, 135, 151, 178, 220, 252, 190, 97,
242, 86, 211, 171, 20, 42, 93, 158, 132, 60, 57, 83, 71, 109, 65, 162,
31, 45, 67, 216, 183, 123, 164, 118, 196, 23, 73, 236, 127, 12, 111, 246,
108, 161, 59, 82, 41, 157, 85, 170, 251, 96, 134, 177, 187, 204, 62, 90,
203, 89, 95, 176, 156, 169, 160, 81, 11, 245, 22, 235, 122, 117, 44, 215,
79, 174, 213, 233, 230, 231, 173, 232, 116, 214, 244, 234, 168, 80, 88, 175 };

```

### Файл «QrCode.cpp»

```

#include "pch.h"
#include "QrCode.h"

QrCode::
QrCode() noexcept
 : m_dataSize()
 , m_version()
 , m_isVersionAuto(true)
 , m_correctionLevel()
 , m_qrCodeInfo()
{
 ::ZeroMemory(&m_qrCodeData, sizeof(m_qrCodeData) /
sizeof(m_qrCodeData[0]));
}

QrCode::
QrCode(const QrCode& other) noexcept
 : m_dataSize(other.m_dataSize)
 , m_version(other.m_version)
 , m_isVersionAuto(other.m_isVersionAuto)
 , m_correctionLevel(other.m_correctionLevel)
 , m_qrCodeInfo(other.m_qrCodeInfo)
{
 ::memcpy(m_qrCodeData, other.m_qrCodeData, m_dataSize);
}

QrCode::
QrCode(QrCode&& other) noexcept
 : m_dataSize(other.m_dataSize)
 , m_version(other.m_version)
 , m_isVersionAuto(other.m_isVersionAuto)

```

```

 , m_correctionLevel(other.m_correctionLevel)
 , m_qrCodeInfo(other.m_qrCodeInfo)
 {
 ::memcpy(m_qrCodeData, other.m_qrCodeData, m_dataSize);
 ::ZeroMemory(&other, sizeof(other));
 }

QrCode&QrCode::
operator =(const QrCode& other) noexcept
{
 if (this != &other)
 {
 m_dataSize = other.m_dataSize;
 m_version = other.m_version;
 m_isVersionAuto = other.m_isVersionAuto;
 m_correctionLevel = other.m_correctionLevel;
 m_qrCodeInfo = other.m_qrCodeInfo;
 ::memcpy(m_qrCodeData, other.m_qrCodeData, m_dataSize);
 }
 return *this;
}

QrCode&QrCode::
operator =(QrCode&& other) noexcept
{
 if (this != &other)
 {
 m_dataSize = other.m_dataSize;
 m_version = other.m_version;
 m_isVersionAuto = other.m_isVersionAuto;
 m_correctionLevel = other.m_correctionLevel;
 m_qrCodeInfo = other.m_qrCodeInfo;
 ::memcpy(m_qrCodeData, other.m_qrCodeData, m_dataSize);

 ::ZeroMemory(&other, sizeof(other));
 }
 return *this;
}

QrCode::
~QrCode() {}

unsigned int QrCode::
Read(unsigned char* buffer, unsigned int lenght, unsigned int offset)
{
 if (m_dataSize<= offset)
 return 0;

 if (m_dataSize - offset <lenght)
 lenght = m_dataSize - offset;

 ::memcpy(buffer, m_qrCodeData + offset, lenght);
 return lenght;
}

```

```

int QrCode::
SetVersion(const unsigned char version) {
 if (version > MAX_QRCODE_VERSION)
 return ERROR_BAD_ARGUMENTS;

 m_isVersionAuto = version == 0 ? true : false;

 m_version = version;
 return NOERROR;
}

void QrCode::
GetVersion(unsigned char& version) {
 version = m_version;
}

unsigned char QrCode::
GetVersion() {
 return m_version;
}

int QrCode::
SetCorrectionLevel(const CorrectionLevelcorrectionLevel) {
 if ((CorrectionLevel::Low > correctionLevel) ||
 (correctionLevel > CorrectionLevel::VeryHigh))
 return ERROR_BAD_ARGUMENTS;

 m_correctionLevel = correctionLevel;
 return NOERROR;
}

void QrCode::
GetCorrectionLevel(CorrectionLevel&correctionLevel) {
 correctionLevel = m_correctionLevel;
}

CorrectionLevelQrCode::
GetCorrectionLevel() {
 return m_correctionLevel;
}

unsigned short QrCode::
Size() {
 return m_dataSize;
}

void QrCode::
Size(unsigned short& size) {
 size = m_dataSize;
}

int QrCode::
Generate(const unsigned char* const userData, const unsigned
short&userDataSize)

```

```

{
 if (userData == nullptr)
 return ERROR_INVALID_DATA;

 int result = QRCode::selectInfo(userDataSize);
 if (result != NOERROR)
 return result;

 result = QRCode::generateBytes(userData, userDataSize);
 if (result != NOERROR)
 return result;

 return result;
}

unsigned char*
GetMultinomial(unsigned char countBytes)
{
 switch (countBytes)
 {
 case 7:
 return new unsigned char[7]{ 87, 229, 146, 149, 238, 102, 21 };

 case 10:
 return new unsigned char[10]{ 251, 67, 46, 61, 118, 70, 64, 94, 32,
45 };

 case 13:
 return new unsigned char[13]{ 74, 152, 176, 100, 86, 100, 106, 104,
130, 218, 206, 140, 78 };

 case 15:
 return new unsigned char[15]{ 8, 183, 61, 91, 202, 37, 51, 58, 58,
237, 140, 124, 5, 99, 105 };

 case 16:
 return new unsigned char[16]{ 120, 104, 107, 109, 102, 161, 76, 3,
91, 191, 147, 169, 182, 194, 225, 120 };

 case 17:
 return new unsigned char[17]{ 43, 139, 206, 78, 43, 239, 123, 206,
214, 147, 24, 99, 150, 39, 243, 163, 136 };

 case 18:
 return new unsigned char[18]{ 215, 234, 158, 94, 184, 97, 118, 170,
79, 187, 152, 148, 252, 179, 5, 98, 96, 153 };

 case 20:
 return new unsigned char[20]{ 17, 60, 79, 50, 61, 163, 26, 187,
202, 180, 221, 225, 83, 239, 156, 164, 212, 212, 188, 190 };

 case 22:
 return new unsigned char[22]{ 210, 171, 247, 242, 93, 230, 14, 109,
221, 53, 200, 74, 8, 172, 98, 80, 219, 134, 160, 105, 165, 231 };

 case 24:

```

```

 return new unsigned char[24]{ 229, 121, 135, 48, 211, 117, 251,
126, 159, 180, 169, 152, 192, 226, 228, 218, 111, 0, 117, 232, 87, 96, 227, 21
};

 case 26:
 return new unsigned char[26]{ 173, 125, 158, 2, 103, 182, 118, 17,
145, 201, 111, 28, 165, 53, 161, 21, 245, 142, 13, 102, 48, 227, 153, 145,
218, 70 };

 case 28:
 return new unsigned char[28]{ 168, 223, 200, 104, 224, 234, 108,
180, 110, 190, 195, 147, 205, 27, 232, 201, 21, 43, 245, 87, 42, 195, 212,
119, 242, 37, 9, 123 };

 case 30:
 return new unsigned char[30]{ 41, 173, 145, 152, 216, 31, 179, 182,
50, 48, 110, 86, 239, 96, 222, 125, 42, 173, 226, 193, 224, 130, 156, 37, 251,
216, 238, 40, 192, 180 };

 default:
 return nullptr;
 }
}

int QrCode::
selectVersion(const unsigned short&userDataSize)
{
 for (unsigned char i = 0; i < 40; ++i) {
 if (userDataSize <= g_qrCodeInfo[(unsigned
char)m_correctionLevel][i].m_maxDataSize) {
 m_version = i + 1;
 return NOERROR;
 }
 }
 return ERROR_BAD_ARGUMENTS;
}

int QrCode::
selectInfo(const unsigned short&userDataSize)
{
 if (m_isVersionAuto)
 {
 if (QrCode::selectVersion(userDataSize) != NOERROR)
 return ERROR_BAD_ARGUMENTS;
 }

 if (userDataSize > g_qrCodeInfo[(unsigned
char)m_correctionLevel][m_version - 1].m_maxDataSize)
 return ERROR_BAD_ARGUMENTS;

 m_qrCodeInfo = g_qrCodeInfo[(unsigned char)m_correctionLevel][m_version
- 1];
 return NOERROR;
}

int QrCode::

```

```

writeServiceInfo(const unsigned char* const userData, const unsigned
short&userDataSize)
{
 if (userData == nullptr)
 return ERROR_INVALID_DATA;

 unsigned char serviceHeaderSize;

 //Методкодирования (побайтовый - 0100)
 m_qrCodeData[0] = 0x40;

 if (m_version < 10)
 {
 //В этом случае информация о количестве данных занимает 8 бит
 m_qrCodeData[0] |= userDataSize >> 4;
 m_qrCodeData[1] = userDataSize << 4;

 serviceHeaderSize = 2;
 }
 else
 {
 //В этом случае информация о количестве данных занимает 16 бит
 m_qrCodeData[0] |= userDataSize >> 12;
 m_qrCodeData[1] = userDataSize >> 4;
 m_qrCodeData[2] = userDataSize << 4;

 serviceHeaderSize = 3;
 }

 //Добавляем байты информации
 ::memcpy(m_qrCodeData + serviceHeaderSize, userData, userDataSize);

 //Сдвигаем байты информации на пустные 4 бита
 for (unsigned short i = serviceHeaderSize; i < userDataSize +
serviceHeaderSize; ++i)
 {
 m_qrCodeData[i - 1] += m_qrCodeData[i] >> 4;
 m_qrCodeData[i] <= 4;
 }

 //Заполняем оставшие байты в поле служебной информации
 bool check = true;
 for (unsigned short i = userDataSize + serviceHeaderSize;
i < m_qrCodeInfo.m_maxDataSize; ++i)
 {
 if (check) {
 m_qrCodeData[i] = 0xec;
 }
 else {
 m_qrCodeData[i] = 0x11;
 }
 ~check;
 }

 return NOERROR;
}

```

```

int QrCode::
makeDataBlocks(std::vector<std::vector<unsigned char>>&vectorDataBlocks)
{
 unsigned short countBytesToBlock = m_qrCodeInfo.m_maxDataSize /
m_qrCodeInfo.m_countBlocks;
 unsigned short remainingBytesToBlock = m_qrCodeInfo.m_maxDataSize %
m_qrCodeInfo.m_countBlocks;

 unsigned char* qrCodeData = m_qrCodeData;

 for (unsigned short i = 0; i<m_qrCodeInfo.m_countBlocks -
remainingBytesToBlock; ++i)
 {
 std::vector<unsigned char>vectorBlock;
 vectorDataBlocks.push_back(vectorBlock);

 vectorDataBlocks[i].reserve(countBytesToBlock);
 vectorDataBlocks[i].resize(countBytesToBlock);

 ::memcpy(vectorDataBlocks[i].data(), qrCodeData,
countBytesToBlock);
 qrCodeData += countBytesToBlock;
 }

 countBytesToBlock++;
 for (unsigned short i = m_qrCodeInfo.m_countBlocks -
remainingBytesToBlock; i<m_qrCodeInfo.m_countBlocks; ++i)
 {
 std::vector<unsigned char>vectorBlock;
 vectorDataBlocks.push_back(vectorBlock);

 vectorDataBlocks[i].reserve(countBytesToBlock);
 vectorDataBlocks[i].resize(countBytesToBlock);

 ::memcpy(vectorDataBlocks[i].data(), qrCodeData,
countBytesToBlock);
 qrCodeData += countBytesToBlock;
 }

 return NOERROR;
}

int QrCode::
writeBlocks(std::vector<std::vector<unsigned char>>&vectorDataBlocks)
{
 unsigned short remainingBytesToBlock = m_qrCodeInfo.m_maxDataSize %
m_qrCodeInfo.m_countBlocks;

 for (size_t i = 0; i<vectorDataBlocks[0].size(); ++i) {
 for (size_t j = 0; j <vectorDataBlocks.size(); ++j) {
 m_qrCodeData[m_dataSize++] = vectorDataBlocks[j][i];
 }
 }
}

```

```

 for (size_ti = vectorDataBlocks.size() - remainingBytesToBlock;
i<vectorDataBlocks.size(); ++i) {
 m_qrCodeData[m_dataSize++] =
vectorDataBlocks[i][vectorDataBlocks[i].size() - 1];
 }

 return NOERROR;
 }

int QrCode::
convertDataBlocksToCorrectionBytesBlocks(std::vector<std::vector<unsigned
char>>&vectorDataBlocks)
{
 unsigned char* multinominal =
GetMultinomial(m_qrCodeInfo.m_countCorrectionBytes);

 for (auto&dataBlocks :vectorDataBlocks)
 {
 unsigned char countBytesToBlock = dataBlocks.size();

 if (dataBlocks.size() <m_qrCodeInfo.m_countCorrectionBytes) {
 dataBlocks.reserve(m_qrCodeInfo.m_countCorrectionBytes);

 ::memset(dataBlocks.data() + dataBlocks.size(),
0, m_qrCodeInfo.m_countCorrectionBytes -
dataBlocks.size());

 dataBlocks.resize(m_qrCodeInfo.m_countCorrectionBytes);
 }

 for (unsigned char j = 0; j <countBytesToBlock; ++j)
 {
 unsigned char A, B, C;

 A = dataBlocks[0];

 //Сдвигаембайтывправо
 ::memcpy(dataBlocks.data(), dataBlocks.data() + 1,
dataBlocks.size() - 1);

 //Присваиваемпоследнемубайту 0
 dataBlocks[dataBlocks.size() - 1] = 0;

 if (A == 0)
 continue;

 B = reverseGauleField[A];

 for (int k = 0; k <m_qrCodeInfo.m_countCorrectionBytes; ++k)
 {
 C = (multinomial[k] + B) % 255;

 dataBlocks[k] ^= gauleField[C];
 }
 }
 }
}

```

```

 dataBlocks.resize(m_qrCodeInfo.m_countCorrectionBytes);
 }

 delete[] multinominal;
 return NOERROR;
}

int QrCode::
writeCorrectionBlocks(std::vector<std::vector<unsigned
char>>&vectorDataBlocks)
{
 for (unsigned short i = 0; i<m_qrCodeInfo.m_countCorrectionBytes; ++i) {
 for (unsigned short j = 0; j <vectorDataBlocks.size(); ++j) {
 m_qrCodeData[m_dataSize++] = vectorDataBlocks[j][i];
 }
 }

 return NOERROR;
}

int QrCode::
generateBytes(const unsigned char* const userData, const unsigned
short&userDataSize)
{
 if (userData == nullptr)
 return ERROR_INVALID_DATA;

 int result = QrCode::writeServiceInfo(userData, userDataSize);
 if (result != NOERROR)
 return result;

 std::vector<std::vector<unsigned char>>vectorDataBlocks;

 result = QrCode::makeDataBlocks(vectorDataBlocks);
 if (result != NOERROR)
 return result;

 m_dataSize = 0;
 result = QrCode::writeBlocks(vectorDataBlocks);
 if (result != NOERROR)
 return result;

 result =
QrCode::convertDataBlocksToCorrectionBytesBlocks(vectorDataBlocks);
 if (result != NOERROR)
 return result;

 result = QrCode::writeCorrectionBlocks(vectorDataBlocks);
 if (result != NOERROR)
 return result;

 return result;
}

```

# Файл «QrCodeDllExports.h»

```
#pragma once

#include "QrCode.h"

#ifdef QRCODEDLL_EXPORTS
 #define QRCODEDLL_API __declspec(dllexport)
#else
 #define QRCODEDLL_API __declspec(dllimport)
#endif

#ifdef __cplusplus
extern "C" {
#endif

 QRCODEDLL_API
 QrCode*
 QrCodeCreate();

 QRCODEDLL_API
 void
 QrCodeDelete(QrCode* newQr);

 QRCODEDLL_API
 int
 QrCodeGenerate(QrCode* newQr, const unsigned char* text, unsigned short
 lenght);

 QRCODEDLL_API
 unsigned char
 QrCodeGetVersion(QrCode* newQr);

 QRCODEDLL_API
 int
 QrCodeSetVersion(QrCode* newQr, unsigned char version);

 QRCODEDLL_API
 CorrectionLevel
 QrCodeGetCorrectionLevel(QrCode* newQr);

 QRCODEDLL_API
 int
 QrCodeSetCorrectionLevel(QrCode* newQr, CorrectionLevel correction);

 QRCODEDLL_API
 unsigned int
 QrCodeRead(QrCode* newQr, unsigned int offset, unsigned char* buffer,
 unsigned int lenght);

 QRCODEDLL_API
 unsigned short
 QrCodeGetDataSize(QrCode* newQr);

#ifdef __cplusplus
}

```

#endif

Файл «QrCodeDllExports.cpp»

```
#include "pch.h"
#include "QrCodeDllExports.h"
```

```
QrCode*
QrCodeCreate() {
 return new QrCode();
}
```

```
void
QrCodeDelete(QrCode* newQr) {
 delete newQr;
}
```

```
int
QrCodeGenerate(QrCode* newQr, const unsigned char* text, unsigned short
lenght) {
 return newQr->Generate(text, lenght);
}
```

```
unsigned char
QrCodeGetVersion(QrCode* newQr) {
 return newQr->GetVersion();
}
```

```
int
QrCodeSetVersion(QrCode* newQr, unsigned char version) {
 return newQr->SetVersion(version);
}
```

```
CorrectionLevel
QrCodeGetCorrectionLevel(QrCode* newQr) {
 return newQr->GetCorrectionLevel();
}
```

```
int
QrCodeSetCorrectionLevel(QrCode* newQr, CorrectionLevel correction) {
 return newQr->SetCorrectionLevel(correction);
}
```

```
unsigned int
QrCodeRead(QrCode* newQr, unsigned int offset, unsigned char* buffer, unsigned
int size) {
 return newQr->Read(buffer, size, offset);
}
```

```
unsigned short
QrCodeGetDataSize(QrCode* newQr) {
 return newQr->Size();
}
```

# Файл «QrCodeDllCore.cs»

```
using System.Runtime.InteropServices;

namespace QrGen
{
 [Flags]
 public enum QrCodeCorrectionLevel : byte
 {
 MIN,
 MIDDLE,
 HIGH,
 VERY_HIGH
 }

 internal class QrCodeDllCore
 {
 private const string DllName = "QrCode.dll";

 [DllImport(DllName)]
 public static extern IntPtr QrCodeCreate();

 [DllImport(DllName)]
 public static extern void QrCodeDelete(IntPtr QrCode);

 [DllImport(DllName)]
 public static extern int QrCodeGenerate(IntPtr QrCode, byte[] bytes,
short bytesLenght);

 [DllImport(DllName)]
 public static extern byte QrCodeGetVersion(IntPtr QrCode);

 [DllImport(DllName)]
 public static extern int QrCodeSetVersion(IntPtr QrCode, byte version);

 [DllImport(DllName)]
 public static extern
QrCodeCorrectionLevel QrCodeGetCorrectionLevel(IntPtr QrCode);

 [DllImport(DllName)]
 public static extern int QrCodeSetCorrectionLevel(IntPtr QrCode,
QrCodeCorrectionLevel correctionLevel);

 [DllImport(DllName)]
 public static extern uint QrCodeRead(IntPtr QrCode, uint offset, byte[]
buffer, uint lenght);

 [DllImport(DllName)]
 public static extern ushort QrCodeGetDataSize(IntPtr QrCode);
 }
}
```

# Файл «QrBox.cs»

```
using System.Collections;
```

```

using System.Text;

namespace QrGen
{
 public struct SMSMessage
 {
 public string Number;
 public string Message;

 public override string ToString()
 {
 var builder = new StringBuilder();
 builder.Append("smsto:").Append(Number).Append(":")
 .AppendLine(Message);

 return builder.ToString();
 }
 }

 public struct EmailMessage
 {
 public string Email;
 public string Title;
 public string Message;

 public override string ToString()
 {
 var builder = new StringBuilder();
 builder.Append("MATMSG:TO:").Append(Email).Append(";")
 .Append(Title).Append(";").Append(Message).AppendLine(";;");

 return builder.ToString();
 }
 }

 public enum WifiSecurityMode

```

```

{
 NONE,
 WEP,
 WPA
}

public struct WifiConnection
{
 public string Name;
 public string Password;
 public bool isHidden;

 public WifiSecurityModesecurityMode;

 public override string ToString()
 {
 var builder = new StringBuilder();
builder.Append("WIFI:T:");

 switch (securityMode)
 {
 case WifiSecurityMode.NONE:
builder.Append("nopass;");
 break;

 case WifiSecurityMode.WPA:
builder.Append("WPA;");
 break;

 case WifiSecurityMode.WEP:
builder.Append("WEP;");
 break;

 default:
builder.Append("nopass;");
 break;
 }
 }
}

```

```
builder.Append("S:").Append(Name).Append(";")
.Append("P:").Append(Password).Append(";")
.Append("H:").Append(isHidden).AppendLine(";;");
```

```
 return builder.ToString();
 }
}
```

```
public struct VCard
```

```
{
```

```
 public string Name;
 public string Surname;
 public string Patronymic;
```

```
 public string Home;
 public string Street;
 public string City;
 public string Oblast;
 public string Country;
 public string MailIndex;
```

```
 public string Phone;
 public string Mobile;
 public string Fax;
```

```
 public string Email;
```

```
 public string Organization;
 public string JobTitle;
```

```
 public string Site;
```

```
 public override string ToString()
 {
```

```
 var builder = new StringBuilder();
 builder.AppendLine("BEGIN:VCARD");
```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 92   |

```

builder.AppendLine("VERSION:3.0");

 //Name
builder.Append("N:").Append(Surname).Append(" ;")
.Append(Name).Append(" ;").AppendLine(Patronymic);

 //Full name
builder.Append("N:").Append(Name).Append(" ").Append(Patronymic)
.Append(" ").AppendLine(Surname);

 //Address
builder.Append("ADR;TYPE = intl, postal, parcel, work;;;").Append(Home)
.Append(" ").Append(Street).Append(";").Append(City).Append(";")
.Append(Oblast).Append(";").Append(MailIndex).Append(";").AppendLine(Country);

 //Other data
builder.Append("ORG:").AppendLine(Organization);
builder.Append("TITLE:").AppendLine(JobTitle);
builder.Append("EMAIL;PREF;INTERNET:").AppendLine(Email);
builder.Append("URL:").AppendLine(Site);
builder.Append("TEL;WORK;VOICE:").AppendLine(Phone);
builder.Append("TEL;CELL;VOICE:").AppendLine(Mobile);
builder.Append("TEL;FAX:").AppendLine(Fax);

builder.AppendLine("End:VCARD");
 return builder.ToString();
 }
};

internal class QrBox :PictureBox
{
 public QrBox()
 {
 QrCode = QrCodeDllCore.QrCodeCreate();
 }
}

```

```

 protected new void Dispose()
 {
base.Dispose();
QrCodeDllCore.QrCodeDelete(QrCode);
 }

 private readonlybyte[] sizesLogo = { 8, 9, 12, 14, 16,
 18, 20, 22, 23, 27,
 30, 31, 22, 34, 32,
 35, 39, 33, 42, 39,
 40, 51, 55, 58, 61,
 62, 66, 62, 69, 67,
 71, 71, 75, 75, 75,
 70, 72, 77, 61, 91 };

 private readonlybyte[,] AlciliatingPoint = { { 0, 0, 0, 0, 0, 0, 0, 0 },
{ 18, 0, 0, 0, 0, 0, 0, 0 }, { 22, 0, 0, 0, 0, 0, 0, 0 }, { 26, 0, 0, 0, 0, 0, 0, 0 },
{ 30, 0, 0, 0, 0, 0, 0, 0 }, { 34, 0, 0, 0, 0, 0, 0, 0 }, { 6, 22, 38, 0, 0, 0, 0, 0 },
{ 6, 24, 42, 0, 0, 0, 0, 0 }, { 6, 26, 46, 0, 0, 0, 0, 0 }, { 6, 28, 50, 0, 0, 0, 0, 0 },
{ 6, 30, 54, 0, 0, 0, 0, 0 }, { 6, 32, 58, 0, 0, 0, 0, 0 }, { 6, 34, 62, 0, 0, 0, 0, 0 },
{ 6, 26, 46, 66, 0, 0, 0, 0 }, { 6, 26, 48, 70, 0, 0, 0, 0 }, { 6, 26, 50, 74, 0, 0, 0, 0 },
{ 6, 30, 54, 78, 0, 0, 0, 0 }, { 6, 30, 56, 82, 0, 0, 0, 0 }, { 6, 30, 58, 86, 0, 0, 0, 0 },
{ 6, 34, 62, 90, 0, 0, 0, 0 }, { 6, 28, 50, 72, 94, 0, 0, 0 }, { 6, 26, 50, 74, 98, 0, 0, 0 },
{ 6, 30, 54, 78, 102, 0, 0, 0 }, { 6, 28, 54, 80, 106, 0, 0, 0 },
{ 6, 32, 58, 84, 110, 0, 0, 0 }, { 6, 30, 58, 86, 114, 0, 0, 0 },
{ 6, 34, 62, 90, 118, 0, 0, 0 }, { 6, 26, 50, 74, 98, 122, 0, 0 },
{ 6, 30, 54, 78, 102, 126, 0, 0 }, { 6, 26, 52, 78, 104, 130, 0, 0 },
{ 6, 30, 56, 82, 108, 134, 0, 0 }, { 6, 34, 60, 86, 112, 138, 0, 0 },
{ 6, 30, 58, 86, 114, 142, 0, 0 }, { 6, 34, 62, 90, 118, 146, 0, 0 },
{ 6, 30, 54, 78, 102, 126, 150, 0 }, { 6, 24, 50, 76, 102, 128, 154, 0 },

```

```

{ 6, 28, 54, 80, 106, 132, 158, 0 }, { 6, 32, 58, 84, 110, 136, 162, 0 },
{ 6, 26, 54, 82, 110, 138, 166, 0 }, { 6, 30, 58, 86, 114, 142, 170, 0 } };

private BitArray[] data_;

private ushort qrCodeSize_;

private int moduleSize_;
private Color moduleColor_ = Color.Black;
private Color backgroundColor_ = Color.White;
private Bitmap bmp_;

private IntPtr qrCode;

private byte version_ = 0;
private QrCodeCorrectionLevel correction_ = QrCodeCorrectionLevel.MIN;

public QrCodeCorrectionLevelCorrectionLevel{ get => correction_;
private set { } }

public byte Version { get => version_; private set { } }
public byte[] QrCodeData { get; set; }
public Image Logo { get; set; } = null;
public bool ModuleSizeAuto{ get; set; } = true;
public int ModuleSize{ get => moduleSize_; set { moduleSize_ = value;
ModuleSizeAuto = false; } }
public Color ModuleColor{ get => moduleColor_; set => moduleColor_ =
value; }
public Color BackgroundColor{ get => backgroundColor_; set
=> backgroundColor_ = value; }
public bool SelectBestMask{ get; set; } = false;

public int SetVersion(byte version)
{
 int result = 0;

```

```

 result = QrCodeDllCore.QrCodeSetVersion(QrCode, version);
if(result != 0)
 return result;

 version_ = version;
 return result;
}

public int SetCorrectionLevel(QrCodeCorrectionLevelcorrectionLevel)
{
 int result = 0;
 result = QrCodeDllCore.QrCodeSetCorrectionLevel(QrCode,
correctionLevel);
 if (result != 0)
 return result;

 correction_ = correctionLevel;
 return result;
}

private void printLogo()
{
if (Logo != null)
 {
int maxWidth = Logo.Width<moduleSize_ ?moduleSize_ : Logo.Width;
 int maxHeight = Logo.Height<moduleSize_ ?moduleSize_ :
Logo.Height;

 if (maxWidth>sizesLogo[version_] * moduleSize_)
maxWidth = sizesLogo[version_] * moduleSize_;

maxWidth -= maxWidth % moduleSize_;

```

```

 if (maxHeight > sizesLogo[version_] * moduleSize_)
maxHeight = sizesLogo[version_] * moduleSize_;

maxHeight -= maxHeight % moduleSize_;

int x, y;

x = (data_[0].Length * moduleSize_ - maxWidth) / 2;
 y = (data_[0].Length * moduleSize_ - maxHeight) / 2;

 if (x < 9 * moduleSize_)
 x = 9 * moduleSize_;

 if (y < 9 * moduleSize_)
 y = 9 * moduleSize_;

x -= x % moduleSize_;
y -= y % moduleSize_;

if (Size.Width - data_[0].Length * moduleSize_ > 0)
 x += (Size.Width - data_[0].Length * moduleSize_) / 2;

if (Size.Height - data_[0].Length * moduleSize_ > 0)
 y += (Size.Height - data_[0].Length * moduleSize_) / 2;

Bitmap newLogo = new Bitmap(Logo, maxWidth, maxHeight);

var graphics = Graphics.FromImage((Image)bmp_);
graphics.DrawImage((Image)newLogo, x, y);

}

}

```

```

public Bitmap GetImage()
{
 bmp_ = new Bitmap(Size.Width, Size.Height);

 var bitmapGraphic = Graphics.FromImage(bmp_);

 bitmapGraphic.FillRectangle(new SolidBrush(backgroundColor_), new Rectangle(0,
 0, Size.Width, Size.Height));

 if (QrCodeData == null)
 return bmp_;

 data_ = GetQrCode();

 if (ModuleSizeAuto)
 {
 if (Size.Width / data_[0].Length < Size.Height /
 data_[0].Length)
 moduleSize_ = Size.Width / data_[0].Length;
 else
 moduleSize_ = Size.Height / data_[0].Length;
 }

 int lenHeight;
 int lenWidth;

 if (Size.Height < data_[0].Length * moduleSize_)
 lenHeight = Size.Height / moduleSize_;
 else
 lenHeight = data_[0].Length;

 if (Size.Width < data_[0].Length * moduleSize_)
 lenWidth = Size.Width / moduleSize_;
 else
 lenWidth = data_[0].Length;

 int x = 0, y = 0;

```

```

 if (Size.Width - data_[0].Length * moduleSize_ > 0)
 x = (Size.Width - data_[0].Length * moduleSize_) / 2;

 if (Size.Height - data_[0].Length * moduleSize_ > 0)
 y = (Size.Height - data_[0].Length * moduleSize_) / 2;

 using (var brush = new SolidBrush(moduleColor_))
 {
 for (int i = 0; i < lenWidth; ++i, x += moduleSize_)
 {
 for (int j = 0; j < lenHeight; ++j, y += moduleSize_)
 {
 if (data_[i][j] == true)
 {
 var rectangle = new Rectangle(x, y, moduleSize_,
moduleSize_);
bitmapGraphic.FillRectangle(brush, rectangle);
 rectangle = Rectangle.Empty;
 }
 }

 if (Size.Height - data_[0].Length * moduleSize_ > 0)
 y = (Size.Height - data_[0].Length * moduleSize_) / 2;
 else
 y = 0;
 }
 }

 printLogo();

 return bmp_;
 }

 private BitArray[] GetQrCode()
 {
 qrCodeSize_ = getSize();
 }

```

```

List<Point> points = new List<Point>();
getPoint(points);

 if (SelectBestMask)
 {
 List<BitArray[]>arraysList = new List<BitArray[]>();

 for (byte i = 0; i < 8; ++i)
 {
var bitMatrix = new BitArray[qrCodeSize_];
 for (uint j = 0; j < bitMatrix.Length; ++j)
bitMatrix[j] = new BitArray(qrCodeSize_);

arraysList.Add(bitMatrix);
 }

 byte maskIndex = 0;
 foreach (var qrArray in arraysList)
 {
printSearchBlock(arraysList[maskIndex]);

printPoint(arraysList[maskIndex], points);

printMaskAndCorrectoin(qrArray, maskIndex);

 if (version_ >= 6)
printVersion(qrArray);
printData(qrArray, points, maskIndex);

 ++maskIndex;
 }
 return arraysList[getPerfectMask(arraysList)];
 }
 else
 {

```

```

 var bitMatrix = new BitArray[qrCodeSize_];
 for (uint j = 0; j < bitMatrix.Length; ++j)
bitMatrix[j] = new BitArray(qrCodeSize_);

printSearchBlock(bitMatrix);

printPoint(bitMatrix, points);

printMaskAndCorrectoin(bitMatrix, 0);

if (version_ >= 6)
printVersion(bitMatrix);

printData(bitMatrix, points, 0);

 return bitMatrix;
 }

 private byte getSize()
 {
 byte qrSize = 0;

 if (version_ != 0)
 {
 for (byte i = 0; i < 7; i++)
 {
 if (i < 6 && AlciliatingPoint[version_, i + 1] == 0)
 {
qrSize = AlciliatingPoint[version_, i];
qrSize += 7;

 break;
 }
 if (i == 6)
 {
qrSize = AlciliatingPoint[version_, i];
qrSize += 7;
 }
 }
 }
 }

```

```

 }
 }
}
else
 return 21;

return qrSize;
}

private void getPoint(List<Point> points)
{
 if (version_ == 0)
 return;

 for (byte i = 0; i < 7; i++)
 {
 if (AlciliatingPoint[version_, i] == 0)
 break;

 for (byte j = 0; j < 7; j++)
 {
 if (AlciliatingPoint[version_, j] == 0)
 break;

 if (version_ >= 6)
 {
 if (AlciliatingPoint[version_, i] == 6
&&AlciliatingPoint[version_, j] == 6)
 continue;

 if (AlciliatingPoint[version_, i] == 6
&&AlciliatingPoint[version_, j + 1] == 0)
 continue;

 if (AlciliatingPoint[version_, i + 1] == 0
&&AlciliatingPoint[version_, j] == 6)

```

```

 continue;

 }

points.Add(new Point(AlciliatingPoint[version_, i], AlciliatingPoint[version_,
j]));
}

 }

}

private void setPoint(BitArray[] qrArray, int x, int y)
{
 for (byte i = 0; i < 5; ++i)
 {
 qrArray[x - 2 + i][y - 2] = true;
 qrArray[x - 2 + i][y + 2] = true;

 qrArray[x - 2][y - 2 + i] = true;
 qrArray[x + 2][y - 2 + i] = true;
 }

 qrArray[x][y] = true;
}

private void printSearchBlock(BitArray[] qrArray)
{
 for (byte i = 0; i < 7; ++i)
 {
 qrArray[i][0] = true;
 qrArray[i][6] = true;

 qrArray[0][i] = true;
 qrArray[6][i] = true;

 qrArray[(qrCodeSize_ - 1) - i][0] = true;
 qrArray[(qrCodeSize_ - 1) - i][6] = true;
 }
}

```

```

qrArray[qrCodeSize_ - 1][i] = true;
qrArray[qrCodeSize_ - 7][i] = true;

 //Оконтовканижного
qrArray[i][qrCodeSize_ - 1] = true;
qrArray[i][qrCodeSize_ - 7] = true;

qrArray[0][(qrCodeSize_ - 1) - i] = true;
qrArray[6][(qrCodeSize_ - 1) - i] = true;
}

for (byte i = 0; i < 3; ++i)
{
 for (byte j = 0; j < 3; ++j)
 {
qrArray[2 + i][2 + j] = true;

qrArray[qrCodeSize_ - 5 + i][2 + j] = true;

qrArray[2 + i][qrCodeSize_ - 5 + j] = true;
 }
}

 for (byte i = 8; i < qrCodeSize_ - 7; i += 2)
 {
qrArray[i][6] = true;
qrArray[6][i] = true;
}

}

private void printPoint(BitArray[] qrArray, List<Point> points)
{
 foreach (var point in points)

```

```

setPoint(qrArray, point.X, point.Y);
}

private void printMaskAndCorrectoin(BitArray[] qrArray, byte
maskIndex)
{
 BitArray data = new BitArray(32);

 if (correction_ == QrCodeCorrectionLevel.MIN) //L
 {
 switch (maskIndex)
 {
 case 0:
 data = new BitArray(BitConverter.GetBytes(0x77c4));
 break;

 case 1:
 data = new BitArray(BitConverter.GetBytes(0x72f3));
 break;

 case 2:
 data = new BitArray(BitConverter.GetBytes(0x7daa));
 break;

 case 3:
 data = new BitArray(BitConverter.GetBytes(0x789d));
 break;

 case 4:
 data = new BitArray(BitConverter.GetBytes(0x662f));
 break;

 case 5:
 data = new BitArray(BitConverter.GetBytes(0x6318));
 break;

 case 6:

```

```

 data = new BitArray(BitConverter.GetBytes(0x6c41));
 break;

 case 7:
 data = new BitArray(BitConverter.GetBytes(0x6976));
 break;
 }
}

if (correction_ == QrCodeCorrectionLevel.MIDDLE) //M
{
 switch (maskIndex)
 {
 case 0:
 data = new BitArray(BitConverter.GetBytes(0x5412));
 break;

 case 1:
 data = new BitArray(BitConverter.GetBytes(0x5125));
 break;

 case 2:
 data = new BitArray(BitConverter.GetBytes(0x5e7c));
 break;

 case 3:
 data = new BitArray(BitConverter.GetBytes(0x5b4b));
 break;

 case 4:
 data = new BitArray(BitConverter.GetBytes(0x45f9));
 break;

 case 5:
 data = new BitArray(BitConverter.GetBytes(0x40ce));
 break;
 }
}

```

```

 case 6:
 data = new BitArray(BitConverter.GetBytes(0x4f97));
 break;

 case 7:
 data = new BitArray(BitConverter.GetBytes(0x4aa0));
 break;
 }
}

if (correction_ == QrCodeCorrectionLevel.HIGH) //Q
{
 switch (maskIndex)
 {
 case 0:
 data = new BitArray(BitConverter.GetBytes(0x355f));
 break;

 case 1:
 data = new BitArray(BitConverter.GetBytes(0x3068));
 break;

 case 2:
 data = new BitArray(BitConverter.GetBytes(0x3f31));
 break;

 case 3:
 data = new BitArray(BitConverter.GetBytes(0x3a06));
 break;

 case 4:
 data = new BitArray(BitConverter.GetBytes(0x24b4));
 break;

 case 5:
 data = new BitArray(BitConverter.GetBytes(0x2183));
 break;
 }
}

```

```

 case 6:
 data = new BitArray(BitConverter.GetBytes(0x2eda));
 break;

 case 7:
 data = new BitArray(BitConverter.GetBytes(0x2bed));
 break;
 }
}

if (correction_ == QrCodeCorrectionLevel.VERY_HIGH) //H
{
 switch (maskIndex)
 {
 case 0:
 data = new BitArray(BitConverter.GetBytes(0x1689));
 break;

 case 1:
 data = new BitArray(BitConverter.GetBytes(0x13be));
 break;

 case 2:
 data = new BitArray(BitConverter.GetBytes(0x1ce7));
 break;

 case 3:
 data = new BitArray(BitConverter.GetBytes(0x19d0));
 break;

 case 4:
 data = new BitArray(BitConverter.GetBytes(0x0762));
 break;

 case 5:
 data = new BitArray(BitConverter.GetBytes(0x0255));

```

```

 break;

 case 6:
 data = new BitArray(BitConverter.GetBytes(0x0d0c));
 break;

 case 7:
 data = new BitArray(BitConverter.GetBytes(0x083b));
break;

 }
}

byte x = 8, y = 0xff;
for (byte i = 0; i < 8; ++i)
{
 ++y;

 if (y == 6)
 ++y;

 if (data[i] == true)
qrArray[x][y] = true;
}

--x;
for (byte i = 8; i < 16; ++i)
{
 if (data[i] == true)
qrArray[x][y] = true;

--x;

 if (x == 6)
 --x;
}

for (byte i = 0; i < 8; ++i)
{

```

```

 if (data[i] == true)
qrArray[qrCodeSize_ - 1 - i][8] = true;
 }
 for (byte i = 8; i < 16; ++i)
 {
 if (data[i] == true)
qrArray[8][qrCodeSize_ - 1 - (i - 8)] = true;
 }

qrArray[8][qrCodeSize_ - 8] = true;

 }

 private void printVersion(BitArray[] qrArray)
 {
int[] VersionCode = new int[34] { 0x027a6, 0x11738, 0x37604, 0x29f80,
0x0febc,
 0x0d91a, 0x2b826, 0x351a2,
0x1309e, 0x1c45c,
 0x3a560, 0x24ce4, 0x02dd8,
0x00a7e, 0x26b42,
 0x382c6, 0x1e3fa, 0x0d364,
0x2b258, 0x35bdc,
 0x13ae0, 0x11d46, 0x37c7a,
0x295fe, 0x0f4c2,
 0x2862d, 0x0e711, 0x10e95,
0x36fa9, 0x3480f,
 0x12933, 0x0c0b7, 0x2a18b,
0x39115 };

 BitArray data = new BitArray(BitConverter.GetBytes(VersionCode[version_ -
6]));

 for (byte i = 0; i < 3; ++i)
 {
 for (byte j = 0; j < 6; ++j)

```

```

 {
 if (data[j + i * 6] == true)
 {
 qrArray[qrCodeSize_ - 9 - i][5 - j] = true;
 qrArray[5 - j][qrCodeSize_ - 9 - i] = true;
 }
 }
 }

 private void printData(BitArray[] qrArray, List<Point> points, byte
maskIndex)
 {
 BitArray data = new BitArray(QrCodeData);

 int x = qrCodeSize_ - 1;
 int y = qrCodeSize_ - 1;

 bool up = true;
 bool offset = true;

 for (int i = 0; i < data.Length; ++i)
 {
 if (x == -1)
 break;

 if (version_ >= 6 && y == qrCodeSize_ - 11 && x <= 5)
 {
 y--;
 x -= 2;

 up = true;
 offset = true;
 }
 }
 }

```

```

 if (up)
 {
 foreach (var point in points)
 {
 int x1 = point.X - 2;
 int y1 = point.Y + 2;

 if (y == y1 && (x == (x1 + 1) || x == (x1 + 2) || x ==
(x1 + 3) || x == (x1 + 4)))
 {
 y -= 5;
 break;
 }

 if (x == x1 && y == y1)
 {
 x--;

 for (int k = 0; k < 5; k++)
 {
 if (y == 6)
 {
 y--;
 continue;
 }

 if (data[(i / 8) * 8 + (7 - i % 8)] == true)
 {
 if (setMask(maskIndex, x, y))

qrArray[x][y] = true;

 else

qrArray[x][y] = false;

 }
 else
 {

```

```

 if (setMask(maskIndex, x, y))
qrArray[x][y] = false;

 else
qrArray[x][y] = true;

 }

i++;

 y--;

 }

 x++;

 }

 }
}
else
{
 foreach (var point in points)
 {
 int x1 = point.X - 2;
 int y1 = point.Y - 2;

 if (y == y1 && (x == (x1 + 1) || x == (x1 + 2) || x ==
(x1 + 3) || x == (x1 + 4)))
 {
 y += 5;
 break;
 }

 if (x == x1 && y == y1)
 {
 x--;
 for (int k = 0; k < 4; k++)
 {
 if (y == 6)
 {

```

```

 y++;
 continue;
 }

 if (data[(i / 8) * 8 + (7 - i % 8)] == true)
 {
 if (setMask(maskIndex, x, y))

qrArray[x][y] = true;

 else

qrArray[x][y] = false;
 }
 else
 {
 if (setMask(maskIndex, x, y))

qrArray[x][y] = false;

 else

qrArray[x][y] = true;
 }

 i++;

 y++;
 offset = false;
}

break;

}

}

}

 if (version_ >= 6)
 {
if (x == qrCodeSize_ - 9 && y == 6)
 {
 y = 0;

```

```

 x = qrCodeSize_ - 12;

 for (int j = 0; j < 6; j++, i++)
 {
 if (data[(i / 8) * 8 + (7 - i % 8)] == true)
 {
 if (setMask(maskIndex, x, y))
 qrArray[x][y] = true;
 else
 qrArray[x][y] = false;
 }
 else
 {
 if (setMask(maskIndex, x, y))
 qrArray[x][y] = false;
 else
 qrArray[x][y] = true;
 }

 y++;
 }

 y++;
 x++;
 up = false;
 offset = true;
 }
}

if (y == 8 && x >= qrCodeSize_ - 8 || y == -1 || y == 8 && x
<= 8)
{
 y++;
 x -= 2;

 up = false;

```

```

 offset = true;
 }

 if (y == qrCodeSize_)
 {
 y--;
 x -= 2;

 if (x <= 8)
 y -= 8;

 up = true;
 }
 offset = true;

 if (y == qrCodeSize_ - 8 && x <= 8)
 {
 y--;
 x -= 2;

 up = true;
 offset = true;
 }

 if (x == 6)
x--;

 if (y == 6 && up)
 y--;

 if (y == 6 && !up)
 y++;

 if (data[(i / 8) * 8 + (7 - i % 8)] == true)
 {
 if (setMask(maskIndex, x, y))
qrArray[x][y] = true;
 }

```

```

 else
qrArray[x][y] = false;
 }
 else
 {
 if (setMask(maskIndex, x, y))
qrArray[x][y] = false;
 else
qrArray[x][y] = true;
 }

```

```

 if (up)
{
 if (offset)
 {
 x--;
 offset = !offset;
 }
 else
 {
 x++;
 y--;
 offset = !offset;
 }
}
else
{
 if (offset)
 {
 x--;
 offset = !offset;
 }
 else
 {
 x++;
 y++;
 }
}

```

```

 offset = !offset;
 }
}

 }
}

private bool setMask(byte number, int x, int y)
{
 switch (number)
 {
 case 0:
 if ((x + y) % 2 != 0)
 return true;
 else
 return false;
 case 1:
 if (y % 2 != 0)
 return true;
 else
 return false;
 case 2:
 if (x % 3 != 0)
 return true;
 else
 return false;
 case 3:
 if ((x + y) % 3 != 0)
 return true;
 else
 return false;
 case 4:
 if ((y / 2 + x / 3) % 2 != 0)
 return true;
 else
 return false;
 }
}

```

```

 case 5:
 if ((y * x) % 2 + (x * y) % 3 != 0)
 return true;
 else
 return false;
 case 6:
 if (((x * y) % 2 + (x * y) % 3) % 2 != 0)
 return true;
 else
 return false;
 case 7:
 if (((x * y) % 3 + (x + y) % 2) % 2 != 0)
 return true;
 else
 return false;

 default:
 return true;
 }
}

private byte getPerfectMask(List<BitArray[]>arraysList)
{
 byte Index = 0;
 int min = int.MaxValue;
 byte ReturnIndex = 0;
 foreach (var qrArray in arraysList)
 {
 int Count = 0;
 int Value = 0;

 for (int i = 0; i < qrCodeSize_; ++i)
 for (int j = 0; j < qrCodeSize_ - 1; ++j)
 {
 if (qrArray[j][i] == qrArray[j + 1][i])
 Count++;

 if (qrArray[j][i] != qrArray[j + 1][i])

```

```

 {
 Value += Count < 4 ? 0 : Count - 1;
 Count = 0;
 }
 }

 Count = 0;

 for (int i = 0; i < qrCodeSize_ - 1; ++i)
 for (int j = 0; j < qrCodeSize_; ++j)
 {
 if (qrArray[i][j] == qrArray[i + 1][j])
 Count++;

 if (qrArray[i][j] != qrArray[i + 1][j])
 {
 Value += Count < 4 ? 0 : Count - 1;
 Count = 0;
 }
 }

 Count = 0;

 for (int i = 0; i < qrCodeSize_ - 1; i += 2)
 for (int j = 0; j < qrCodeSize_ - 1; ++j)
 {
 if (qrArray[i][j] == qrArray[i][j + 1] && qrArray[i][j]
== qrArray[i + 1][j + 1] && qrArray[i][j] == qrArray[i + 1][j])
 Value += 3;
 }

 for (int i = 0; i < qrCodeSize_; ++i)
 for (int j = 0; j < qrCodeSize_ - 15; ++j)
 {
 if (qrArray[i][j] == false && qrArray[i][j + 1] ==
false && qrArray[i][j + 2] == false && qrArray[i][j + 3] == false

```

```

&&qrArray[i][j + 4] == true &&qrArray[i][j + 5] == false &&qrArray[i][j + 6]
== true &&qrArray[i][j + 7] == true &&qrArray[i][j + 8] == true
&&qrArray[i][j + 9] == false &&qrArray[i][j + 10] == true &&qrArray[i][j + 11]
== false &&qrArray[i][j + 12] == false
&&qrArray[i][j + 13] == false &&qrArray[i][j + 14] == false)
Value += 40;

 }

for (int i = 0; i < qrCodeSize_; ++i)
 for (int j = 0; j < qrCodeSize_ - 10; ++j)
 {
 if (qrArray[i][j] == false &&qrArray[i][j + 1] ==
false &&qrArray[i][j + 2] == false &&qrArray[i][j + 3] == false
&&qrArray[i][j + 4] == true &&qrArray[i][j + 5] == false &&qrArray[i][j + 6]
== true &&qrArray[i][j + 7] == true
&&qrArray[i][j + 8] == true &&qrArray[i][j + 9] == false &&qrArray[i][j + 10]
== true)
Value += 40;

 }

for (int i = 0; i < qrCodeSize_; ++i)
 for (int j = 0; j < qrCodeSize_ - 10; ++j)
 {
 if (qrArray[i][j] == true &&qrArray[i][j + 1] == false
&&qrArray[i][j + 2] == true &&qrArray[i][j + 3] == true &&qrArray[i][j + 4] ==
true
&&qrArray[i][j + 5] == false &&qrArray[i][j + 6] == true &&qrArray[i][j + 7]
== false &&qrArray[i][j + 8] == false &&qrArray[i][j + 9] == false
&&qrArray[i][j + 10] == false)
Value += 40;

 }

int Black = 0;

 for (int i = 0; i < qrCodeSize_; ++i)
 for (int j = 0; j < qrCodeSize_; ++j)

```

```

 Black += qrArray[i][j] == true ? 1 : 0;

 Value += Math.Abs(Black / ((qrCodeSize_) * 2) * 100 - 50) * 2;

 if (min > Value)
 {
 min = Value;
ReturnIndex = Index;
 }

 ++Index;
 }
 return ReturnIndex;
}

public virtual void SetText(string text)
{
byte[] textBytes = Encoding.UTF8.GetBytes(text);
 if (QrCodeDllCore.QrCodeGenerate(QrCode, textBytes,
(short)(textBytes.Length)) != 0) {
MessageBox.Show("Слишком много данных для данной версии и уровня коррекции");
return;
 }

ushortsizeQrCodeData = QrCodeDllCore.QrCodeGetDataSize(QrCode);

QrCodeData = new byte[sizeQrCodeData];
QrCodeDllCore.QrCodeRead(QrCode, 0, QrCodeData, sizeQrCodeData);

 version_ = (byte)(QrCodeDllCore.QrCodeGetVersion(QrCode) - 1);
 correction_ = QrCodeDllCore.QrCodeGetCorrectionLevel(QrCode);
 }

protected override void OnPaint(PaintEventArgs pe)
{
 base.OnPaint(pe);
}

```

```

pe.Graphics.Clear(Color.White);

 Bitmap qr = this.GetImage();
pe.Graphics.DrawImage(qr, new Point(0, 0));
 }
}
}

```

#### Файл «QrGen.cs»

```

namespace QrGen
{
 public partial class QrGen : Form
 {
 enum ActivPanel
 {
 Text,
 Ref,
 SMS,
 Wifi,
 VCard,
 Email
 }

 private ActivPanel activPanel_ = ActivPanel.Text;

 private void updateInfoQrCode()
 {
 switch (activPanel_)
 {
 case ActivPanel.Text:
 {
 qrBox1.SetText(textBoxText.Text.ToString());
 break;
 }
 case ActivPanel.Ref:
 {
 qrBox1.SetText("URL:" + textBoxRef.Text.ToString());
 break;
 }
 case ActivPanel.SMS:
 {
 SMSMessage smsMessage = new();
 smsMessage.Number = textBoxSMSNumber.Text.ToString();
 smsMessage.Message = textBoxSMSMessage.Text.ToString();
 qrBox1.SetText(smsMessage.ToString());
 break;
 }
 case ActivPanel.Email:
 {
 EmailMessage emailMessage = new();
 emailMessage.Message = textBoxEmailMessage.Text.ToString();
 emailMessage.Email = textBoxEmailAddress.Text.ToString();
 }
 }
 }
 }
}

```

```

emailMessage.Title = textBoxEmailTitle.Text.ToString();
 qrBox1.SetText(emailMessage.ToString());
 break;
 }
 case ActivPanel.Wifi:
 {
WifiConnectionwifiConnection = new();
wifiConnection.Name = textBoxWifiName.Text.ToString();
wifiConnection.Password = textBoxWifiPassword.Text.ToString();

wifiConnection.isHidden = checkBoxIsHidden.Checked;

 if ((checkBoxWifiSecurityWPA.Checked ||
checkBoxWifiSecurityWEP.Checked) == false)
wifiConnection.securityMode = WifiSecurityMode.NONE;

 if (checkBoxWifiSecurityWPA.Checked)
wifiConnection.securityMode = WifiSecurityMode.WPA;

 if (checkBoxWifiSecurityWEP.Checked)
wifiConnection.securityMode = WifiSecurityMode.WEP;

 qrBox1.SetText(wifiConnection.ToString());
 break;
 }
 case ActivPanel.VCard:
 {
VCard vCard = new();
vCard.Name = textBoxVCardFIO.Text.ToString();

vCard.Email = textBoxVCardEmail.Text.ToString();

vCard.Mobile = textBoxVCardNumberMobile.Text.ToString();
vCard.Fax = textBoxVCardNumberFax.Text.ToString();
vCard.Phone = textBoxVCardNumberCity.Text.ToString();

vCard.Organization = textBoxVCardCompanyName.Text.ToString();
vCard.JobTitle = textBoxVCardCompanyPosition.Text.ToString();

vCard.Country = textBoxVCardLocationCity.Text.ToString();
vCard.Oblast = textBoxVCardLocationOblast.Text.ToString();
vCard.City = textBoxVCardLocationTown.Text.ToString();
vCard.Street = textBoxVCardLocationStreet.Text.ToString();
vCard.Home = textBoxVCardLocationHomeNumber.Text.ToString();
vCard.MailIndex = textBoxVCardLocationIndex.Text.ToString();

vCard.Site = textBoxVCardSite.Text.ToString();

 qrBox1.SetText(vCard.ToString());
 break;
 }
 default:
 break;
 }
 }
}

```

```

 public QrGen()
 {
InitializeComponent();
comboBoxCorrectionLevel.SelectedIndex = 0;
comboBoxVersion.SelectedIndex = 0;
 }

 private void comboBoxCorrectionLevel_SelectedIndexChanged(object
sender, EventArgs e)
 {
 int result = 0;
 result =
qrBox1.SetCorrectionLevel((QrCodeCorrectionLevel)comboBoxCorrectionLevel.Selec
tedIndex);
 if (result != 0)
MessageBox.Show("Неудалось устаноовить коррекции");
 }

 private void comboBoxVersion_SelectedIndexChanged(object sender,
EventArgs e)
 {
 int result = 0;
 result = qrBox1.SetVersion((byte)comboBoxVersion.SelectedIndex);
 if (result != 0)
MessageBox.Show("Неудалось устаноовить версию");
 }

 private void unVisiablePanel()
 {
 switch (activPanel_)
 {
 case ActivPanel.Text:
 {
panelText.Visible = false;
 break;
 }
 case ActivPanel.Ref:
 {
panelRef.Visible = false;
 break;
 }
 case ActivPanel.SMS:
 {
panelSMS.Visible = false;
 break;
 }
 case ActivPanel.Email:
 {
panelEmail.Visible = false;
 break;
 }
 case ActivPanel.Wifi:
 {
panelWifi.Visible = false;
 break;
 }
 }
 }
 }

```

```

 }
 case ActivPanel.VCard:
 {
panelVCard.Visible = false;
 break;
 }
 default:
 break;
 }
}

private void buttonText_Click(object sender, EventArgs e)
{
unVisiablePanel();
panelText.Visible = true;
activPanel_ = ActivPanel.Text;
}

private void buttonRef_Click(object sender, EventArgs e)
{
unVisiablePanel();
panelRef.Visible = true;
activPanel_ = ActivPanel.Ref;
}

private void buttonSMS_Click(object sender, EventArgs e)
{
unVisiablePanel();
panelSMS.Visible = true;
activPanel_ = ActivPanel.SMS;
}

private void buttonEmail_Click(object sender, EventArgs e)
{
unVisiablePanel();
panelEmail.Visible = true;
activPanel_ = ActivPanel.Email;
}

private void buttonVCard_Click(object sender, EventArgs e)
{
unVisiablePanel();
panelVCard.Visible = true;
activPanel_ = ActivPanel.VCard;
}

private void buttonWifi_Click(object sender, EventArgs e)
{
unVisiablePanel();
panelWifi.Visible = true;
activPanel_ = ActivPanel.Wifi;
}

private void checkBoxWifiIsPasswordVisiable_CheckedChanged(object
sender, EventArgs e)
{

```

```

 if (checkBoxWifiIsPasswordVisiable.Checked)
textBoxWifiPassword.PasswordChar = '*';
 else
textBoxWifiPassword.PasswordChar = '\0';
 }

 private void checkBoxWifiSecurityNone_CheckedChanged(object sender,
EventArgs e)
 {
 if (checkBoxWifiSecurityNone.Checked)
 {
 checkBoxWifiSecurityWEP.Checked = false;
 checkBoxWifiSecurityWPA.Checked = false;
 }
 }

 private void checkBoxWifiSecurityWPA_CheckedChanged(object sender,
EventArgs e)
 {
 if (checkBoxWifiSecurityWPA.Checked)
 {
 checkBoxWifiSecurityNone.Checked = false;
 checkBoxWifiSecurityWEP.Checked = false;
 }
 }

 private void checkBoxWifiSecurityWEP_CheckedChanged(object sender,
EventArgs e)
 {
 if (checkBoxWifiSecurityWEP.Checked)
 {
 checkBoxWifiSecurityNone.Checked = false;
 checkBoxWifiSecurityWPA.Checked = false;
 }
 }

 private void buttonDeleteLogo_Click(object sender, EventArgs e)
 {
 qrBox1.Logo = null;
 comboBoxCorrectionLevel.Enabled = true;
 }

 private void buttonAddLogo_Click(object sender, EventArgs e)
 {
 using (OpenFileDialogofd = new())
 {
 ofd.Filter = "Image files (*.BMP;*.PNG;*.GIF;*.JPEG;*.ICON)
|*.BMP;*.PNG;*.GIF;*.JPEG;*.ICON";
 if (ofd.ShowDialog() == DialogResult.OK)
 {
 comboBoxCorrectionLevel.SelectedIndex = 3;
 comboBoxCorrectionLevel.Enabled = false;

 qrBox1.Logo = Image.FromFile(ofd.FileName);

 qrBox1.SetCorrectionLevel((QrCodeCorrectionLevel)comboBoxCorrectionLevel.Selec
tedIndex);
 updateInfoQrCode();
 }
 }
 }

```

```

 qrBox1.Refresh();
 }
}
private void buttonSaveAsFile_Click(object sender, EventArgs e)
{
 using (SaveFileDialogsfd = new())
 {
 sfd.Filter = "Image files (*.BMP;*.PNG;*.GIF;*.JPEG;*.ICON)
|*.BMP;*.PNG;*.GIF;*.JPEG;*.ICON";
 if (sfd.ShowDialog() == DialogResult.OK)
 {
 if (sfd.FileName != "")
 {
 if (sfd.FileName.IndexOf(".bmp") != -1)
 {
 qrBox1.GetImage().Save(sfd.FileName,
System.Drawing.Imaging.ImageFormat.Bmp); ;

 if (sfd.FileName.IndexOf(".gif") != -1)
 {
 qrBox1.GetImage().Save(sfd.FileName,
System.Drawing.Imaging.ImageFormat.Gif);

 if (sfd.FileName.IndexOf(".icon") != -1)
 {
 qrBox1.GetImage().Save(sfd.FileName,
System.Drawing.Imaging.ImageFormat.Icon);

 if (sfd.FileName.IndexOf(".jpeg") != -1)
 {
 qrBox1.GetImage().Save(sfd.FileName,
System.Drawing.Imaging.ImageFormat.Jpeg);

 if (sfd.FileName.IndexOf(".png") != -1)
 {
 qrBox1.GetImage().Save(sfd.FileName,
System.Drawing.Imaging.ImageFormat.Png);
 }
 }
 }
 }
 }
 }
 }
 }

 private void buttonGenerate_Click(object sender, EventArgs e)
 {
 updateInfoQrCode();
 qrBox1.Refresh();
 }
}

```

# ПриложениеВ

(Обязательное)

Генератор QR-кода

Презентация

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
|      |      |          |         |      |                       | 129  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

# Тема работы: «Генератор QR-кода»

Автор работы: студент группы 19ВВП1 Ильин С.Р.

Научный руководитель: к.т.н., доцент каф. ВТ ПГУ Карамышева Н.С.

Пенза 2023

## Цель и объект работы

### Цель работы

Разработать приложение «Генератор QR-кода» на операционной системе *Windows*, позволяющее генерировать QR-код и хранить внутри QR-кода данные разных типов, а также имеющее возможность сохранять QR-код в виде файла.

### Объект работы

Приложение, выполняющее следующие функции:

- генерация QR-кода по исходным данным;
- гибкая настройка параметров QR-кода;
- сохранения QR-кода в файл;
- наличие удобного графического интерфейса.

# Постановка задачи

В процессе разработки должны быть решены следующие задачи:

- запись в QR-код разных типов данных;
- создание компонента графического интерфейса для отображения QR-кода;
- размещение логотипа на QR-коде;
- сохранение QR-кода в файл;
- возможность ручного выбора версии и уровня коррекции QR-кода.

## Аналоги

**QR Coder.ru**  
Генератор QR-кодов / QR Code Generator

[создание кода в один клик](#) [программы для распознавания](#)

---

### ГЕНЕРАТОР QR КОДОВ

закодировать: [любой текст](#) [ссылку на сайт](#) [визитную карточку](#) [sms-сообщение](#)

введите текст для кодирования:

размер: ☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐ 6

**ИСПОЛЬЗОВАНИЕ:**

1. Возьмите мобильный телефон с камерой,
2. Запустите программу для сканирования кода,
3. Наведите объектив камеры на код,
4. Получите информацию!

**ПРИМЕНЕНИЕ:**

В качестве применения qr-кодов можно назвать: размещение их изображений в интернете, нанесение на визитные карточки, футболки, рекламные вывески и многое другое.

**ЧТО ТАКОЕ QR-КОД:**

QR код «QR - Quick Response - Быстрый Отклик» – это двухмерный штрихкод (бар-код), предоставляющий информацию для быстрого ее распознавания с помощью камеры на мобильном телефоне.

При помощи QR-кода можно закодировать любую информацию, например: текст, номер телефона, ссылку на сайт или визитную карточку.

**ВАШ QR-КОД:**



Постоянная ссылка на изображение:

HTML-код для вставки в блог:

## Аналоги

Один

Списком

Текст или ссылка

Телефон

Email

Цифровая визитка

WiFi

Введите текст или ссылку

Произвольный текст который нужно закодировать

Использовать кодируемый текст как подпись

Подпись

Под QR кодом

Цвет QR кода

Цвет фона

Прозрачный фон

Логотип, png

Выберите файл

Файл не выбран

Для лучшего отображения логотипа возможно понадобится его обработать в графическом редакторе

Коррекция ошибок

Уровень H (30%)

Настройки

PNG

SVG

EPS

PDF

Размер, px

500

Отступ, px

0

Результат

Сгенерировать код

Сервис «code-qr.ru»

## Аналоги

Ссылка

Текст

Вконтакте

Tik Tok

Instagram

YouTube

Telegram

Facebook

Электронная почта

Местоположение

Телефон

СМС

WhatsApp

Skype

Zoom

Wi-Fi

Визитка

Событие

PayPal

Bitcoin

Текст

Сообщение

Привет мир!

Цвета

Дизайн

Логотип

Рамка

Параметры

Сохранить

Поддержите проект

Назначение платежа

Перевод по кнопке

Сумма

100 Р

Пожертвовать

Сервис «qr-online.ru»

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.08.001 ПЗ

Лист

132

# Алгоритм создания QR-кода

Процесс генерации *QR* кода делится на несколько чётких шагов:

- кодирование данных;
- добавление служебной информации и заполнения;
- разделение информации на блоки;
- создание байтов коррекции;
- объединение блоков;
- размещение информации на *QR* коде.

## Добавление служебной информации

|   | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | 10    |
|---|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| L | 152   | 272   | 440   | 640   | 864   | 1088  | 1248  | 1552  | 1856  | 2192  |
| M | 128   | 224   | 352   | 512   | 688   | 864   | 992   | 1232  | 1456  | 1728  |
| Q | 104   | 176   | 272   | 384   | 496   | 608   | 704   | 880   | 1056  | 1232  |
| H | 72    | 128   | 208   | 288   | 368   | 480   | 528   | 688   | 800   | 976   |
|   | 11    | 12    | 13    | 14    | 15    | 16    | 17    | 18    | 19    | 20    |
| L | 2592  | 2960  | 3424  | 3688  | 4184  | 4712  | 5176  | 5768  | 6360  | 6888  |
| M | 2032  | 2320  | 2672  | 2920  | 3320  | 3624  | 4096  | 4504  | 5016  | 5352  |
| Q | 1440  | 1648  | 1952  | 2088  | 2360  | 2500  | 2936  | 3176  | 3560  | 3880  |
| H | 1120  | 1264  | 1440  | 1576  | 1784  | 2024  | 2264  | 2504  | 2728  | 3080  |
|   | 21    | 22    | 23    | 24    | 25    | 26    | 27    | 28    | 29    | 30    |
| L | 7456  | 8048  | 8752  | 9392  | 10208 | 10960 | 11744 | 12248 | 13048 | 13880 |
| M | 5712  | 6296  | 6880  | 7312  | 8000  | 8496  | 9024  | 9544  | 10136 | 10984 |
| Q | 4096  | 4544  | 4912  | 5312  | 5744  | 6032  | 6464  | 6968  | 7288  | 7880  |
| H | 3248  | 3536  | 3712  | 4112  | 4304  | 4768  | 5024  | 5288  | 5608  | 5960  |
|   | 31    | 32    | 33    | 34    | 35    | 36    | 37    | 38    | 39    | 40    |
| L | 14744 | 15640 | 16568 | 17528 | 18448 | 19472 | 20528 | 21616 | 22496 | 23648 |
| M | 11640 | 12328 | 13048 | 13800 | 14496 | 15312 | 15936 | 16816 | 17728 | 18672 |
| Q | 8264  | 8920  | 9368  | 9848  | 10288 | 10832 | 11408 | 12016 | 12656 | 13328 |
| H | 6344  | 6760  | 7208  | 7688  | 7888  | 8432  | 8768  | 9136  | 9776  | 10208 |

Таблица количества информации для версий QR-кода

## Добавление служебной информации

|                   | Версия 1–9 | Версия 10–26 | Версия 27–40 |
|-------------------|------------|--------------|--------------|
| Цифровое          | 10 бит     | 12 бит       | 14 бит       |
| Буквенно-цифровое | 9 бит      | 11 бит       | 13 бит       |
| Побайтовое        | 8 бит      | 16 бит       | 16 бит       |

Таблица размеров служебных полей

## Разделение информации на блоки

|   |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|
|   | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
| L | 1  | 1  | 1  | 1  | 1  | 2  | 2  | 2  | 2  | 4  |
| M | 1  | 1  | 1  | 2  | 2  | 4  | 4  | 4  | 5  | 5  |
| Q | 1  | 1  | 2  | 2  | 4  | 4  | 6  | 6  | 8  | 8  |
| H | 1  | 1  | 2  | 4  | 4  | 4  | 5  | 6  | 8  | 8  |
|   | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| L | 4  | 4  | 4  | 4  | 6  | 6  | 6  | 6  | 7  | 8  |
| M | 5  | 8  | 9  | 9  | 10 | 10 | 11 | 13 | 14 | 16 |
| Q | 8  | 10 | 12 | 16 | 12 | 17 | 16 | 18 | 21 | 20 |
| H | 11 | 11 | 16 | 16 | 18 | 16 | 19 | 21 | 25 | 25 |
|   | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| L | 8  | 9  | 9  | 10 | 12 | 12 | 12 | 13 | 14 | 15 |
| M | 17 | 17 | 18 | 20 | 21 | 23 | 25 | 26 | 28 | 29 |
| Q | 23 | 23 | 25 | 27 | 29 | 34 | 34 | 35 | 38 | 40 |
| H | 25 | 34 | 30 | 32 | 35 | 37 | 40 | 42 | 45 | 48 |
|   | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| L | 16 | 17 | 18 | 19 | 19 | 20 | 21 | 22 | 24 | 25 |
| M | 31 | 33 | 35 | 37 | 38 | 40 | 43 | 45 | 47 | 49 |
| Q | 43 | 45 | 48 | 51 | 53 | 56 | 59 | 62 | 65 | 68 |
| H | 51 | 54 | 57 | 60 | 63 | 66 | 70 | 74 | 77 | 81 |

Таблица количества блоков

## Создание байтов коррекции

|   |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|
|   | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
| L | 7  | 10 | 15 | 20 | 26 | 18 | 20 | 24 | 30 | 18 |
| M | 10 | 16 | 26 | 18 | 24 | 16 | 18 | 22 | 22 | 26 |
| Q | 13 | 22 | 18 | 26 | 18 | 24 | 18 | 22 | 20 | 24 |
| H | 17 | 28 | 22 | 16 | 22 | 28 | 26 | 26 | 24 | 28 |
|   | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| L | 20 | 24 | 26 | 30 | 22 | 24 | 28 | 30 | 28 | 28 |
| M | 30 | 22 | 22 | 24 | 24 | 28 | 28 | 26 | 26 | 26 |
| Q | 28 | 26 | 24 | 20 | 30 | 24 | 28 | 28 | 26 | 30 |
| H | 24 | 28 | 22 | 24 | 24 | 30 | 28 | 28 | 26 | 28 |
|   | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| L | 28 | 28 | 30 | 30 | 26 | 28 | 30 | 30 | 30 | 30 |
| M | 26 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 |
| Q | 28 | 30 | 30 | 30 | 30 | 28 | 30 | 30 | 30 | 30 |
| H | 30 | 24 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
|   | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| L | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| M | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 |
| Q | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| H | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |

Таблица количества байтов коррекции  
для QR-кода

| Количество байтов коррекции | Генерирующий многочлен                                                                                                                      |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 7                           | 87, 229, 146, 149, 238, 102, 21                                                                                                             |
| 10                          | 251, 67, 46, 61, 118, 70, 64, 94, 32, 45                                                                                                    |
| 13                          | 74, 152, 176, 100, 86, 100, 106, 104, 130, 218, 206, 140, 78                                                                                |
| 15                          | 8, 163, 61, 91, 202, 37, 51, 58, 58, 237, 140, 124, 5, 99, 105                                                                              |
| 16                          | 120, 104, 107, 109, 102, 161, 76, 3, 91, 191, 147, 169, 182, 194, 225, 120                                                                  |
| 17                          | 43, 139, 206, 78, 43, 239, 123, 206, 214, 147, 24, 99, 150, 39, 243, 163, 136                                                               |
| 18                          | 215, 234, 158, 94, 184, 97, 118, 170, 79, 187, 152, 148, 252, 179, 5, 98, 96, 153                                                           |
| 20                          | 17, 60, 79, 50, 61, 163, 26, 187, 202, 180, 221, 225, 83, 239, 156, 164, 212, 212, 188, 190                                                 |
| 22                          | 210, 171, 247, 242, 93, 230, 14, 109, 221, 53, 200, 74, 8, 172, 98, 80, 219, 134, 160, 105, 165, 231                                        |
| 24                          | 229, 121, 135, 48, 211, 117, 251, 126, 159, 180, 169, 152, 192, 226, 228, 218, 111, 0, 117, 232, 87, 96, 227, 21                            |
| 26                          | 173, 125, 158, 2, 103, 182, 118, 17, 145, 201, 111, 28, 165, 53, 161, 21, 245, 142, 13, 102, 48, 227, 153, 145, 218, 70                     |
| 28                          | 168, 223, 200, 104, 224, 234, 108, 180, 110, 190, 195, 147, 205, 27, 232, 201, 21, 43, 245, 87, 42, 195, 212, 119, 242, 37, 9, 123          |
| 30                          | 41, 173, 145, 152, 216, 31, 179, 182, 50, 48, 110, 86, 239, 96, 222, 125, 42, 173, 226, 193, 224, 130, 156, 37, 251, 216, 238, 40, 192, 180 |

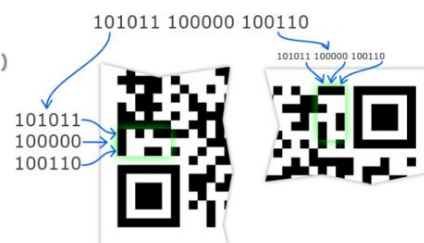
Генерирующие многочлены

## Размещение информации на QR-коде



Элементы QR-кода

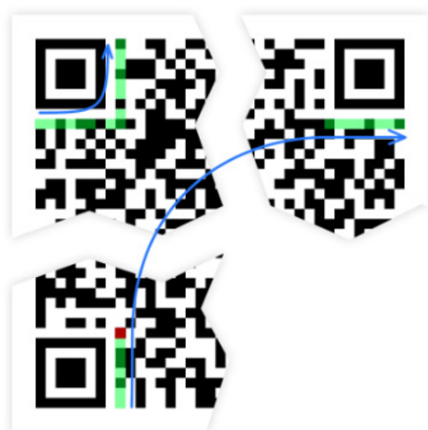
- Поисковые узоры
- Выравнивающие узоры
- Полосы синхронизации
- Код версии (с 7-й версии)
- Код маски и уровня коррекции
- Отступ



Код версии QR-кода

## Маска и уровень коррекции

001011010001001



Код маски и уровня коррекции

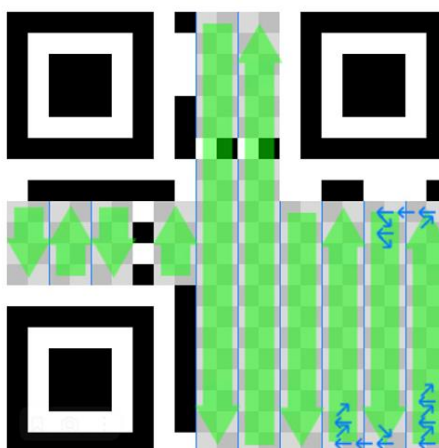
| Номер маски | Маска                            |
|-------------|----------------------------------|
| 0           | $(X+Y) \% 2$                     |
| 1           | $Y \% 2$                         |
| 2           | $X \% 3$                         |
| 3           | $(X + Y) \% 3$                   |
| 4           | $(X/3 + Y/2) \% 2$               |
| 5           | $(X*Y) \% 2 + (X*Y) \% 3$        |
| 6           | $((X*Y) \% 2 + (X*Y) \% 3) \% 2$ |
| 7           | $((X*Y) \% 3 + (X+Y) \% 2) \% 2$ |

Формулы масок

| Уровень коррекции | Допустимый процент нарушений |
|-------------------|------------------------------|
| L                 | до 7%                        |
| M                 | до 15%                       |
| Q                 | до 25%                       |
| H                 | до 30%                       |

Процент нарушений уровней коррекции

## Добавление данных

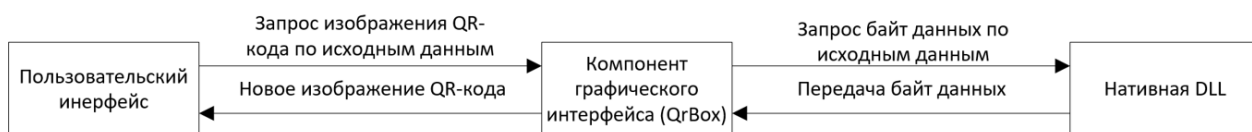


Добавление данных на QR-код

## Стек технологий



## Модель взаимодействия компонентов приложения



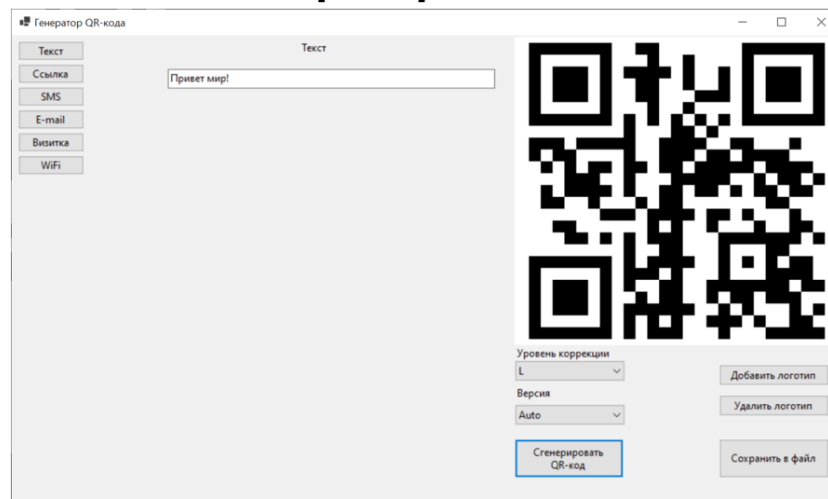
## Диаграмма вариантов использования



## Диаграмма последовательностей

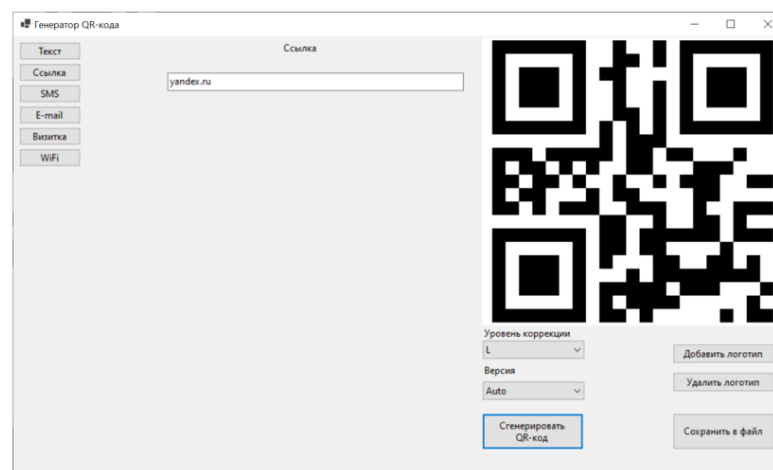


## Генератор QR-кода



Генератор QR-кода  
текстовая информация

## Генератор QR-кода



Генератор QR-кода  
ссылка

## Генератор QR-кода

Генератор QR-кода

Ссылка

Номер телефона:

7777777777

Сообщение:

Привет, мир!

Уровень коррекции

L

Версия

Auto

Добавить логотип

Удалить логотип

Сгенерировать QR-код

Сохранить в файл

Генератор QR-кода  
SMS-сообщение

## Генератор QR-кода

Генератор QR-кода

Ссылка

Электронная почта:

hello@mail.ru

Тема:

Hello

Сообщение:

hello world!

Уровень коррекции

L

Версия

Auto

Добавить логотип

Удалить логотип

Сгенерировать QR-код

Сохранить в файл

Генератор QR-кода  
электронное-сообщение

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

# Генератор QR-кода

Генератор QR-кода

Визитка

Текст Ссылка SMS E-mail Визитка WiFi

ФИО: Иванов Иван Иванович

Электронная почта: ivan@mail.ru

Номера: Мобильный: 7777777777

Городской:

Факс:

Компания: Название: Кофейня Должность: Бариста

Местоположение: Страна: Россия Область: Пензенская Город: Пенза Улица: Дом: Индекс: Веб-сайт: barista.ru

Уровень коррекции: L Версия: Auto

Добавить логотип Удалить логотип

Сгенерировать QR-код Сохранить в файл

Генератор QR-кода  
визитка

# Генератор QR-кода

Генератор QR-кода

WiFi

Текст Ссылка SMS E-mail Визитка WiFi

Название сети: Wi-Fi

☒ Скрытая

Пароль:

☒ Скрывать пароль

Шифрование: ☐ Нет ☒ WPA/WPA2 ☐ WEP

Уровень коррекции: L Версия: Auto

Добавить логотип Удалить логотип

Сгенерировать QR-код Сохранить в файл

Генератор QR-кода  
подключение к сети Wi-Fi

## Заключение

В ходе выполнения выпускной квалификационной работы был разработан генератор QR-кода на языках программирования C# и C++. Графический интерфейс был реализован с помощью фреймворка [WinForms](#).

**Разработанное приложение позволяет:**

- выбрать уровень коррекции QR-кода;
- выбрать версию QR-кода;
- добавлять логотип на QR-код;
- сохранять сгенерированный QR-код в файл;

**Разработанное приложение позволяет создать QR-код содержащий следующую информацию:**

- ссылка;
- текст;
- визитка;
- SMS-сообщение;
- E-mail-сообщение
- Wi-Fi

**Возможно усовершенствовать разработанную автоматизированную систему, добавив:**

- изменение формы и дизайна QR-кода и его модулей

# Спасибо за внимание!

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 142  |