Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ГЕОСИСТЕМ И ТЕХНОЛОГИЙ»

(СГУГиТ)



Кафедра прикладной информатики и информационных систем

Выпускная квалификационная работа соответствует установленным требованиям и направляется в ГЭК для защиты

Заведующий кафедрой Т. Ю. Бугакова

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Программа бакалавриата

09.03.02 – Информационные системы и технологии

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ПРОВЕРКИ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ МАШИННОГО ЗРЕНИЯ



Выпускник А. А. Бедрин



Руководитель П. Ю. Бугаков

Нормоконтролер С. Ю. Кацко

Новосибирск ***–*** 2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ГЕОСИСТЕМ И ТЕХНОЛОГИЙ»

(СГУГиТ)

УТВЕРЖДАЮ

Зав. кафедрой Т. Ю. Бугакова

« » 201\_\_ г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студенту (студентке) Бедрину Алексею Андреевичу Группа БИ-41 Институт геодезии и менеджмента (ИГиМ)

Направление подготовки 09.03.02 – Информационные системы и технологии

Тема ВКР Разработка автоматизированной системы проверки результатов тестирования с использованием технологии машинного зрения

Руководитель Бугаков Петр Юрьевич

Ученое звание, ученая степень руководителя доцент, канд. техн. наук

Место работы, должность руководителя СГУГиТ, доцент Срок сдачи полностью оформленного задания на кафедру Задание на ВКР (перечень рассматриваемых вопросов):

1. Обзор существующих разработок.
2. Выбор программных средств реализации.
3. Подготовка типового бланка ответов.
4. Разработка и тестирование прототипа приложение.
5. Подготовка к портированию на мобильные платформы.

Перечень графического материала с указанием основных чертежей и (или) иллюстративного материала (формат А1): нет

Исходные данные к ВКР (перечень основных материалов, собранных в период преддипломной практики или выданных руководителем)

Информационные, текстовые и графические материалы по теме ВКР

ГРАФИК ВЫПОЛНЕНИЯ ВКР

|  |  |  |
| --- | --- | --- |
| Номер этапа | Этапы ВКР | Срок исполнения |
| 1 | Начало выполнения ВКР |  |
| 2 | Подбор литературы и исходных материалов |  |
| 3 | Выполнение исследовательских, экспериментальных, расчетных работ (нужное подчеркнуть) |  |
| 4 | Выполнение графических (иллюстративных) работ |  |
| 5 | Текстовая часть ВКР (указать ориентировочные названия разделов и конкретные сроки их написания) |  |
|  | Анализ предметной области |  |
|  | Разработка прототипа приложения для проверки результатов тестирования |  |
|  | Тестирование и подготовка приложения к портированию на мобильные платформы |  |
| 6 | Первый просмотр руководителем |  |
| 7 | Второй просмотр руководителем |  |
| 8 | Срок сдачи ВКР на кафедру |  |



« » 2020г. Руководитель



Задание принял к исполнению и с графиком согласен

РЕФЕРАТ

Бедрин Алексей Андреевич. Разработка автоматизированной системы для проверки результатов тестирования с использованием технологии машинного зрения.

Место дипломирования: Сибирский государственный университет геосистем и технологий, кафедра прикладной информатики и информационных систем.

Руководитель – канд. тех. наук, доцент СГУГиТ Бугаков П.Ю.

2020 г., 09.03.02 «Информационные системы и технологии», программа бакалавриата.

66 с., 27 рис., 18 источников, 1 приложение.

VISUAL STUDIO, C#, OPENCV, EMGUCV, ПРОГРАММИРОВАНИЕ, ТЕСТИРОВАНИЕ, ПРИЛОЖЕНИЕ, КОМПБЮТЕРНОЕ ЗРЕНИЕ

Целью выпускной квалификационной работы является разработка автоматизированной системы проверки результатов тестирования с использованием технологии машинного зрения.

В выпускной квалификационной работе выполнен обзор существующих разработок; выбор программных средств реализации; подготовка типового бланка ответов; разработка и тестирование прототипа приложения; подготовка к портированию на мобильные платформы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8
   1. Описание предметной области 8
   2. Существующие разработки 8
   3. Современные технологии в области машинного зрения 10
2. РАЗРАБОТКА ПРОТОТИПА ПРИЛОЖЕНИЯ ДЛЯ ПРОВЕРКИ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ 14
   1. Выбор средств разработки программного обеспечения 14
   2. Настройка среды разработки 16
   3. Разработка структуры бланка ответов 19
   4. Описание процесса разработки программной части 20
3. ТЕСТИРОВАНИЕ И ПОДГОТОВКА ПРИЛОЖЕНИЯ К ПОРТИРОВАНИЮ НА МОБИЛЬНЫЕ ПЛАТФОРМЫ 27
   1. Тестирование разработанного программного обеспечения 27
   2. Создание прототипа Android-приложения 31

ЗАКЛЮЧЕНИЕ 36

СПИСОК ЛИТЕРАТУРЫ 37

ПРИЛОЖЕНИЕ A (обязательное) ПРОГРАММНЫЙ КОД 40

ВВЕДЕНИЕ

В настоящее время тесты стали одним из основных видов контроля знаний [1, 3-10]. Они активно используются как в системе образования, так и для подтверждения квалификации сотрудников компаний. Основное неудобство при работе с тестами заключается в большом объеме однообразных действий при составлении бланков и их дальнейшей проверке. Особенно это проявляется при одновременном тестировании большого количества респондентов). Для устранения данного недостатка необходимы средства автоматизации процесса проверки результатов тестирования.

В связи с этим основной целью работы является разработка мобильного приложения, позволяющего преподавателям и учителям быстро проверять большое количество однотипных тестов, написанных обучающимися во время проверочных работ. Для достижения поставленной цели требуется решить следующие задачи: выбрать среду разработки и язык программирования; подобрать библиотеку, позволяющую использовать технологии машинного зрения; разработать концепт приложения, его интерфейс и внешний вид; разработать само приложение, а также провести его тестирование.

Уникальность и, главное, востребованность описываемой разработки – отсутствие на рынке бесплатных аналогов, предназначенных именно для рядовых учителей и преподавателей. На государственном уровне подобные системы уже давно применяются, например, для проверки ЕГЭ и ОГЭ. Но для повседневного учебного процесса подобных приложений пока что нет. Разработка автоматизированной системы проверки результатов тестирования с использованием технологии машинного зрения позволит существенно сократить время, затраченное преподавателем на проверку большого количества однотипных тестов.

В связи с этим, было принято решение разработать прототип приложения, способного с помощью технологии машинного зрения определять количество правильных ответов на бланке для тестирования, провести тестирование созданного прототипа и портировать приложение на мобильные платформы.

Целью работы является разработка автоматизированной системы проверки результатов тестирования с использованием технологии машинного зрения

Для достижения поставленной цели необходимо решить следующие задачи:

* выполнить обзор существующих разработок;
* выбрать программные средства реализации;
* подготовить типовой блок ответов;
* разработать и протестировать прототип приложения;
* подготовить приложение к портированию на мобильные платформы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Предметной областью данного исследования является создание приложений [7], а именно создание программного продукта с использованием технологий машинного зрения с целью распознавания бланка ответов в целом и количество правильных ответов в частности. Прототип приложения может быть реализован для работы на настольных компьютерах. Это обуславливается простотой и высокой скоростью разработки, однако в дальнейшем прототип необходимо портировать на мобильные устройства, потому что, во-первых, процесс переноса изображения на ПК занимает определённое количество времени, а во-вторых, мобильная разработка в настоящее время является более перспективной деятельностью.

1.2 Существующие разработки

На данный момент на рынке мобильных предложений представлены несколько продуктов, представляющих возможность быстрого проведения тестирований и/или опросов. Но самыми заметными можно отметить два приложения: Plicker [16] и ZipGrade [18].

Первое приложение служит не сколько для проведения тестирований, сколько для тестирования. То есть респондентам не раздаются бланки для ответов, они получают набор карточек, каждая с определённым рисунком, определяющим вариант ответа. Проводящий опрос задаёт вопрос и просит каждого поднять определённую табличку (чаще да/нет), после он включает на своём смартфоне приложение, наводит камеру на аудиторию и получает результат (рисунок 1).

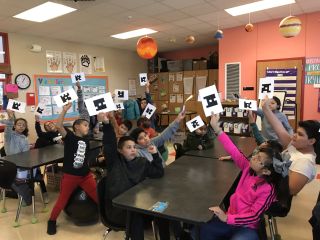


Рисунок 1 – Процесс проведения опроса с помощью приложения Plicker

Также, в число минусов данного приложения (если мы говорим о русскоязычных пользователях) можно отнести исключительно англоязычный интерфейс.

Второе приложение уже полностью удовлетворяет запросам, поставленным мною в начале данного проекта: оно имеет русскоязычный интерфейс, с его помощью можно проводить полноценное тестирование, рассчитанное на любое количество вопросов (в зависимости от распечатанных бланков), включая вопросы с несколькими вариантами ответа. В ZipGrade результаты проведённого тестирования выводятся на экран смартфона в процентах либо в баллах, которые после можно сохранить в отдельный файл (рисунок 2).

Существенным недостатком данного мобильного приложения является тот факт, что оно условно бесплатное, то есть в месяц каждому авторизированному пользователю выделяется по 100 проверок (не тестирований, а именно проверок). Цена не сильно кусается – годовая подписка составляет 6.99$.

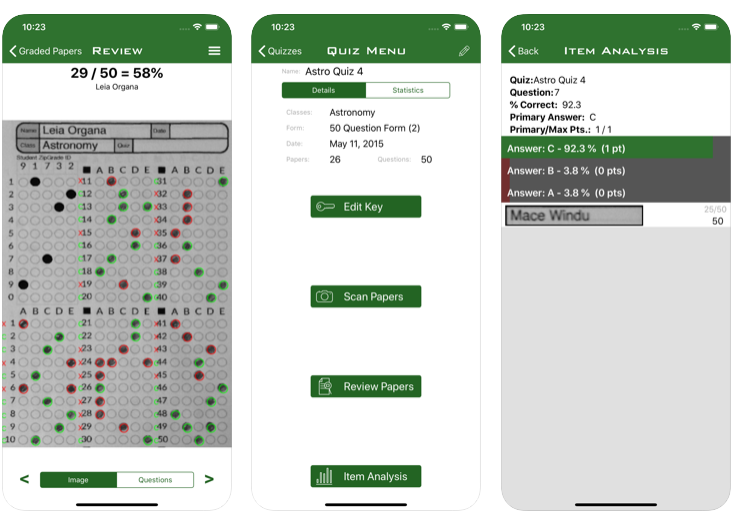


Рисунок 2 – Интерфейс приложения ZipGrade

1.3 Современные технологии в области машинного зрения

Машинное зрение – это научное направление в области искусственного интеллекта, в частности робототехники, и связанные с ним технологии получения изображений объектов реального мира, их обработки и использования, полученных данных для решения разного рода прикладных задач без участия (полного или частичного) человека [11].

Системы машинного зрения запрограммированы для выполнения узкоспециализированных задач, таких как подсчет объектов на конвейере, чтение серийных номеров или поиск поверхностных дефектов. Польза системы визуальной инспекции на основе машинного зрения заключается в высокой скорости работы с увеличением оборота, возможности 24-часовой работы и точности повторяемых измерений. Также преимущество машин перед людьми заключается в отсутствии утомляемости, болезней или невнимательности. Тем не менее, люди обладают тонким восприятием в течение короткого периода и большей гибкостью в классификации и адаптации к поиску новых дефектов [17].

Компьютеры не могут «видеть» таким же образом, как это делает человек. Фотокамеры не эквивалентны системе зрения человека, и в то время как люди могут опираться на догадки и предположения, системы машинного зрения должны «видеть» путём изучения отдельных пикселей изображения, обрабатывая их и пытаясь сделать выводы с помощью базы знаний и набора функций таких, как устройство распознавания образов. Хотя некоторые алгоритмы машинного зрения были разработаны, чтобы имитировать зрительное восприятие человека, большое количество уникальных методов были разработаны для обработки изображений и определения соответствующих свойств изображения.

Хотя машинное зрение – процесс применения компьютерного зрения для промышленного применения, полезно перечислить часто используемые аппаратные и программные компоненты. Типовое решение системы машинного зрения включает в себя несколько следующих компонентов:

* одна или несколько цифровых или аналоговых камер (черно-белые или цветные) с подходящей оптикой для получения изображений;
* программное обеспечение для изготовления изображений для обработки. Для аналоговых камер это оцифровщик изображений;
* процессор (современный ПК c многоядерным процессором или встроенный процессор, например, ЦСП);
* программное обеспечение машинного зрения, которое предоставляет инструменты для разработки отдельных приложений программного обеспечения;
* оборудование ввода-вывода или каналы связи для доклада о полученных результатах;
* умная камера: одно устройство, которое включает в себя все вышеперечисленные пункты;
* очень специализированные источники света (светодиоды, люминесцентные и галогенные лампы и т. д.);
* специфичные приложения программного обеспечения для обработки изображений и обнаружения соответствующих свойств;
* датчик для синхронизации частей обнаружения (часто оптический или магнитный датчик) для захвата и обработки изображений;
* приводы определенной формы, используемые для сортировки или отбрасывания бракованных деталей.

Коммерческие пакеты программ для машинного зрения и пакеты программ с открытым исходным кодом обычно включают в себя ряд методов обработки изображений, таких как:

* счетчик пикселей: подсчитывает количество светлых или темных пикселей;
* бинаризация: преобразует изображение в серых тонах в бинарное (белые и черные пиксели);
* сегментация: используется для поиска и/или подсчета деталей:

1. поиск и анализ блобов: проверка изображения на отдельные блобы связанных пикселей (например, черной дыры на сером объекте) в виде опорных точек изображения. Эти блобы часто представляют цели для обработки, захвата или производственного брака;
2. надежное распознавание по шаблонам: поиск по шаблону объекта, который может быть повернут, частично скрыт другим объектом, или отличным по размеру)

* чтение штрих-кодов: декодирование 1D и 2D кодов, разработанных для считывания или сканирования машинами;
* оптическое распознавание символов: автоматизированное чтение текста, например, серийных номеров;
* измерение: измерение размеров объектов в дюймах или миллиметрах
* обнаружение краев: поиск краев объектов;
* сопоставление шаблонов: поиск, подбор, и/или подсчет конкретных моделей;

В большинстве случаев, системы машинного зрения используют последовательное сочетание этих методов обработки для выполнения полного инспектирования. Например, система, которая считывает штрих-код может также проверить поверхность на наличие царапин или повреждения и измерить длину и ширину обрабатываемых компонентов.

2 РАЗРАБОТКА ПРОТОТИПА ПРИЛОЖЕНИЯ ДЛЯ ПРОВЕРКИ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

2.1 Выбор среды разработки

Первым этапом рассмотрим языки программирования, позволяющие написать полноценное мобильное приложение – это Java, Kotlin для Android, Swift для iOS и C# для кроссплатформенной разработки.

Java – строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems.

Kotlin – статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains. Также компилируется в JavaScript и в исполняемый код ряда платформ через инфраструктуру LLVM.

Swift – открытый мультипарадигмальный компилируемый язык программирования общего назначения. Создан компанией Apple в первую очередь для разработчиков iOS и macOS.

C# – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework.

Было принято решение писать кроссплатформенное приложение. Значит, в данной разработке будет использоваться язык программирования C#. Далее необходимо выбрать среду разработки. Данный язык программирования можно использовать при разработке мобильных приложений в двух IDE: Xamarin Studio и Microsoft VisualStudio (начиная с версии 2016 года).

Xamarin Studio позволяет разработчикам быстро создавать настольные и веб-приложения для iOS, Android, Windows, Linux и Mac. Приложения Xamarin созданы со стандартными средствами управления пользовательским интерфейсом. Приложения не только выглядят так, как ожидает конечный пользователь, но и ведут себя так же. Xamarin Studio предлагает интегрированный отладчик для отладки моно и нативных приложений, полностью настраиваемые раскладки окон, определяемые пользователем привязки клавиш, внешние инструменты, автозавершение кода для C #, шаблоны кода и свертывание кода, поддержка C #, F #, Visual Basic .NET, C / C ++ и языки Vala, возможность создания веб-проектов ASP.net с полной поддержкой автозавершения кода и тестирование на XSP, веб-сервере Mono. Явным преимуществом данного IDE является стабильно работающий встроенный эмулятор мобильных платформ и ОС. Распространяется бесплатно.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight. Имеется возможно устанавливать дополнительные модули, плагины и фреймворки, в том числе и Xamarin. Из явных недостатков данного IDE можно выделить высокую требовательность к железу вашего ПК, нестабильно работающий эмулятор (в моём случае вместо эмулятора использовал личный смартфон), а также непомерно высокая цена, неприемлемая для рядового студента (от 45 $ в месяц до 1199 $ в год). К счастью, если пользователь скачивает VisualStudio 2019 с официального сайта не версию Professional, а любую ниже (в моём случае это Commnity), и в личном кабинете Microsoft указать себя как студент, не имеющий общего заработка 10000 $ в год, то вам лицензия будет предоставлена бесплатно.

Для выполнения данного проекта была выбрана среда разработки Microsoft VisualStudio Community 2019 с установленным фреймворком Xamarin, так как, во-первых, на моём компьютере уже стоит самая актуальная лицензионная версия VisualStudio, во-вторых, интерфейс VisualStudio более привычен, чем XamarinStudio. Мощности моего ПК хватает для спокойной работы именно в данной IDE, к тому же изначально предполагалось использования личного смартфона вместо эмулятора, поэтому проблема с медленно работающим эмулятором в VisualStudio так же отпадает. Во всех остальных основных аспектах эти IDE практически взаимозаменяемы.

2.2 Настройка среды разработки

Для реализации данного проекта необходимо установить и настроить среду разработки Microsoft VisualStudio Community 2019, выбрать «Разработка классических приложений .NET» и «C# и Visual Basic» (для непосредственной работы на языке программирования C#), «Разработка мобильных приложений на .NET» и «Xamarin» (для возможности работать с фреймворком Xamarin), «Кроссплатформенная разработка .NET Core» (для адаптации кода под различные платформы), «Пакет SDK для .NET Core» и «Windows 10 SDK» (для работы с Android), а также «Диспетчер пакетов NuGet» (для работы с дополнительными пакетами). После установки необходимо обновить .NET Framework на своей системе до актуальной версии (в данном случае – версия 4.7) (рисунки 3, 4).

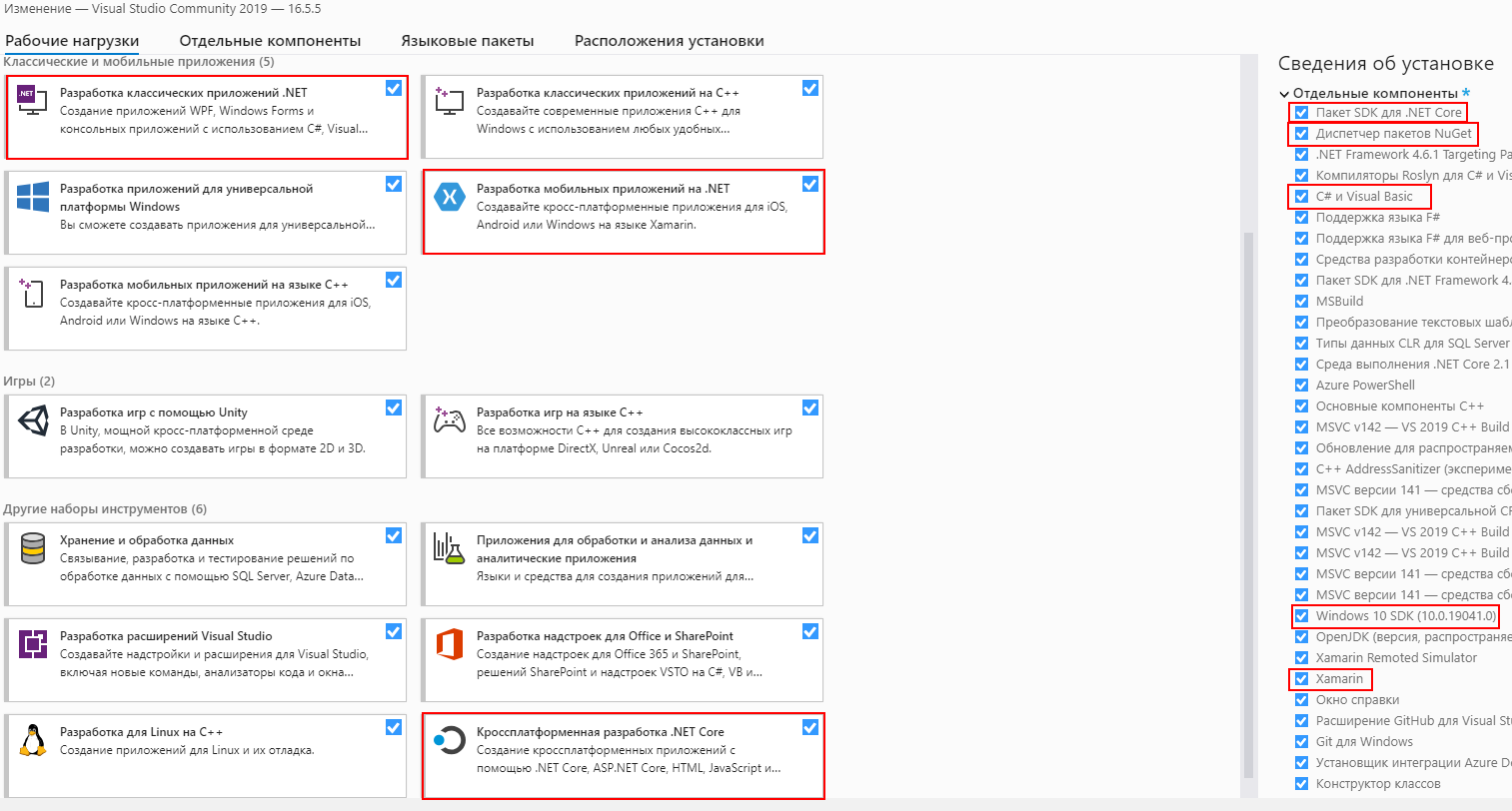


Рисунок 3 – Установка Microsoft VisualStudio Community 2019

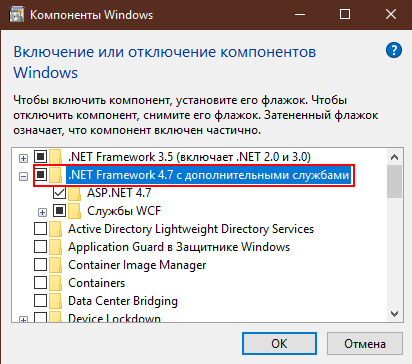


Рисунок 4 – Обновление .NET Framework

Emgu CV – кроссплатформенная оболочка для .NET библиотеки обработки изображений OpenCV [15]. Скачать Emgu CV можно либо с официального репозитория GitHub[12], либо (если говорить о работе в VisualStudio) установить пакет Emgu CV с помощью менеджера пакетов NuGet [14].

Чтобы воспользоваться пакетом Emgu.CV в проекте VisualStudio, сначала необходимо скачать пакет ZedGraph для адекватной работы объектов типа Bitmap. После уже можно установить сам пакет Emgu.CV версии 4.1.1 – это важно, так как версии ниже не совместимы с обновлёнными компонентами VisualStudio (это для тех случаев, когда лицензия официальная, и обновление VisualStudio происходит регулярно и автоматически), а последняя версия 4.2.0 не поддерживает конвертацию изображений с помощью явного преобразования Bitmap (рисунки 5, 6).

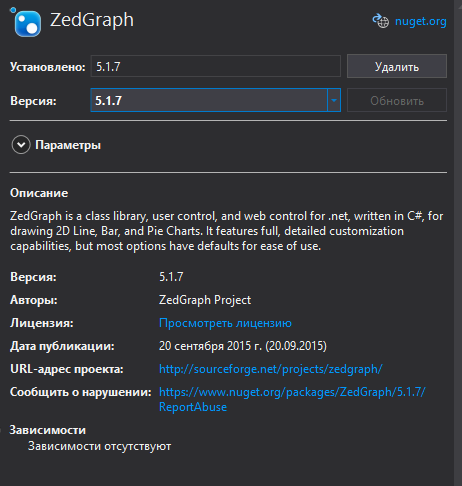


Рисунок 5 – Установка пакета ZedGraph

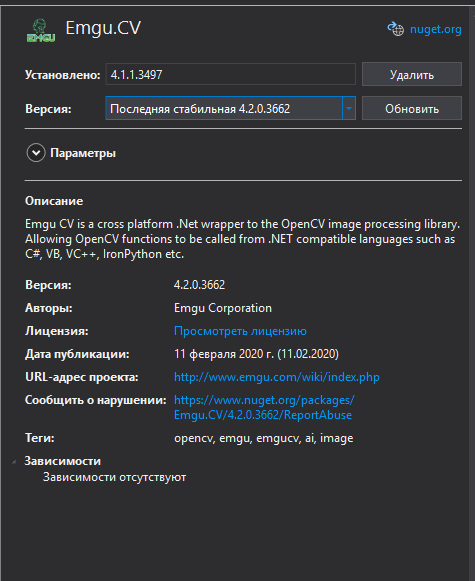


Рисунок 6 – Установка пакета Emgu.CV версии 4.1.1

Теперь необходимо в проекте подключить необходимые для работы пакета Emgu.CV директивы: Emgu.CV, Emgu.CV.Structure, Emgu.CV.Util (рисунок 7).



Рисунок 7 – Подключение директив для работы с пакетом Emgu.CV

2.3 Разработка структуры бланка ответов

Бланк ответов состоит из групп пустых кружков, предназначенных для закрашивания, подписей номеров вопросов (в данной разработке предусмотрен бланк на 20 вопросов), подписей вариантов ответов (заглавные латинские буквы, в данной разработке предусмотрены 5 вариантов ответа) и шести марок, представленных чёрными квадратами по периметру бланка (рисунок 8).

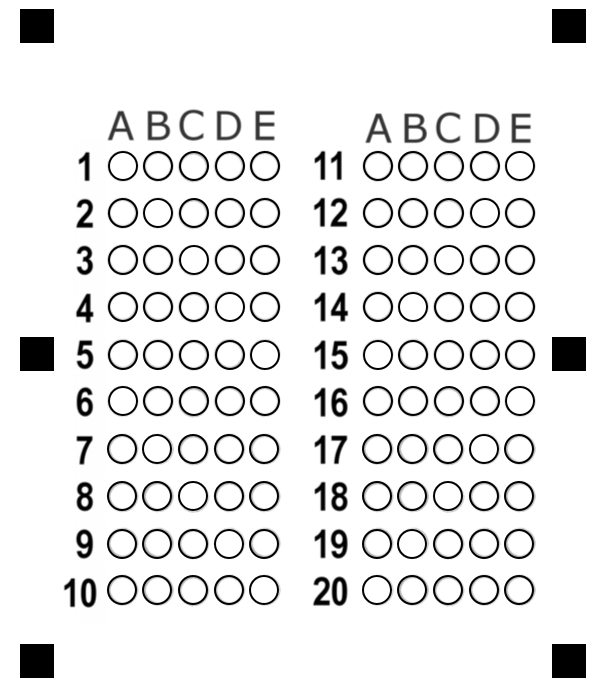


Рисунок 8 – Структура бланка ответов

Марки по периметру служат для определения границы бланка и при необходимости – кадрирования. Алгоритм определяет квадраты по количеству сторон (4 штуки) и соотношению сторон (с учётом погрешности от 0.95 до 1.05), поэтому размер марок не критичен. Периметр кружков более важен – значение периметра учитывается в алгоритме и находится в диапазоне от 104 до 110 при размере изображения 600 на 700 пикселей.

2.4 Описание процесса разработки программной части

Визуальная часть разработки состоит из двух форм: форма для выделения правильных вариантов ответов (далее по тексту «Эталон») и формы работы с изображением (далее по тексту «Сканер»). Эталон состоит из двух вертикальных (номера вопросов) и двух горизонтальных (варианты ответа) блоков с подписями в соответствии с исходным бланком.

Также, на Эталоне располагаются 20 блоков по 5 RadioButton для отметки преподавателем правильных вариантов, и кнопка «Запомнить», которая сохраняет результаты в двумерный массив quePrepod и закрывает Эталон (рисунок 9).

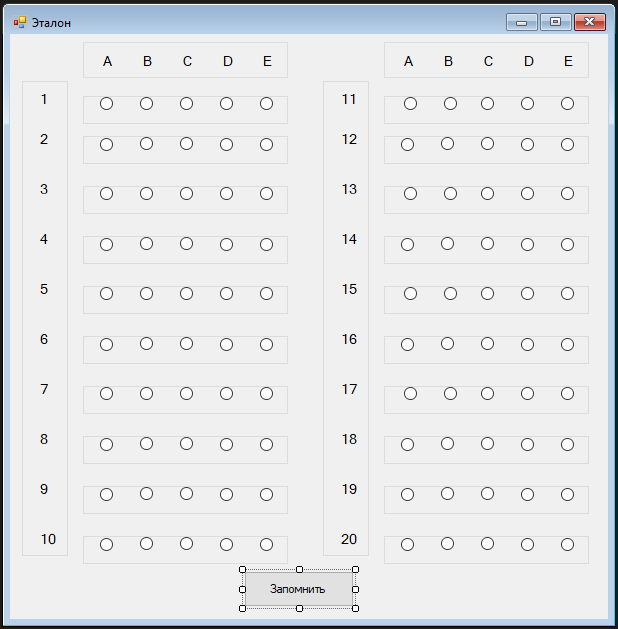


Рисунок 9 – Внешний вид формы «Эталон»

На Сканере располагаются pictureBox – первый для исходного изображения, второй для обработанного изображения – listBox для вывода списка правильных ответов, кнопка «Вывести ответы» для вывода результатов, а также меню с вкладками «Файл» и «Инструменты». Вкладка «Файл» содержит в себе кнопку «Открыть» для загрузки изображения, вкладка «Инструменты» содержит кнопку «Найти ответы» для обработки исходного изображения (рисунок 10).

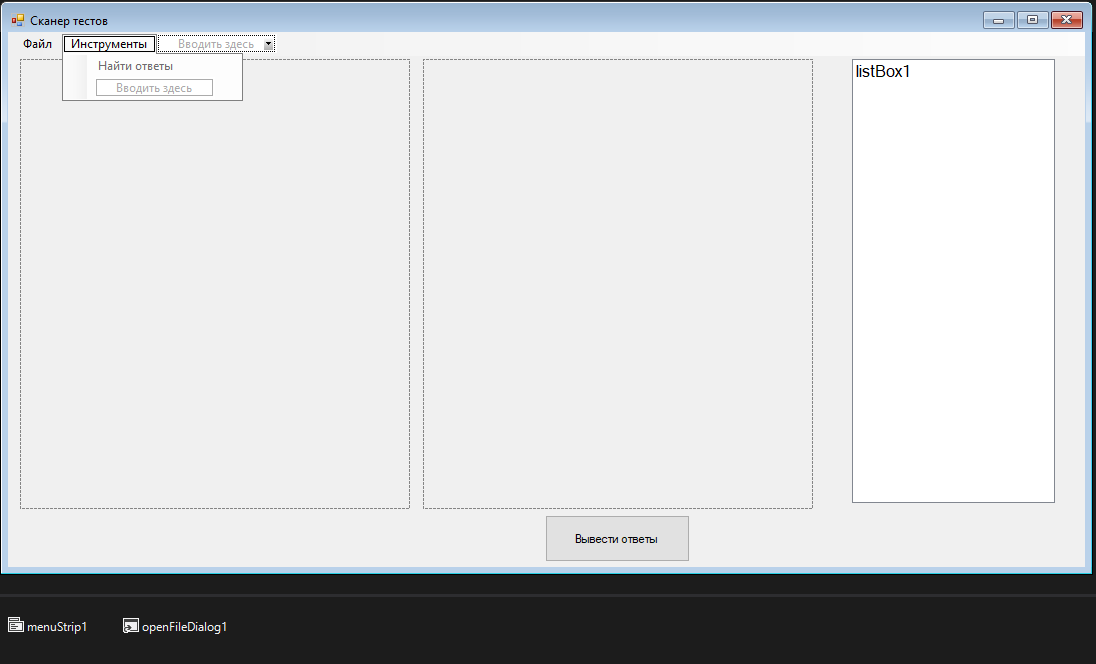


Рисунок 10 – Внешний вид формы «Сканер тестов»

При запуске программы сначала пользователю даётся возможность заполнить Эталон. После заполнения необходимо нажать кнопку «Запомнить», после чего массив quePrepod заполняется нулями, а элементы массива quePrepod[i, j] (где i – номер вопроса и j – вариант ответа) заполняются в соответствии с выбранными вариантами – если radioBitton.Checked == true, то в соответствующий элемент массива quePrepod записывается 1, если radioButton.Checked == false, то в соответствующем элементе массива quePrepod остаётся 0. После чего Эталон закроется и откроется Сканер (рисунок 11).

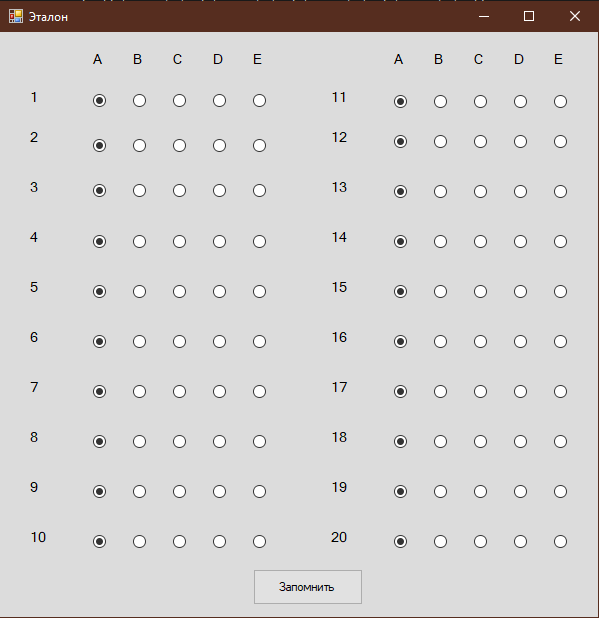


Рисунок 11 – Работа формы «Эталон»

После открытия Сканера пользователю недоступны кнопки «Вывести ответы» и «Найти ответы». Пользователю доступна только кнопка «Открыть», которая вызывает диалоговое окно открытия изображения. После загрузки изображения, оно будет отображаться в первом pictureBox, а также станет доступна кнопка «Найти ответы» (рисунок 12).



Рисунок 12 – Загрузка изображения в pictureBox1

Закруженное изображение помещается в переменную типа Image<Bgr, byte>, чтобы можно было его разбить на массив байтов отдельных цветовых составляющий (синий, зелёный, красный) и подвергается ресайзингу до размеров 600x700 с помощью команды Emgu.CV.CvEnum.Inter.Cubic. Делается это для того, чтобы под один и тот же метод подходили изображения изначально разных размеров.

После нажатия на кнопку «Найти изображение» создаётся ещё один экземпляр типа Image<>, только на этот раз не Bgr, а Gray, и с помощью Гауссово преобразования [13] происходит инверсия цветного исходного изображения в серое. Далее создаётся экземпляр типа VectorOfVectorOfPoint под именем contours для хранения всех контуров, которые будут найдены на изображении. На основе аппроксимаций линий в сером изображении с помощью команды CvInvoke.FindContours в contours записываются все найденные контура.

В цикле программа перебирает все контура, каждому по очереди с помощью команды CvInvoke.ArcLength записывается периметр и накладывается на изображение с точностью 4% от периметра (точность была подобрана вручную).

С помощью команды CvInvoke.Moments программа находит моменты контура, из которых берётся координаты центра x и y. Также, создаётся переменная типа Rectangle, чтобы определить соотношение сторон контура.

Далее идёт проверка на квадрат: если количество сторон контура равна 4 и, если соотношение сторон в пределе от 0.95 до 1.05 – это квадрат. Тогда этот контур наносится на изображение красным цветом толщиной 2.

Если контур не квадрат, то идёт проверка на круг путём сравнения периметров в пределе от 104 до 110 (цифры подобраны вручную). Берётся пиксель из центра каждого круга и сравнивается в RGBA: если цветовые составляющие R, G, и B > 240 и A = 255 – цвет можно считать белым, а круг закрашенным. Тогда контур наносится на изображение синим цветом толщиной 2. Если же круг закрашен, то контур на изображение наносится зелёным цветом толщиной 2. Далее идёт заполнение двумерного массива queStud, который изначально заполнен нулями. Центр данного контура попадает в центр зелёного контура – в массив queStud заносится 1 в необходимую позицию.

После отрисовки всех контуров, изображение выводится во второй pictureBox, на который также накладывается контур, прорисованный между центрами квадратов-меток. Изображение обрезается и кадрируется по осям x и y по данному контуру. Результат кадрирования остаётся абстрактным, то есть во втором pictureBox будет выведено оригинальное изображение с наложенными контурами. Становится доступна кнопка «Вывести ответы» (рисунок 13).

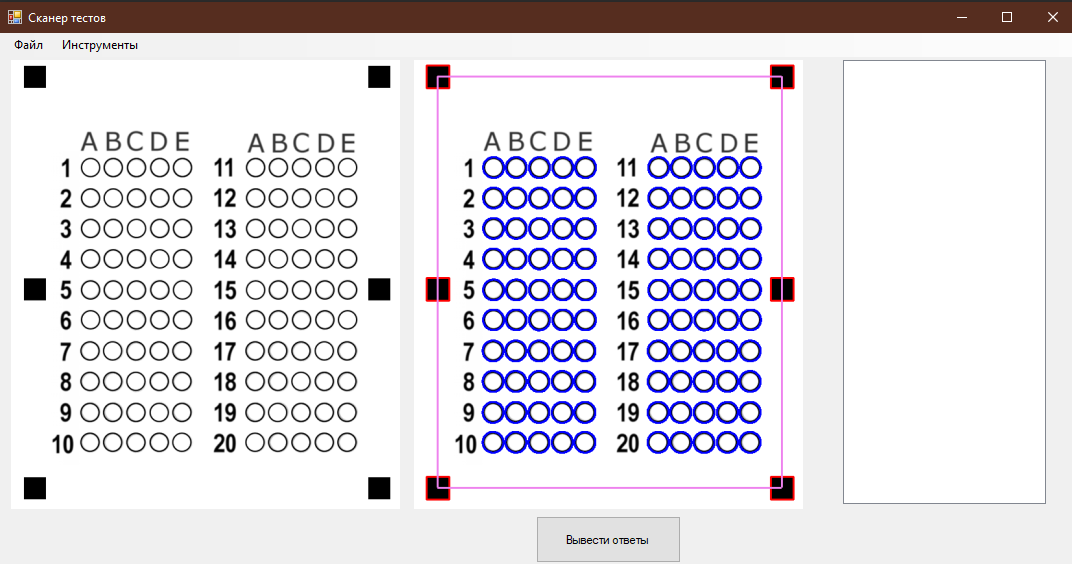


Рисунок 13 – Результат обработки изображения

При клике на кнопку «Вывести ответы» в listBox выводятся все правильные ответы, где 0 – не отмечен вариант и 1 – вариант отмечен. Два массива queStud и quePrep сравниваются, и если ответ студента совпадает с ответом преподавателя и равен 1, то к счётчику добавляется единица. В конце выводится MessageBox с информацией, сколько правильных ответов было дано (рисунок 14).

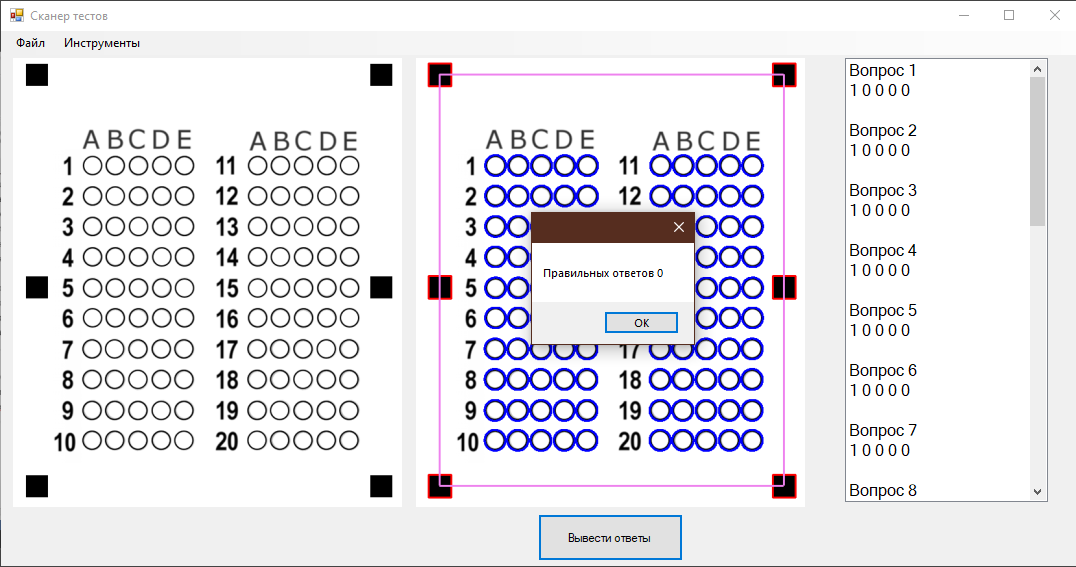


Рисунок 14 – Результат работы программы

1. ТЕСТИРОВАНИЕ И ПОДГОТОВКА ПРИЛОЖЕНИЯ К ПОРТИРОВАНИЮ НА МОБИЛЬНЫЕ ПЛАТФОРМЫ

3.1 Тестирование разработанного программного обеспечения

Сперва необходимо провести тестирование, чтобы убедиться, что правильные ответы адекватно распознаются, а результат, выводимый на экран, был реален. Для этого на Эталоне были отмечены варианты для вопросов 1-5 и 6-10 в порядке возрастания латинского алфавита от A к E, а для вопросов 11-15 и 16-20 – в обратном порядке соответственно (рисунок 15).

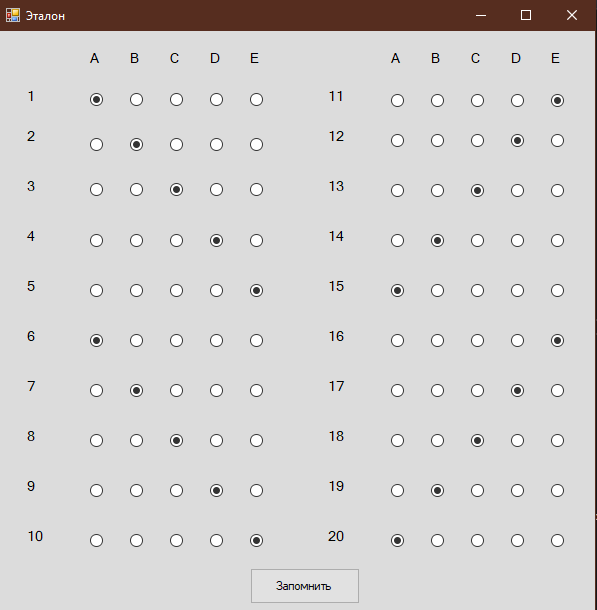


Рисунок 15 – Отмеченные варианты ответов для первого тестирования

Далее необходимо загрузить не пустой бланк, а заполненный в том же соответствии, что и выбранные ранее варианты на Эталоне (рисунок 16).

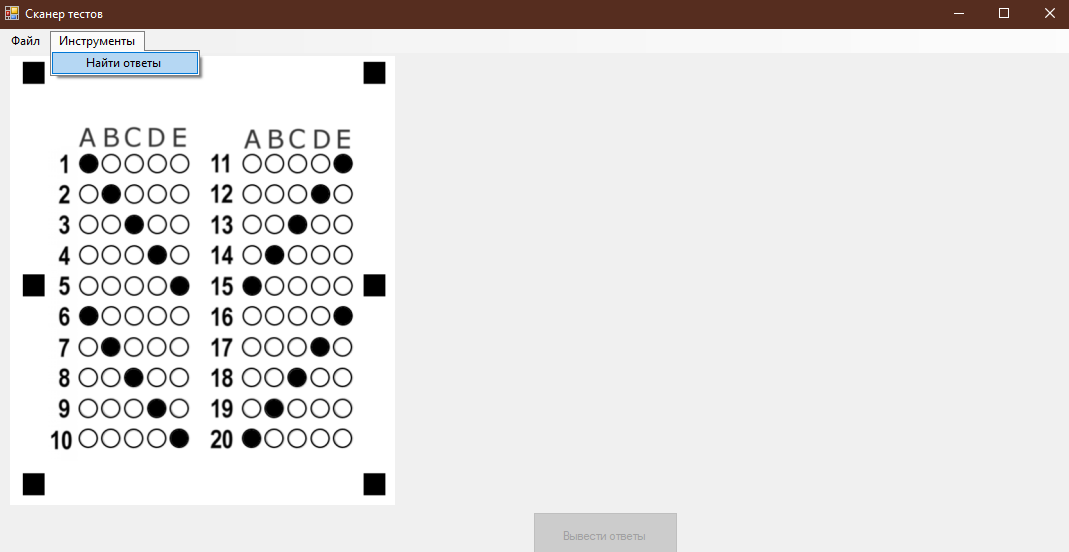


Рисунок 16 – Загрузка заполненного бланка для первого теста

Теперь, при нажатии на кнопку «Найти ответы», все закрашенные кружки будут помечены зелёным контуром, а количество правильных ответов будет равно 20 (рисунок 17).

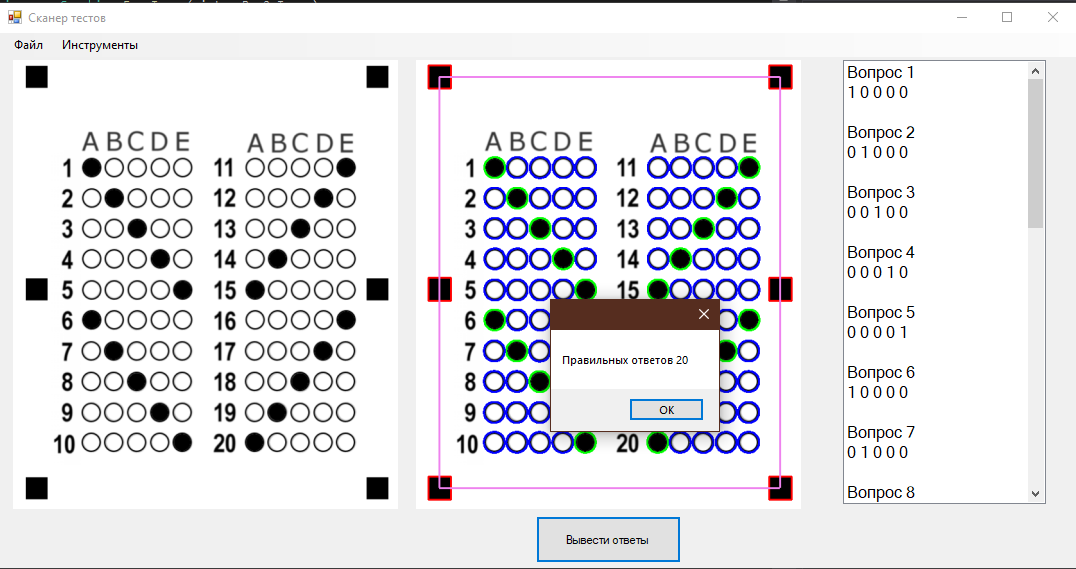


Рисунок 17 – Результат первого тестирования

Для второго тестирования необходимо внести в Эталон заведомо неверные ответы, отличные от заполненного бланка. Пусть это будет стандартный набор вариантов – для всех вопросов вариант ответа A (рисунок 11).

Для наглядности будет использоваться иначе заполненный бланк, в котором некоторые вопросы пропущены, а у некоторых отмечены несколько вариантов ответа (рисунок 18).

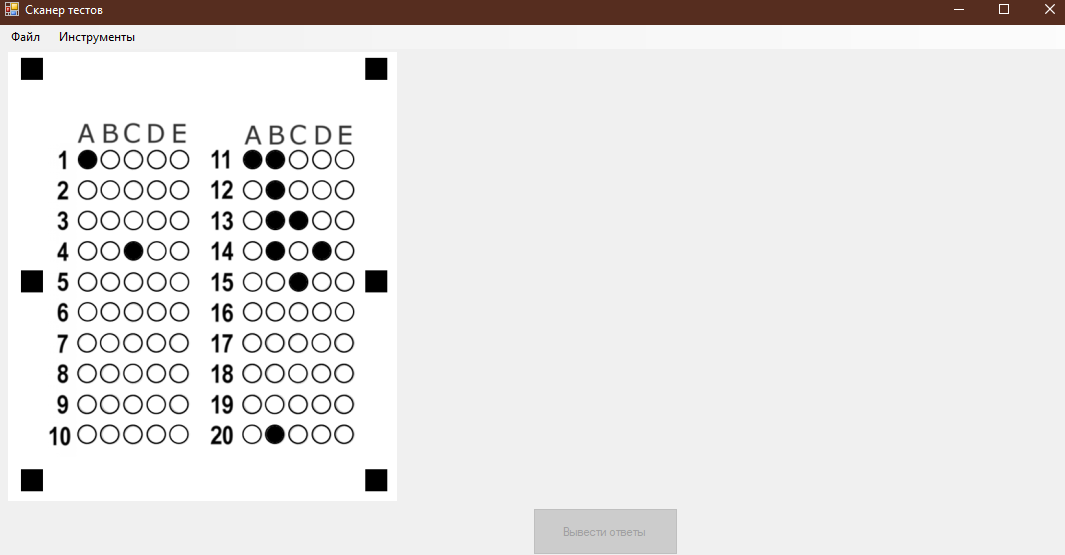


Рисунок 18 – Загрузка заполненного бланка для второго теста

Теперь, после нажатия на кнопку «Найти ответы», все закрашенные кружки также будут помечены зелёным контуром, но правильных ответов будет только 2: на 1 и 11 вопросы (рисунок 19).

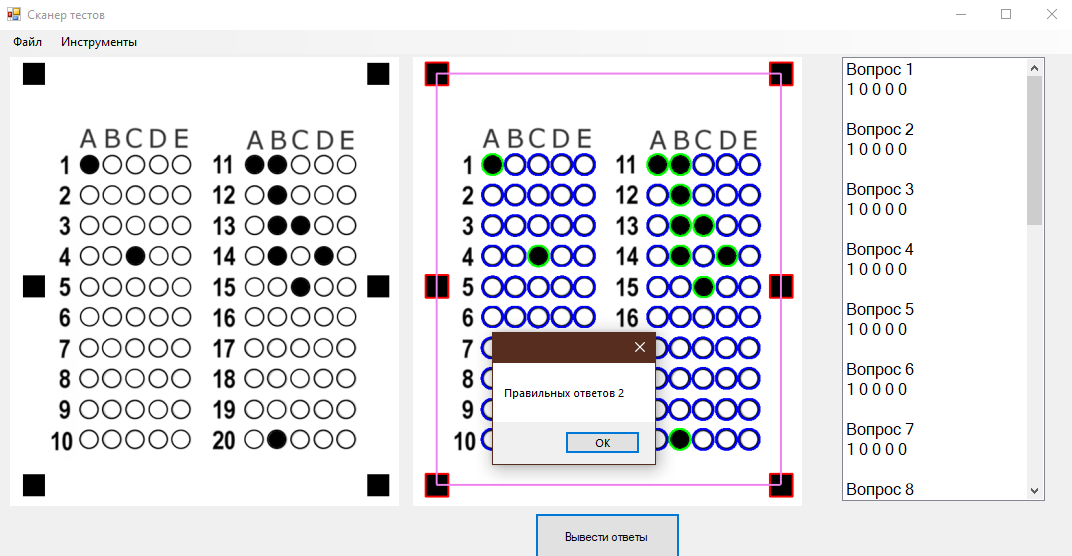


Рисунок 19 – Результат второго тестирования

Также, стоит учесть такие факторы, как неполное закрашивание кружка, использование чернил не чёрного цвета либо помарки (случайные либо намеренные) на полях бланка. Проведём третий тестовый запуск с учётом всех вышеописанных факторов (рисунок 20).

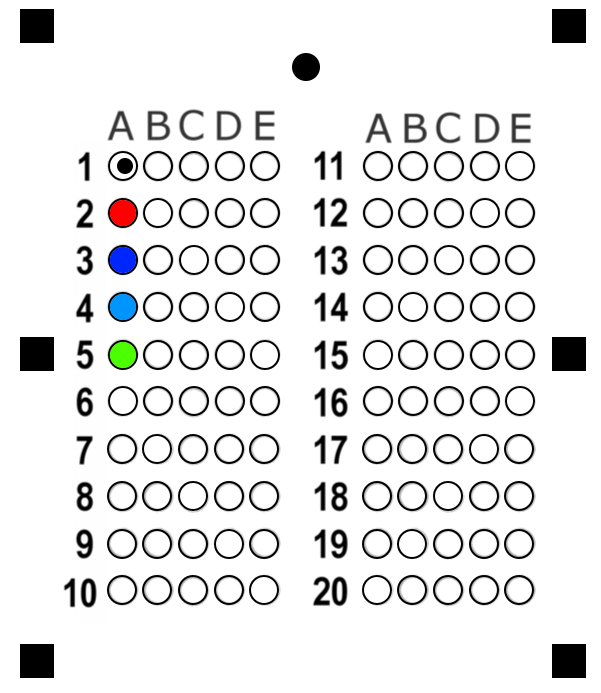


Рисунок 20 – Заполненный бланк для третьего теста

На Эталоне оставим стандартные отметки (рисунок 11). После запуска приложения и обработки изображения ожидается получить в результате 5 правильных ответов, так как отметки стоят только у пяти вопросов. В итоге получается как раз 5 правильных ответов, потому что алгоритм проверяет на «закрашенность» не весь контур, а лишь небольшую область в центре контура размером 15 на 15 пикселей. С цветными кружками также проблем не возникло, так как неотмеченным (белым) считается тот, у которого каждый составляющий элемент RGB-гаммы превышает значение 240 (с учётом некачественной цветопередачи камеры/сканера). Лишняя отметка вне поля не была засчитана за вариант ответа, так как она выходит за установленные пределы проверки алгоритма (она даже не помечается контуром) (рисунок 21).

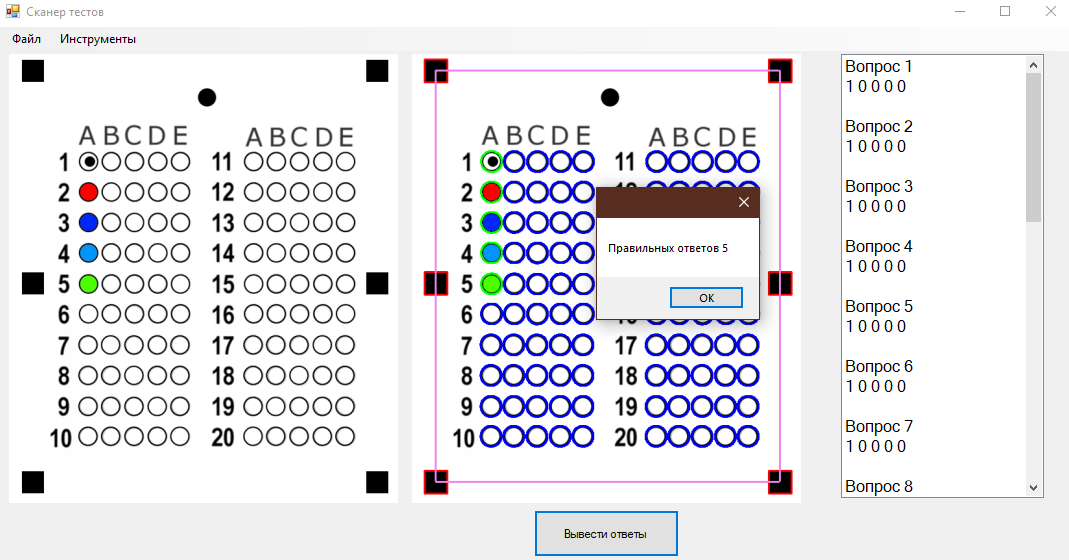


Рисунок 21 – Результат третьего тестирования

Таким образом, можно сказать, что данный прототип приложения для определения верных ответов тестирования работает корректно.

3.2 Создание прототипа Android-приложения

Для перенесения кода из десктопной версии приложения в мобильную, необходимо подготовить для этого пространство. Для начала необходимо создать мобильное приложение, используя фреймворк Xamarin, мобильное приложение, способное делать фото или выбирать необходимое изображение из галереи телефона и выгружать его на экран устройства.

Чтобы воспользоваться функциями камеры, необходимо установить пакетXam.Plugin.Media самой актуальной версии (в данном случае это версия 5.0.1). На данном этапе могут возникнуть ошибки, так как стандартный проект Xamarin.Forms создаётся на базе необновлённых пакетов Xamarin.Essentials и Xamarin.Forms, их заблаговременно необходимо обновить до самой актуальной версии через медеджер пакетов NuGet (рисунок 22).

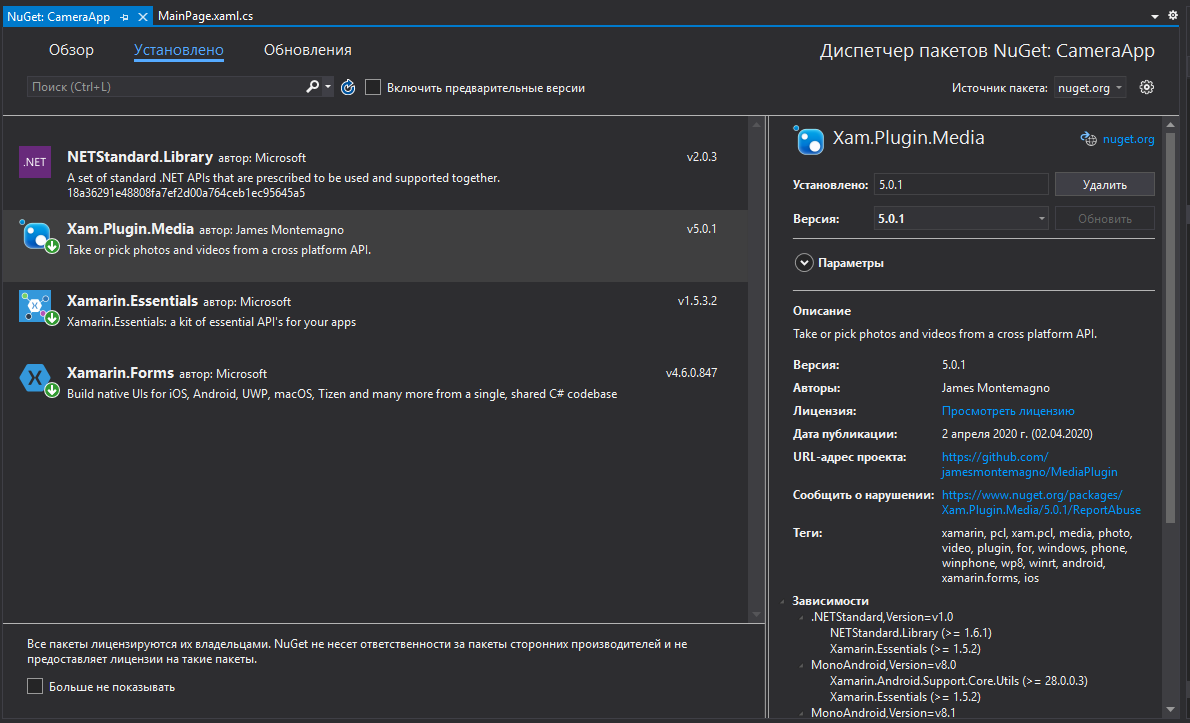


Рисунок 22 – Обновлённые стандартные пакеты и установленный пакет   
NuGet Xam.Plugin.Media

Далее, следуя тексту из открывшегося файла readme.txt необходимо выполнить следующие шаги: в файл AndroidManifest.xml внести изменения в тег <application>, создать в папке Android/Resource директорию xml, в ней создать файл file\_paths.xml и наполнить его в соответствии с текстом из readme.txt (рисунок 23, 24).

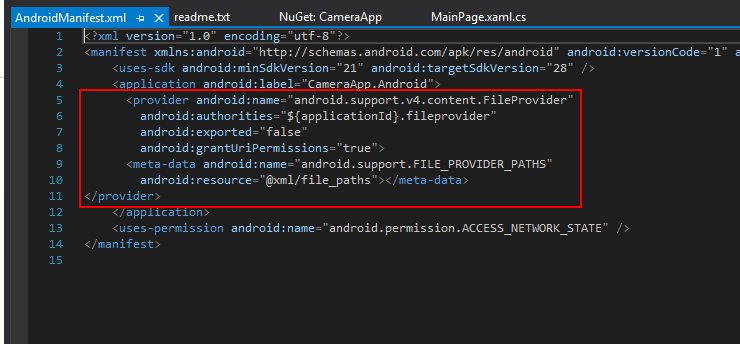


Рисунок 23 – Внесённые изменения в файл AndroidManifest.xml

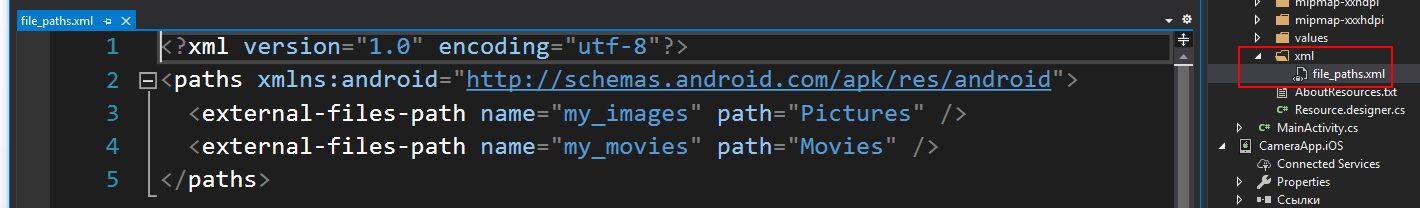


Рисунок 24 – Созданная директории xml, файл file\_paths.xml и его наполнение

Теперь необходимо подключить директивы пакета Xam.Plugin.Media для работы с камерой и галереей телефона (рисунок 25).

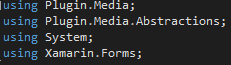


Рисунок 25 – Подключение директив пакета Xam.Plugin.Media

Для прототипа мобильного приложения хватит двух кнопок – фото и взятие изображения из галереи устройства – и самого объекта типа Image для вывода изображения на экран телефона. Подпись к кнопкам будет соответственно «Сделать фото» и «Выбрать фото», объект типа Image сначала будет невидим.

Функции для каждой из кнопок будут асинхронными, чтобы каждый раз фото или взятие из галереи устройства изображения не останавливала всю работу устройства. Для взятия изображения из галереи устройства функция будет создавать новый объект типа MediaFile и с помощью команды CrossMedia.Current.PickPhotoAsync() помещать в него выбранное из галереи устройства изображения, после чего помещать его в изображения, выводя картинку на экран телефона.

Функция, позволяющая сделать фото при помощи камеры устройства, также создаёт новый объект типа MediaFile, задаёт ему разрешение на сохраниение в альбом (SaveToAlbom = true), указывает на директорию либо создаёт новую (Directory = “Sample”), задаёт имя файлу (в данном случае именем служит настоящее время, благодаря использованию DateTime.Now.ToString()) и также выводит изображение на экран.

Осталось только определить эти три визуальных объекта на экране телефона в классе Content и расположить их согласно задумке (обе кнопки в одном StackLayout в горизонтальном положении и изображение по центру ниже кнопок). Приложение отлично работает и снимает на камеру либо берёт изображение из галереи устройства (рисунок 26, 27).

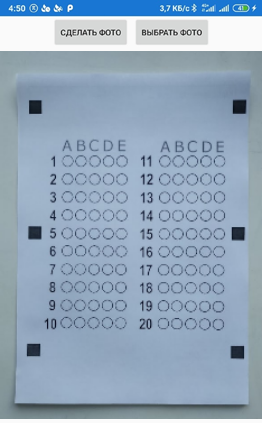


Рисунок 26 – Фото, сделанное на основную камеру

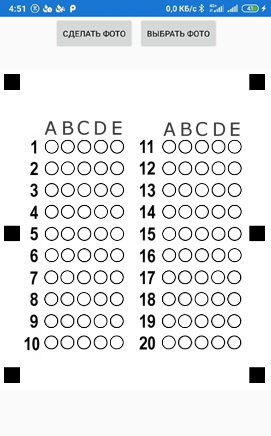


Рисунок 27 – Фото, выбранное из галереи устройства

Основной проблемой переноса метода распознавания фигур и, главное, вывода результатов тестирования стала оболочка Emgu.CV. Вся библиотека OpenCV (библиотека машинного зрения для промышленного использования) для десктопных приложений распространяется бесплатно на любой поддерживаемый язык программирования. Для мобильной разработки на основе Xamarin на .NET-технологии существует особая оболочка Emgu CV, которая также распространяется бесплатно для десктопных разработок. Для мобильных разработок данный пакет запрашивает покупку коммерческой лицензии в размере 399 $ единым платежом (не месячная подписка), что по актуальному курсу примерно 27500 рублей.

ЗАКЛЮЧЕНИЕ

В соответствии с поставленной целью, в рамках выпускной квалификационной работы была разработана автоматизированная система проверки тестирования с использованием технологии машинного зрения на базе обёртки Emgu CV библиотеки OpenCV. В дальнейшем планируется портировать данное приложение на мобильные платформы типа Android и IOS. Это позволит проверяющему ещё быстрее получить результаты тестирования, не прибегая к переносу фотографий с телефона на ПК или к использованию сканера.

* ходе выполнения работы были усовершенствованы навыки работы VisualStudio 2019, языка программирования C# в общем и Window.Forms.C# в частности, а также приобретены навыки работы с библиотекой, использующей технологию машинного зрения.

В результате выполнения работы были решены следующие задачи:

* выполнен обзор предметной области;
* выбраны программные средства реализации;
* подготовлен типовой бланк ответов;
* разработан и протестирован прототип приложения;
* выполнена подготовка к портированию приложения на мобильные платформы.

Таким образом, цель выпускной квалификационной работы была достигнута. Приложение применимо для преподавателей и иных проводящих опросы и тестирования лиц, имеющих возможность распечатать типовые бланки, перенести с памяти устройства фото на компьютер или воспользоваться сканером.

СПИСОК ЛИТЕРАТУРЫ

1. Бондарчук, М. М. Применение компьютерной диагностики для оценки качества освоения образовательных программ [Текст] / М. М. Бондарчук, В. Е. Грязнова. – Тамбов : Альманах современной науки и образования, 2014. – 90 с.
2. Василькина, К. В. Создание тестирующих приложений с использованием объектно-ориентированных языков программирования [Текст] / К. В. Василькина. – Ульяновск : сб. материалов Наука через призму времени Всероссийский молодёжный фестиваль, 2015. – 460 с.
3. Гаврилов, В. В. Опыт применения информационных технологий в процессе обучения [Текст] / В. В. Гаврилов. – М. : Вестник Московского финансово-юридического университета, 2013. – 280 с.
4. Густяхина В. П. Программы-конструкторы тестов как инструмент контроля знаний [Текст] / В. П. Густяхина. – Новокузнецк : Информационно-коммуникационные технологии в педагогическом образовании, 2018. – 182 с.
5. Джанибекова, К. Р. Использование технологии компьютерного тестирования в учебном процессе [Текст] / К. Р. Джанибекова, Г. И. Шевченко. – Ставрополь : сб. материалов IX Всерос. науч.-технич. конф. : Студенческая наука для развития информационного общества, 2019. – 390 с.
6. Дмитриев, В. Л. Автоматизированная система контроля знаний на основе использования блочной разметки теста [Текст] / В. Л. Дмитриев. – Тольятти : Информационные технологии. Радиоэлектроника. Телекоммуникации, 2015. – 296 с.
7. Егорова, Ю. Н. Программа-тест как средство контроля и самоконтроля знаний бакалавров [Текст] / Ю. Н. Егорова, О. А. Егорова. – Чебоксары : сб. материалов Всероссийской с международным участием науч.-практич. конф. : Интернет-технологии в образовании, 2015. – 180 с.
8. Магомадова, З. С. Использование информационных систем в образовательном пространстве современной средней школы [Текст] / З. С. Магомедова. – Уфа : сб. статей Междунар. науч.-практ. конф. : Совершенствование методологии и организации научных исследований в целях развития общества, 2020. – 210 с.
9. Магомадова, З. С. Понятия о компьютерном тестовом контроле знаний [Текст] // З. С. Магомедова. – Ижевск : сб. статей Междунар. науч.-практ. конф. : Человеческий капитал как фактор инновационного развития общества, 2020. – 160 с.
10. Покатилов, А. Е. Применение информационных технологий при контроле знаний студентов [Текст] / А. Е. Покатилов, В. Н. Попов. – Кемерово : сб. материалов Междунар. науч.-практич. конф. «Современные тенденции развития науки и производства» : в 4-х томах. Западно-Сибирский научный центр, Кузбасский государственный технический университет имени Т.Ф. Горбачева, Международный научно-образовательный центр КузГТУ-Arena Multimedia, 2014. – 270 с.
11. Пузырева, Г. А. Разработка тестовых программ для контроля качества знаний студентов при изучении графических дисциплин [Текст] / Г. А. Пузырева, И. И. Михеев, Т. П. Кузнецова. – Тверь : сб. материалов докладов науч.-методич. конф. : Опыт и перспективы создания модульно-рейтинговой системы в ТГТУ, 2008. – 130 с.
12. GitHub: платформа для разработчиков и разработки [Электронный ресурс]. – Режим доступа : https://github.com/opencv/opencv, свободный. – Загл. с экрана.
13. METANIT.COM: Сайт о программировании [Электронный ресурс]. – Режим доступа : https://metanit.com/sharp/xamarin, свободный. – Загл. с экрана.
14. Microsoft Docs: хранилище документации Microsoft для пользователей, разработчиков и IT-специалистов [Электронный ресурс]. – Режим доступа : https://docs.microsoft.com/ru-ru/xamarin, свободный. – Загл. с экрана.
15. OpenCV: сайт open source библиотеки программного обеспечения для машинного зрения с открытым исходным кодом [Электронный ресурс]. – Режим доступа : https://opencv.org, свободный. – Загл. с экрана.
16. Plickers: сайт программного продукта [Электронный ресурс]. – Режим доступа : https://get.plickers.com, свободный. – Загл. с экрана.
17. RoboCraft: Сообщество любителей робототехники, электроники и программирования [Электронный ресурс]. – Режим доступа : http://robocraft.ru/blog/computervision/575.html, свободный. – Загл. с экрана.
18. ZipGrade: сайт программного продукта [Электронный ресурс]. – Режим доступа : https://www.zipgrade.com, свободный. – Загл. с экрана.

ПРИЛОЖЕНИЕ A

(обязательное)

ПРОГРАММНЫЙ КОД

Программный код глобального класса Program.cs:

using System;

using System.Windows.Forms;

namespace ShapeRecognision

{

static class Program

{

public static bool res = false;

public static int[,] quePrepod = new int[20, 5];

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Que());

Application.Run(new Form1());

}

}

}

Программный код Эталона Que.cs:

using System;

using System.Windows.Forms;

namespace ShapeRecognision

{

public partial class Que : Form

{

public Que()

{

InitializeComponent();

}

public void button1\_Click(object sender, EventArgs e)

{

Program.res = true;

for (int i = 0; i < 20; i++)

for (int j = 0; j < 5; j++)

Program.quePrepod[i, j] = 0;

for (int i = 0; i < 20; i++)

for (int j = 0; j < 5; j++)

{

//Вопрос 1 вариант 1

if (radioButton1.Checked == true)

Program.quePrepod[0, 0] = 1;

//Вопрос 1 вариант 2

if (radioButton2.Checked == true)

Program.quePrepod[0, 1] = 1;

//Вопрос 1 вариант 3

if (radioButton3.Checked == true)

Program.quePrepod[0, 2] = 1;

//Вопрос 1 вариант 4

if (radioButton4.Checked == true)

Program.quePrepod[0, 3] = 1;

//Вопрос 1 вариант 5

if (radioButton5.Checked == true)

Program.quePrepod[0, 4] = 1;

//Вопрос 2 вариант 1

if (radioButton6.Checked == true)

Program.quePrepod[1, 0] = 1;

//Вопрос 2 вариант 2

if (radioButton7.Checked == true)

Program.quePrepod[1, 1] = 1;

//Вопрос 2 вариант 3

if (radioButton8.Checked == true)

Program.quePrepod[1, 2] = 1;

//Вопрос 2 вариант 4

if (radioButton9.Checked == true)

Program.quePrepod[1, 3] = 1;

//Вопрос 2 вариант 5

if (radioButton10.Checked == true)

Program.quePrepod[1, 4] = 1;

//Вопрос 3 вариант 1

if (radioButton11.Checked == true)

Program.quePrepod[2, 0] = 1;

//Вопрос 3 вариант 2

if (radioButton12.Checked == true)

Program.quePrepod[2, 1] = 1;

//Вопрос 3 вариант 3

if (radioButton13.Checked == true)

Program.quePrepod[2, 2] = 1;

//Вопрос 3 вариант 4

if (radioButton14.Checked == true)

Program.quePrepod[2, 3] = 1;

//Вопрос 3 вариант 5

if (radioButton15.Checked == true)

Program.quePrepod[2, 4] = 1;

//Вопрос 4 вариант 1

if (radioButton16.Checked == true)

Program.quePrepod[3, 0] = 1;

//Вопрос 4 вариант 2

if (radioButton17.Checked == true)

Program.quePrepod[3, 1] = 1;

//Вопрос 4 вариант 3

if (radioButton18.Checked == true)

Program.quePrepod[3, 2] = 1;

//Вопрос 4 вариант 4

if (radioButton19.Checked == true)

Program.quePrepod[3, 3] = 1;

//Вопрос 4 вариант 5

if (radioButton20.Checked == true)

Program.quePrepod[3, 4] = 1;

//Вопрос 5 вариант 1

if (radioButton21.Checked == true)

Program.quePrepod[4, 0] = 1;

//Вопрос 5 вариант 2

if (radioButton22.Checked == true)

Program.quePrepod[4, 1] = 1;

//Вопрос 5 вариант 3

if (radioButton23.Checked == true)

Program.quePrepod[4, 2] = 1;

//Вопрос 5 вариант 4

if (radioButton24.Checked == true)

Program.quePrepod[4, 3] = 1;

//Вопрос 5 вариант 5

if (radioButton25.Checked == true)

Program.quePrepod[4, 4] = 1;

//Вопрос 6 вариант 1

if (radioButton26.Checked == true)

Program.quePrepod[5, 0] = 1;

//Вопрос 6 вариант 2

if (radioButton27.Checked == true)

Program.quePrepod[5, 1] = 1;

//Вопрос 6 вариант 3

if (radioButton28.Checked == true)

Program.quePrepod[5, 2] = 1;

//Вопрос 6 вариант 4

if (radioButton29.Checked == true)

Program.quePrepod[5, 3] = 1;

//Вопрос 6 вариант 5

if (radioButton30.Checked == true)

Program.quePrepod[5, 4] = 1;

//Вопрос 7 вариант 1

if (radioButton31.Checked == true)

Program.quePrepod[6, 0] = 1;

//Вопрос 7 вариант 2

if (radioButton32.Checked == true)

Program.quePrepod[6, 1] = 1;

//Вопрос 7 вариант 3

if (radioButton33.Checked == true)

Program.quePrepod[6, 2] = 1;

//Вопрос 7 вариант 4

if (radioButton34.Checked == true)

Program.quePrepod[6, 3] = 1;

//Вопрос 7 вариант 5

if (radioButton35.Checked == true)

Program.quePrepod[6, 4] = 1;

//Вопрос 8 вариант 1

if (radioButton36.Checked == true)

Program.quePrepod[7, 0] = 1;

//Вопрос 8 вариант 2

if (radioButton37.Checked == true)

Program.quePrepod[7, 1] = 1;

//Вопрос 8 вариант 3

if (radioButton38.Checked == true)

Program.quePrepod[7, 2] = 1;

//Вопрос 8 вариант 4

if (radioButton39.Checked == true)

Program.quePrepod[7, 3] = 1;

//Вопрос 8 вариант 5

if (radioButton40.Checked == true)

Program.quePrepod[7, 4] = 1;

//Вопрос 9 вариант 1

if (radioButton41.Checked == true)

Program.quePrepod[8, 0] = 1;

//Вопрос 9 вариант 2

if (radioButton42.Checked == true)

Program.quePrepod[8, 1] = 1;

//Вопрос 9 вариант 3

if (radioButton43.Checked == true)

Program.quePrepod[8, 2] = 1;

//Вопрос 9 вариант 4

if (radioButton44.Checked == true)

Program.quePrepod[8, 3] = 1;

//Вопрос 9 вариант 5

if (radioButton45.Checked == true)

Program.quePrepod[8, 4] = 1;

//Вопрос 10 вариант 1

if (radioButton46.Checked == true)

Program.quePrepod[9, 0] = 1;

//Вопрос 10 вариант 2

if (radioButton47.Checked == true)

Program.quePrepod[9, 1] = 1;

//Вопрос 10 вариант 3

if (radioButton48.Checked == true)

Program.quePrepod[9, 2] = 1;

//Вопрос 10 вариант 4

if (radioButton49.Checked == true)

Program.quePrepod[9, 3] = 1;

//Вопрос 10 вариант 5

if (radioButton50.Checked == true)

Program.quePrepod[9, 4] = 1;

//Вопрос 11 вариант 1

if (radioButton51.Checked == true)

Program.quePrepod[10, 0] = 1;

//Вопрос 11 вариант 2

if (radioButton52.Checked == true)

Program.quePrepod[10, 1] = 1;

//Вопрос 11 вариант 3

if (radioButton53.Checked == true)

Program.quePrepod[10, 2] = 1;

//Вопрос 11 вариант 4

if (radioButton54.Checked == true)

Program.quePrepod[10, 3] = 1;

//Вопрос 11 вариант 5

if (radioButton55.Checked == true)

Program.quePrepod[10, 4] = 1;

//Вопрос 12 вариант 1

if (radioButton56.Checked == true)

Program.quePrepod[11, 0] = 1;

//Вопрос 12 вариант 2

if (radioButton57.Checked == true)

Program.quePrepod[11, 1] = 1;

//Вопрос 12 вариант 3

if (radioButton58.Checked == true)

Program.quePrepod[11, 2] = 1;

//Вопрос 12 вариант 4

if (radioButton59.Checked == true)

Program.quePrepod[11, 3] = 1;

//Вопрос 12 вариант 5

if (radioButton60.Checked == true)

Program.quePrepod[11, 4] = 1;

//Вопрос 13 вариант 1

if (radioButton61.Checked == true)

Program.quePrepod[12, 0] = 1;

//Вопрос 13 вариант 2

if (radioButton62.Checked == true)

Program.quePrepod[12, 1] = 1;

//Вопрос 13 вариант 3

if (radioButton63.Checked == true)

Program.quePrepod[12, 2] = 1;

//Вопрос 13 вариант 4

if (radioButton64.Checked == true)

Program.quePrepod[12, 3] = 1;

//Вопрос 13 вариант 5

if (radioButton65.Checked == true)

Program.quePrepod[12, 4] = 1;

//Вопрос 14 вариант 1

if (radioButton66.Checked == true)

Program.quePrepod[13, 0] = 1;

//Вопрос 14 вариант 2

if (radioButton67.Checked == true)

Program.quePrepod[13, 1] = 1;

//Вопрос 14 вариант 3

if (radioButton68.Checked == true)

Program.quePrepod[13, 2] = 1;

//Вопрос 14 вариант 4

if (radioButton69.Checked == true)

Program.quePrepod[13, 3] = 1;

//Вопрос 14 вариант 5

if (radioButton70.Checked == true)

Program.quePrepod[13, 4] = 1;

//Вопрос 15 вариант 1

if (radioButton71.Checked == true)

Program.quePrepod[14, 0] = 1;

//Вопрос 15 вариант 2

if (radioButton72.Checked == true)

Program.quePrepod[14, 1] = 1;

//Вопрос 15 вариант 3

if (radioButton73.Checked == true)

Program.quePrepod[14, 2] = 1;

//Вопрос 15 вариант 4

if (radioButton74.Checked == true)

Program.quePrepod[14, 3] = 1;

//Вопрос 15 вариант 5

if (radioButton75.Checked == true)

Program.quePrepod[14, 4] = 1;

//Вопрос 16 вариант 1

if (radioButton76.Checked == true)

Program.quePrepod[15, 0] = 1;

//Вопрос 16 вариант 2

if (radioButton77.Checked == true)

Program.quePrepod[15, 1] = 1;

//Вопрос 16 вариант 3

if (radioButton78.Checked == true)

Program.quePrepod[15, 2] = 1;

//Вопрос 16 вариант 4

if (radioButton79.Checked == true)

Program.quePrepod[15, 3] = 1;

//Вопрос 16 вариант 5

if (radioButton80.Checked == true)

Program.quePrepod[15, 4] = 1;

//Вопрос 17 вариант 1

if (radioButton81.Checked == true)

Program.quePrepod[16, 0] = 1;

//Вопрос 17 вариант 2

if (radioButton82.Checked == true)

Program.quePrepod[16, 1] = 1;

//Вопрос 17 вариант 3

if (radioButton83.Checked == true)

Program.quePrepod[16, 2] = 1;

//Вопрос 17 вариант 4

if (radioButton84.Checked == true)

Program.quePrepod[16, 3] = 1;

//Вопрос 17 вариант 5

if (radioButton85.Checked == true)

Program.quePrepod[16, 4] = 1;

//Вопрос 18 вариант 1

if (radioButton86.Checked == true)

Program.quePrepod[17, 0] = 1;

//Вопрос 18 вариант 2

if (radioButton87.Checked == true)

Program.quePrepod[17, 1] = 1;

//Вопрос 18 вариант 3

if (radioButton88.Checked == true)

Program.quePrepod[17, 2] = 1;

//Вопрос 18 вариант 4

if (radioButton89.Checked == true)

Program.quePrepod[17, 3] = 1;

//Вопрос 18 вариант 5

if (radioButton90.Checked == true)

Program.quePrepod[17, 4] = 1;

//Вопрос 19 вариант 1

if (radioButton91.Checked == true)

Program.quePrepod[18, 0] = 1;

//Вопрос 19 вариант 2

if (radioButton92.Checked == true)

Program.quePrepod[18, 1] = 1;

//Вопрос 19 вариант 3

if (radioButton93.Checked == true)

Program.quePrepod[18, 2] = 1;

//Вопрос 19 вариант 4

if (radioButton94.Checked == true)

Program.quePrepod[18, 3] = 1;

//Вопрос 19 вариант 5

if (radioButton95.Checked == true)

Program.quePrepod[18, 4] = 1;

//Вопрос 20 вариант 1

if (radioButton96.Checked == true)

Program.quePrepod[19, 0] = 1;

//Вопрос 20 вариант 2

if (radioButton97.Checked == true)

Program.quePrepod[19, 1] = 1;

//Вопрос 20 вариант 3

if (radioButton98.Checked == true)

Program.quePrepod[19, 2] = 1;

//Вопрос 20 вариант 4

if (radioButton99.Checked == true)

Program.quePrepod[19, 3] = 1;

//Вопрос 20 вариант 5

if (radioButton100.Checked == true)

Program.quePrepod[19, 4] = 1;

}

this.Close();

}

}

}

Программный код Сканера Form1.cs:

using System;

using System.Drawing;

using System.Windows.Forms;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.Util;

namespace ShapeRecognision

{

public partial class Form1 : Form

{

private Image<Bgr, byte> inputImage = null;

Point[] points = new Point[6];

int[,] queStud = new int[20, 5];

int sum = 0;

public Form1()

{

InitializeComponent();

}

private void открытьToolStripMenuItem\_Click(object sender, EventArgs e)

{

DialogResult res = openFileDialog1.ShowDialog();

if (res == DialogResult.OK)

{

inputImage = new Image<Bgr, byte>(openFileDialog1.FileName).Resize(600, 700, Emgu.CV.CvEnum.Inter.Cubic, true);

pictureBox1.Image = inputImage.Bitmap;

}

}

public void найтиФигурыToolStripMenuItem\_Click(object sender, EventArgs e)

{

Image<Gray, byte> grayImage = inputImage.SmoothGaussian(5).Convert<Gray, byte>().ThresholdBinaryInv(new Gray(230), new Gray(255));

VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();

Mat hierarchy = new Mat();

Color center;

for (int i = 0; i < 20; i++)

for (int k = 0; k < 5; k++)

queStud[i, k] = 0;

CvInvoke.FindContours(grayImage, contours, hierarchy, Emgu.CV.CvEnum.RetrType.External, Emgu.CV.CvEnum.ChainApproxMethod.ChainApproxSimple);

int j = 0;

for (int i = 0; i < contours.Size; i++)

{

double perimeter = CvInvoke.ArcLength(contours[i], true);

VectorOfPoint approximation = new VectorOfPoint();

CvInvoke.ApproxPolyDP(contours[i], approximation, 0.04 \* perimeter, true);

Moments moments = CvInvoke.Moments(contours[i]);

int x = (int)(moments.M10 / moments.M00);

int y = (int)(moments.M01 / moments.M00);

Rectangle rect = CvInvoke.BoundingRectangle(contours[i]);

double aspectRatio = (double)rect.Width / (double)rect.Height;

if (approximation.Size == 4 && (aspectRatio >= 0.95 && aspectRatio <= 1.05))

{

CvInvoke.DrawContours(inputImage, contours, i, new MCvScalar(0, 0, 255), 2);

points[j] = new Point(x, y);

j++;

}

if (perimeter > 104 && perimeter < 110)

{

center = inputImage.Bitmap.GetPixel(x, y);

if (center.R > 240 && center.G > 240 && center.B > 240 && center.A == 255)

{

CvInvoke.DrawContours(inputImage, contours, i, new MCvScalar(255, 0, 0), 2);

//listBox1.Items.Add(x + " " + y + " Blue");

}

else

{

CvInvoke.DrawContours(inputImage, contours, i, new MCvScalar(0, 255, 0), 2);

//listBox1.Items.Add(x + " " + y + " Green");

//Вопрос 20

if ((x >= 505 && x <= 530) && (y >= 585 && y <= 600))

queStud[19, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 585 && y <= 600))

queStud[19, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 585 && y <= 600))

queStud[19, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 585 && y <= 600))

queStud[19, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 585 && y <= 600))

queStud[19, 0] = 1;

//вопрос 10

if ((x >= 250 && x <= 275) && (y >= 585 && y <= 600))

queStud[9, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 585 && y <= 600))

queStud[9, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 585 && y <= 600))

queStud[9, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 585 && y <= 600))

queStud[9, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 585 && y <= 600))

queStud[9, 0] = 1;

//Вопрос 19

if ((x >= 505 && x <= 530) && (y >= 535 && y <= 560))

queStud[18, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 535 && y <= 560))

queStud[18, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 535 && y <= 560))

queStud[18, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 535 && y <= 560))

queStud[18, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 535 && y <= 560))

queStud[18, 0] = 1;

//вопрос 9

if ((x >= 250 && x <= 275) && (y >= 535 && y <= 560))

queStud[8, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 535 && y <= 560))

queStud[8, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 535 && y <= 560))

queStud[8, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 535 && y <= 560))

queStud[8, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 535 && y <= 560))

queStud[8, 0] = 1;

//Вопрос 18

if ((x >= 505 && x <= 530) && (y >= 485 && y <= 510))

queStud[17, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 485 && y <= 510))

queStud[17, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 485 && y <= 510))

queStud[17, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 485 && y <= 510))

queStud[17, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 485 && y <= 510))

queStud[17, 0] = 1;

//вопрос 8

if ((x >= 250 && x <= 275) && (y >= 485 && y <= 510))

queStud[7, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 485 && y <= 510))

queStud[7, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 485 && y <= 510))

queStud[7, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 485 && y <= 510))

queStud[7, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 485 && y <= 510))

queStud[7, 0] = 1;

//Вопрос 17

if ((x >= 505 && x <= 530) && (y >= 435 && y <= 460))

queStud[16, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 435 && y <= 460))

queStud[16, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 435 && y <= 460))

queStud[16, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 435 && y <= 460))

queStud[16, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 435 && y <= 460))

queStud[16, 0] = 1;

//вопрос 7

if ((x >= 250 && x <= 275) && (y >= 435 && y <= 460))

queStud[6, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 435 && y <= 460))

queStud[6, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 435 && y <= 460))

queStud[6, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 435 && y <= 460))

queStud[6, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 435 && y <= 460))

queStud[6, 0] = 1;

//Вопрос 16

if ((x >= 505 && x <= 530) && (y >= 380 && y <= 415))

queStud[15, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 380 && y <= 415))

queStud[15, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 380 && y <= 415))

queStud[15, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 380 && y <= 415))

queStud[15, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 380 && y <= 415))

queStud[15, 0] = 1;

//вопрос 6

if ((x >= 250 && x <= 275) && (y >= 380 && y <= 415))

queStud[5, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 380 && y <= 415))

queStud[5, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 380 && y <= 415))

queStud[5, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 380 && y <= 415))

queStud[5, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 380 && y <= 415))

queStud[5, 0] = 1;

//Вопрос 15

if ((x >= 505 && x <= 530) && (y >= 340 && y <= 365))

queStud[14, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 340 && y <= 365))

queStud[14, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 340 && y <= 365))

queStud[14, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 340 && y <= 365))

queStud[14, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 340 && y <= 365))

queStud[14, 0] = 1;

//вопрос 5

if ((x >= 250 && x <= 275) && (y >= 340 && y <= 365))

queStud[4, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 340 && y <= 365))

queStud[4, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 340 && y <= 365))

queStud[4, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 340 && y <= 365))

queStud[4, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 340 && y <= 365))

queStud[4, 0] = 1;

//Вопрос 14

if ((x >= 505 && x <= 530) && (y >= 295 && y <= 320))

queStud[13, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 295 && y <= 320))

queStud[13, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 295 && y <= 320))

queStud[13, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 295 && y <= 320))

queStud[13, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 295 && y <= 320))

queStud[13, 0] = 1;

//вопрос 4

if ((x >= 250 && x <= 275) && (y >= 295 && y <= 320))

queStud[3, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 295 && y <= 320))

queStud[3, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 295 && y <= 320))

queStud[3, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 295 && y <= 320))

queStud[3, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 295 && y <= 320))

queStud[3, 0] = 1;

//Вопрос 13

if ((x >= 505 && x <= 530) && (y >= 250 && y <= 275))

queStud[12, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 250 && y <= 275))

queStud[12, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 250 && y <= 275))

queStud[12, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 250 && y <= 275))

queStud[12, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 250 && y <= 275))

queStud[12, 0] = 1;

//вопрос 3

if ((x >= 250 && x <= 275) && (y >= 250 && y <= 275))

queStud[2, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 250 && y <= 275))

queStud[2, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 250 && y <= 275))

queStud[2, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 250 && y <= 275))

queStud[2, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 250 && y <= 275))

queStud[2, 0] = 1;

//Вопрос 12

if ((x >= 505 && x <= 530) && (y >= 200 && y <= 225))

queStud[11, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 200 && y <= 225))

queStud[11, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 200 && y <= 225))

queStud[11, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 200 && y <= 225))

queStud[11, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 200 && y <= 225))

queStud[11, 0] = 1;

//вопрос 2

if ((x >= 250 && x <= 275) && (y >= 200 && y <= 225))

queStud[1, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 200 && y <= 225))

queStud[1, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 200 && y <= 225))

queStud[1, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 200 && y <= 225))

queStud[1, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 200 && y <= 225))

queStud[1, 0] = 1;

//Вопрос 11

if ((x >= 505 && x <= 530) && (y >= 150 && y <= 175))

queStud[10, 4] = 1;

if ((x >= 470 && x <= 495) && (y >= 150 && y <= 175))

queStud[10, 3] = 1;

if ((x >= 435 && x <= 460) && (y >= 150 && y <= 175))

queStud[10, 2] = 1;

if ((x >= 400 && x <= 425) && (y >= 150 && y <= 175))

queStud[10, 1] = 1;

if ((x >= 365 && x <= 390) && (y >= 150 && y <= 175))

queStud[10, 0] = 1;

//вопрос 1

if ((x >= 250 && x <= 275) && (y >= 150 && y <= 175))

queStud[0, 4] = 1;

if ((x >= 215 && x <= 240) && (y >= 150 && y <= 175))

queStud[0, 3] = 1;

if ((x >= 180 && x <= 205) && (y >= 150 && y <= 175))

queStud[0, 2] = 1;

if ((x >= 145 && x <= 170) && (y >= 150 && y <= 175))

queStud[0, 1] = 1;

if ((x >= 110 && x <= 135) && (y >= 150 && y <= 175))

queStud[0, 0] = 1;

}

}

pictureBox2.Image = inputImage.Bitmap;

}

Graphics g = Graphics.FromImage(pictureBox2.Image);

g.DrawLine(new Pen(Brushes.Violet, 3), points[0].X, points[0].Y, points[1].X, points[1].Y);

g.DrawLine(new Pen(Brushes.Violet, 3), points[1].X, points[1].Y, points[3].X, points[3].Y);

g.DrawLine(new Pen(Brushes.Violet, 3), points[3].X, points[3].Y, points[5].X, points[5].Y);

g.DrawLine(new Pen(Brushes.Violet, 3), points[5].X, points[5].Y, points[4].X, points[4].Y);

g.DrawLine(new Pen(Brushes.Violet, 3), points[4].X, points[4].Y, points[2].X, points[2].Y);

g.DrawLine(new Pen(Brushes.Violet, 3), points[2].X, points[2].Y, points[0].X, points[0].Y);

listBox1.Visible = true;

button1.Enabled = true;

}

private void инструментыToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (Program.res == true && inputImage != null)

найтиФигурыToolStripMenuItem.Enabled = true;

}

private void button1\_Click(object sender, EventArgs e)

{

for (int i = 0; i < 20; i++)

for (int k = 0; k < 5; k++)

if (queStud[i, k] == Program.quePrepod[i, k] && queStud[i, k] == 1)

sum++;

for (int i = 0; i < 20; i++)

{

listBox1.Items.Add("Вопрос " + (i + 1));

listBox1.Items.Add(Program.quePrepod[i, 0] + " " + Program.quePrepod[i, 1] + " " + Program.quePrepod[i, 2] + " " + Program.quePrepod[i, 3] + " " + Program.quePrepod[i, 4]);

listBox1.Items.Add("\n");

}

MessageBox.Show("Правильных ответов " + sum);

}

}

}

Программный код Мобильного приложения MainPage.xaml.cs:

using Plugin.Media;

using Plugin.Media.Abstractions;

using System;

using Xamarin.Forms;

namespace CameraApp

{

public partial class MainPage : ContentPage

{

public MainPage()

{

//InitializeComponent();

Button takePhotoBtn = new Button { Text = "Сделать фото" };

Button getPhotoBtn = new Button { Text = "Выбрать фото" };

Image img = new Image();

// выбор фото

getPhotoBtn.Clicked += async (o, e) =>

{

if (CrossMedia.Current.IsPickPhotoSupported)

{

MediaFile photo = await CrossMedia.Current.PickPhotoAsync();

img.Source = ImageSource.FromFile(photo.Path);

}

};

// съемка фото

takePhotoBtn.Clicked += async (o, e) =>

{

if (CrossMedia.Current.IsCameraAvailable && CrossMedia.Current.IsTakePhotoSupported)

{

MediaFile file = await CrossMedia.Current.TakePhotoAsync(new StoreCameraMediaOptions

{

SaveToAlbum = true,

Directory = "Sample",

Name = $"{DateTime.Now.ToString("dd.MM.yyyy\_hh.mm.ss")}.jpg"

});

if (file == null)

return;

img.Source = ImageSource.FromFile(file.Path);

}

};

Content = new StackLayout

{

HorizontalOptions = LayoutOptions.Center,

Children = {

new StackLayout

{

Children = {takePhotoBtn, getPhotoBtn},

Orientation =StackOrientation.Horizontal,

HorizontalOptions = LayoutOptions.CenterAndExpand

},

img

}

};

}

}

}