МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА ИРКУТСКА   
СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА № 34

**ПРОЕКТНАЯ РАБОТА**

**Разработка программ для создания и сортировки резюме**

Выполнили:

Байков Артем, 11А

Лукин Иван, 11А

Руководитель:

Преображенская Людмила

Юрьевна, учитель информатики

Иркутск, 2020

# Содержание

[Введение 3](#_Toc58418831)

[Основная часть 4](#_Toc58418832)

1. [Литературный обзор 4](#_Toc58418833)

2. [Методика 5](#_Toc58418834)

3. [Результаты работы 18](#_Toc58418835)

[Выводы 19](#_Toc58418836)

[Заключение 19](#_Toc58418837)

[Литературные источники 20](#_Toc58418838)

# Введение

В данной проектной работе мы решили разобраться с насущной **проблемой**, которая стоит перед нами - определение пути дальнейшего развития в жизни и как следствие выбор высшего учебного заведения и специальности, на которую мы собираемся поступать после окончания школы. **Мы считаем, что данная проблема актуальна** не только для нас, но и для многих других выпускников как 9, так и 11 классов, так как цена неверного выбора профессии может быть очень высока.

Мы, как люди, не понаслышке знакомые с компьютерами и их внутренним устройством, всерьез рассматриваем профессию программиста в качестве одного из вариантов. Но, к сожалению, мы даже представить себе не можем каково это - быть настоящим программистом уровня Middle или даже Senior. Поэтому мы решили разобраться с данной проблемой, попробовав себя в создании настоящих программ с графическим интерфейсом для компьютера под управлением операционной системы Windows. **Мы поставили для себя цель** создать две программы: одну программу для компоновки резюме (часть проектной работы Ивана Лукина), другую - для их удобной сортировки и фильтрации (часть проектной работы Артема Байкова). **Они же и являются предполагаемым продуктом данной проектной работы.**

**Для достижения поставленной в рамках данной проектной работы цели мы выделили следующие задачи:**

1. Выбрать язык программирования. В наше время их существует огромное множество, при этом каждый из них имеет разную сложность изучения и может использоваться для выполнения широкого круга определенных задач, поэтому важно правильно выбрать какой из них больше всего подходит для выполнения конкретно нашей.
2. Ознакомиться с литературой по программированию на выбранном языке, чтобы ознакомиться с его синтаксисом, типами данных, функциями, возможностями и т. д.
3. Спроектировать графический интерфейс программы
4. Определить алгоритм работы будущей программы
5. Реализовать алгоритм на выбранном языке программирования
6. Определить подходящий метод тестирования программного обеспечения
7. Протестировать программы выбранным методом
8. По результатам теста исправить выявленные ошибки

# Основная часть

## Обзор литературы

В наше время современных технологий, когда отовсюду слышится, что скоро многие профессии исчезнут и что нас заменят роботы, [8] действительно может быть довольно трудно определиться с тем, какой профессии стоит посвятить свою жизнь. Согласно опросу фонда развития культуры и кинематографии «Страна» у большинства (35%) подростков нет конкретной мечты [9], а редактор сайта “Российская Газета” Марина Гусенко утверждает, что “более 70% школьников “неосознанно некомпетентны”, то есть не имеют ни малейшего представления о том, какую профессию хотели бы получить и чем хотели бы заниматься в жизни” [10]. Подтверждает их слова о том, что современные школьники не знают, кем хотят стать и исследование национального исследовательского института «Высшая школа экономики» под названием «Траектории в образовании и профессии», стартовавшее еще в 2012 году [11]. Причины такого положения дел могут быть разные, например недостаточная информированность о разнообразном мире профессий [12]. или подавление родителями тяготений ребенка к определенным занятиям, когда ребенок четко знает, что “зажигает и вдохновляет” его, но родители не разрешают ему посвятить этому занятию свою жизнь, например из-за сложившихся в обществе стереотипов [8]. Мы также относимся к числу тех, кто не совсем уверен в том, какое направление развития им выбрать, поэтому мы очень заинтересованы в том, чтобы не упустить возможность почувствовать себя настоящим программистом, выполнив данную проектную работу, и понять, нравится ли нам эта профессия или нет.

## Методика

Для выполнения проектной работы мы рассматривали множество различных языков [18] (таб. 1)

**Таблица 1 - Сравнение различных языков программирования**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Язык программирования  (графический фреймворк) | Скорость | Читабельность | Простота | Кросс-  платформенность | Рейтинг в индексе TIOBE на 2020 г. [6] |
| C++ (Qt) | 8 | 6 | 3 | 7 | 4 (~7%) |
| Python (PyQt5) | 2 | 9 | 10 | 10 | 3 (~11,3%) |
| C# (WPF/WinForms) | 7 | 7 | 5 | 5 | 5 (~4%) |
| Java (JavaFx/Swing) | 6 | 7 | 6 | 10 | 2 (~12,6%) |
| JavaScript (Electron) | 6 | 5 | 6 | 5 | 7 (~2,14%) |

В итоге для выполнения данной проектной работы мы выбрали одну из последних версий высокоуровневого объектно-ориентированного языка программирования **Python**, созданного голландским программистом Гвидо ван Россумом в 1991 году [7]. Согласно индексу TIOBE, оценивающему популярность языков программирования, на октябрь 2020 года **Python занимает 3 место с рейтингом в ~11.3%**, уверенно обходя ближайшего конкурента в лице C++ с рейтингом в ~7% [6]. У профессионалов сферы программирования такое положение языка в рейтинге популярности не вызывает никакого удивления: хоть Python и является довольно медленным языком по сравнению с его собратьями, но в условиях изобилия вычислительных мощностей компьютеров в наше время этот его минус является незначительным, но при этом взамен **он предлагает очень простой и понятный синтаксис, который принуждает программиста писать красивый и читаемый код, множество различных типов данных с которыми очень удобно работать, богатую стандартную библиотеку, динамическую типизацию, кроссплатформенность и множество других возможностей, которые позволяют разработчику быстро писать качественный код**. Мы же выбрали Python просто потому, что мы уже в некоторой степени знакомы с ним, следовательно нам не придется с нуля полностью изучать новый, непонятный для нас язык программирования.

Для реализации графического интерфейса был использован набор расширений ("биндингов”) **графического фреймворка Qt**, изначально используемого для разработки программного обеспечения на языке программирования C++, для языка программирования Python под названием **PyQt5**. Qt - “взрослый” фреймворк, использующийся многими крупными компаниями в таких приложениях как **Autodesk Maya** (редактор трёхмерной графики), **Skype** (приложение для совершения онлайн-видеозвонков), **Telegram** (мессенджер), **VLC** (медиапроигрыватель), **Zoom** (программа для организации видеоконференций), **Wireshark** (программа для анализа трафика компьютерных сетей) и т. д [13].

После того, как мы определились с основой нашего проекта мы начали подробно изучать различную литературу по Python и PyQt5 [1, 2, 3]. Мы знали Python и до начала работы над данным проектом, но лишь на базовом уровне, явно недостаточном для создания полноценного настольного приложения. В ходе изучения мы познакомились с различными типами данных, циклами, конструкциями, многими функциями стандартной библиотеки языка и т. д. После ознакомления с ней мы начали увереннее чувствовать при работе с языком.

После того, как мы стали хорошо разбираться в “скелете” нашего проекта мы принялись за проектирование графического интерфейса пользователя (англ. graphical user interface, GUI). Для этого PyQt5 комплектуется визуальной средой разработки графического интерфейса **Qt Designer**, позволяющей создавать диалоги и формы в режиме **WYSIWYG** (“**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et”), что означает, что в процессе редактирования в окне программы мы видим интерфейс максимально близко похожим на то, чем он будет являться в итоге [14]. Согласно википедии, общедоступной свободной интернет-энциклопедии, резюме — это документ, содержащий информацию о навыках, опыте работы, образовании, и другую относящуюся к делу информацию, обычно требуемую для рассмотрения кандидатуры человека для найма на работу [15]. Мы учли это при проектировании интерфейса (рис. 1), и для компоновки резюме решили использовать следующую информацию о конечном пользователе продукта:

* Фамилию, Имя и Отчество
* Пол, возраст
* Контактный телефон и адрес электронной почты
* Уровень образования
* Желаемую должность
* Профессиональные навыки
* Личные качества
* Опыт работы в других компаниях

Для каждого из перечисленных пунктов мы добавили соответствующее текстовое поле или выпадающее меню. В нижней части программы мы также добавили кнопки, позволяющие приложить личное фото и сохранить резюме. В каждое из текстовых полей мы добавили пример его заполнения. В заголовке программы мы добавили иконку в виде листа с текстом с загнутым верхним правым уголком, а также название программы. Программу решено было назвать **“Resumaker” (от фр. résumé «сводка» и англ. make - создавать, делать)**.

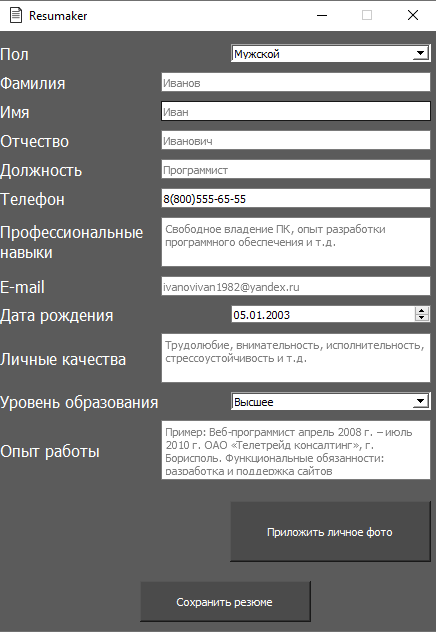


рис. 1

Программа **“ResuSearch” (от фр. résumé «сводка» и англ. search - искать)** будет являться дополнением к программе Resumaker. С помощью нее конечный пользователь, например, менеджер по персоналу (в наше время для этой профессии все чаще используется термин HR менеджер) сможет просматривать все резюме, сделанные в Resumaker, отсортировать и отфильтровать их. В качестве иконки программы была выбрано изображение с листом текста и показывающей на него лупой. В интерфейс программы мы добавили несколько ключевых элементов:

* Основное окно программы, выполненное в виде таблицы, подобно таблице в программе Microsoft Excel. В нем будет отображаться общая информация о каждом из кандидатов на работу. Также при нажатии на заголовок любого из столбцов программа автоматически отсортирует всех кандидатов по выбранному критерию.
* Выпадающее меню “Выделить по профессии, выбрав одну из профессий в котором пользователю будут показаны только кандидаты на определенную должность.
* Кнопки “Просмотреть/Редактировать резюме” и “Удалить резюме” которые позволяют открыть резюме в программе Microsoft Word чтобы ознакомиться с ним более подробно, либо удалить резюме из списка если оно не подходит по какому-либо из критериев соответственно (сам файл резюме все еще останется на диске)
* Кнопка “Экспортировать в таблицу Microsoft Excel”, которая позволяет создать на основе всех данных, показанных в программе таблицу в стандартном формате Microsoft Excel (XLSX), чтобы впоследствии совершать с этими данными различные офисные операции, недоступные в рамках функционала программы ResuSearch (строить графики процента людей определенного возраста или уровня образования на основе данных таблицы, вставлять таблицу в документ Microsoft Word, распечатать таблицу и т. д.)

Итоговый вариант спроектированного окна программы выглядит следующим образом (рис. 2):

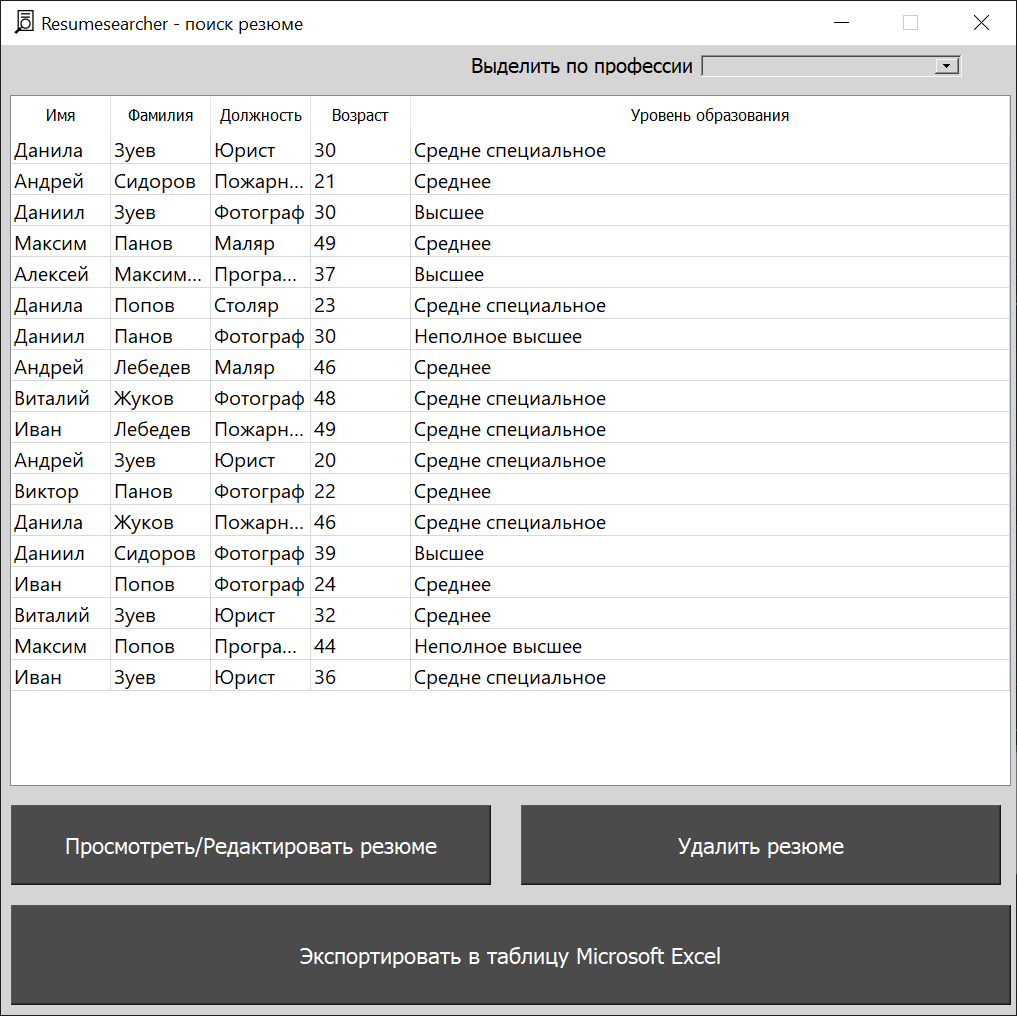


рис. 2

Далее перед нами стояла задача определения алгоритма работы будущей программы. На основе совокупности алгоритма и графического интерфейса впоследствии и будет написан исходный код программы, который позволит запускать ее на компьютере. Алгоритм создавался в векторном графическом редакторе Microsoft Visio с учетом спроектированного графического интерфейса и всех функций, которые мы хотели бы реализовать в программе. Для программы Resumaker это, непосредственно, создание резюме, а также создание базы данных (рис. 3)

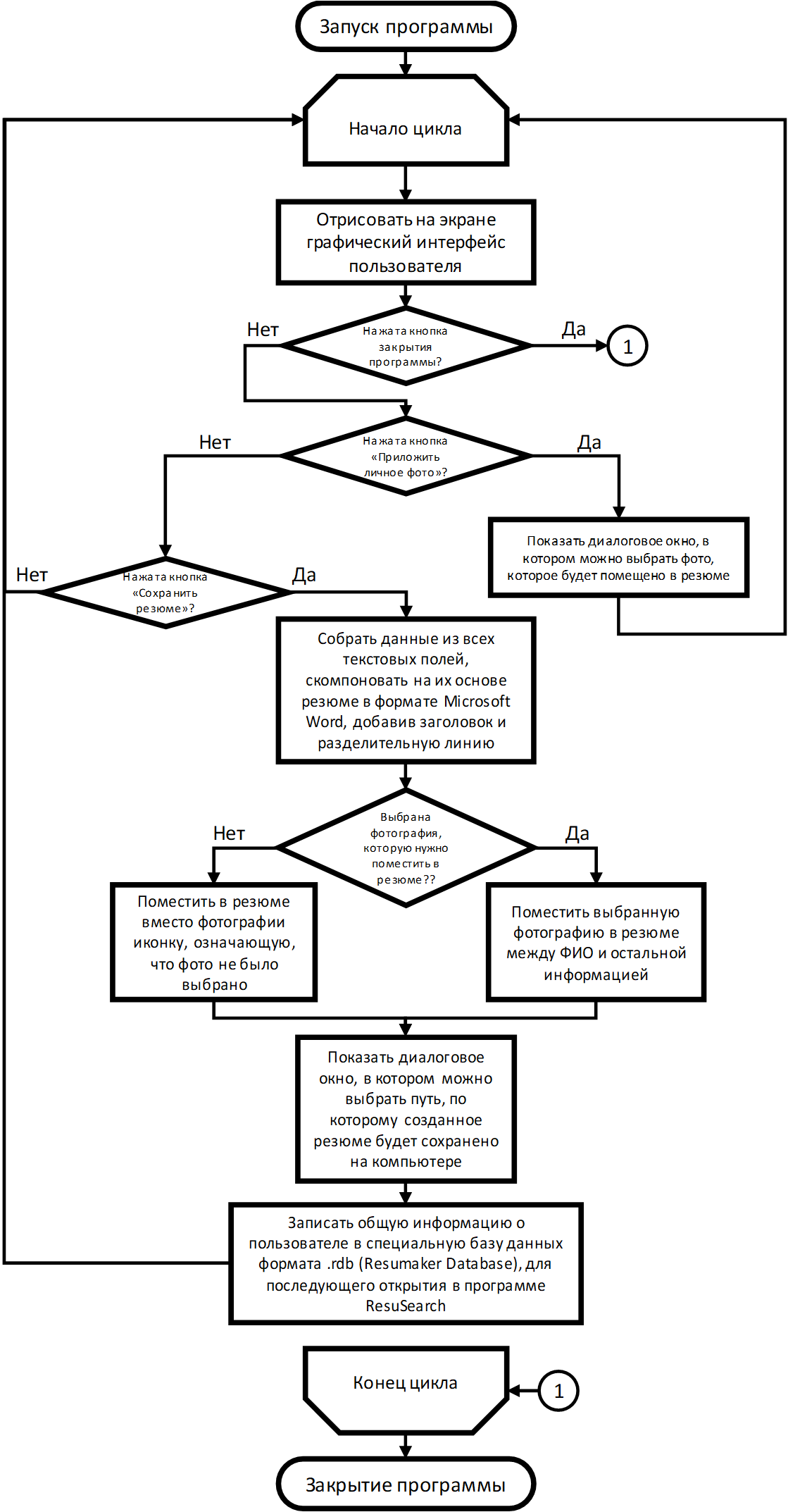


рис. 3

Для программы ResuSearch это фильтрация и сортировка резюме, возможность просмотреть и удалить файл резюме, а также возможность экспортировать базу данных Resumaker как стандартный файл Microsoft Excel (.xlsx) для дальнейшей работы с данными за пределами программы (рис. 4)

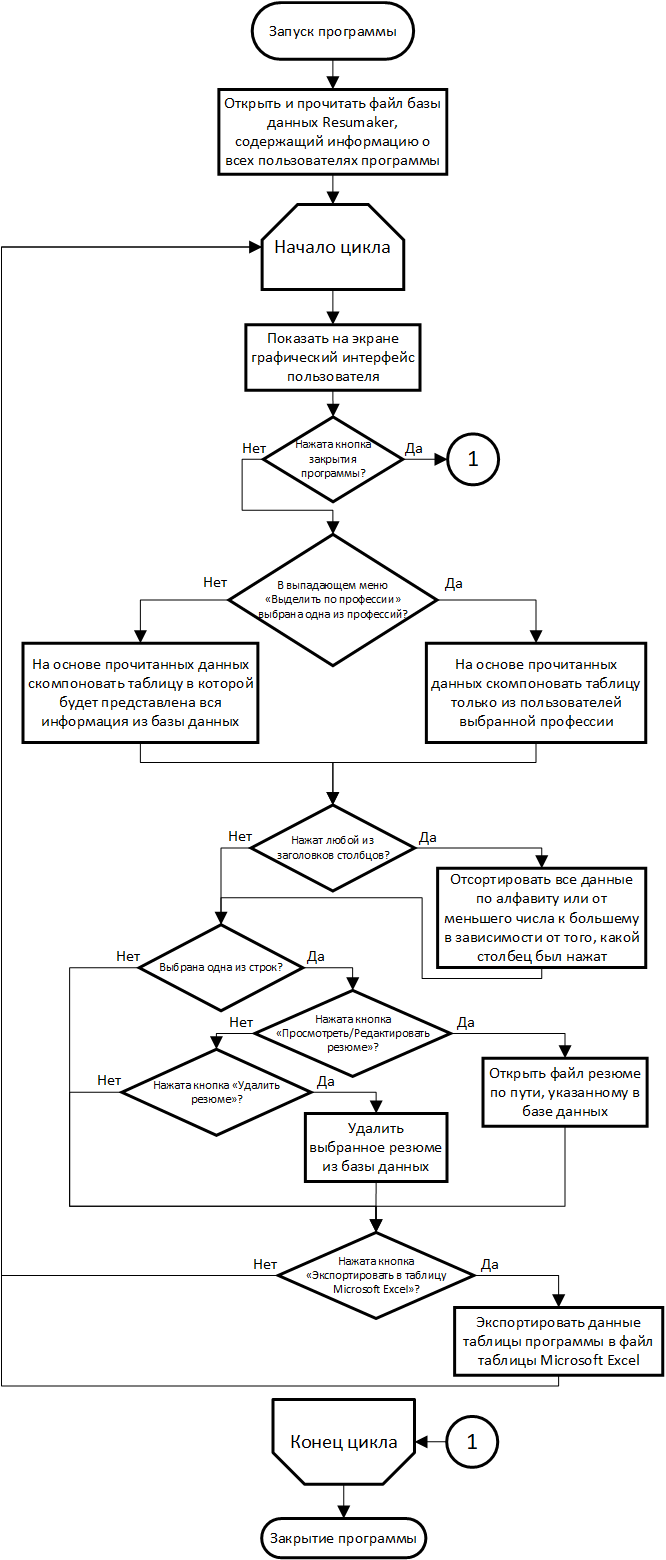


рис. 4

По увеличению количества функционала и требований к будущей программе можно заметить, как растет размер и сложность алгоритма: для программы Resumaker он немного меньше и намного проще, чем для программы ResuSearch.

Теперь, когда мы четко знаем и понимаем алгоритм работы каждой из программ, наконец-то пришло время приступить к программированию, но прежде чем приступить к написанию кода программы необходимо обзавестись незаменимым инструментом для любого программиста - интегрированной средой разработки или ИСР (англ. IDE). Интегрированная среда разработки представляет собой специальный текстовый редактор, включающий в себя средства для удобного редактирования и отладки кода, его транслятор (или же, в случае с языком программирования Python - интерпретатор), интеграцию с системами управления версиями и т. д. [17] Для выполнения данного проекта мы выбрали бесплатную версию IDE для языка программирования Python под названием **PyCharm Community Edition версии 2019.2.1**, созданную компанией JetBrains на основе другой среды разработки для разработки на языках Java и JavaScript - IntelliJ IDEA. Данное программное обеспечение является мощным и функциональным редактором кода предлагающим автодополнение, авто-отступы, отладчик кода, функцию быстрого просмотра документации прямо в окне редактора или просмотра внешней документации в браузере, интеграцию с системами контроля версий и баг-трекерами (в частности интеграцию с GitHub), огромную, постоянно пополняемую коллекцию плагинов и многое другое [20]. Думаю, что можно смело заявить, что PyCharm на данный момент является одним из лучших редакторов кода для языка Python на данный момент.

При написании кода мы ориентировались на статью о создании графического приложения на Python и PyQt на сайте TProger.ru [4]. В данной проектной работе будет описан процесс написания исходного кода лишь одной программы, т. к. различия в написании этих двух программ вызваны только разным набором функционала, который нужно было реализовать и в целом незначительны.

Для начала работы нужно сконвертировать файл разметки ранее созданного графического интерфейса программы расширения “.ui” в расширение “.py”, чтобы его можно было подключить в коде программы и работать с ним в дальнейшем. Делается это с помощью консольной утилиты **pyuic5**. Для того, чтобы сгенерировать файл нужно передать утилите в качестве аргумента полный путь до файла разметки графического интерфейса, а также полный путь, по которому мы хотим сохранить итоговый файл кода графического интерфейса расширения “.py” с ключом “-o” (англ. output - вывод). Делается это следующим образом:

C:\Users\Chuck> pyuic5 C:\School\_Project\Resumaker\ui\resumaker\_gui.ui -o

Создадим файл resumaker\_main.py, в котором будет описана вся логика работы программы. Для начала работы с PyQt нужно подключить непосредственно сам модуль, а также только что сгенерированный файл кода графического интерфейса. Дальше нужно создать класс, указав в качестве аргументов наследуемые классы. В данном случае это класс QMainWindow в модуле PyQt (PyQt5.QtWidgets.QMainWindow) а также класс в сгенерированном файле кода графического интерфейса resumaker\_gui.UI\_MainWindow. Впоследствии внутри этого класса и будет описана вся логика работы программы, а пока что нам лишь нужно создать функцию “\_\_init\_\_”, которая будет вызываться при создании класса и вызвать внутри ее 2 функции: обеспечивающую доступ к переменным и методам внутри файла кода графического интерфейса, а также инициализирующую сам интерфейс.

После создания класса опишем функцию main, в которой создадим объект QApplication из модуля PyQt, создадим объект класса Resumaker, вызовем функцию внутри этого класса, которая покажет окно программы и запустим непосредственно само приложение, вызвав функцию внутри созданного объекта QApplication.

Чтобы выполнить эту функцию, напишем стандартную конструкцию if \_\_name\_\_ == ‘main’, которая будет проверять, напрямую ли запущен файл, или он был импортирован. Если файл был запущен напрямую, вызовем функцию main, которая запустит приложение.

В итоге файл resumaker\_main.py выглядит примерно следующим образом:

1. **from** **sys** **import** argv # модуль sys нужен для передачи argv в объект QApplication
2. **from** **PyQt5** **import** QtWidgets # подключим модуль PyQt
3. **import** **resumaker\_gui** # и файл кода графического интерфейса
4. **class** **Resumaker**(QtWidgets.QMainWindow, resumaker\_gui.UI\_MainWindow): # создание класса Resumaker
5. **def** **\_\_init\_\_**(self): # эта функция вызывается при создании класса
6. super().\_\_init\_\_() # получим доступ к переменным и
7. # методам внутри родительского класса UI\_MainWindow
8. self.setupUi(self) # инициализируем интерфейс
9. **def** **main**(): # эта функция будет запускать приложение
10. app = QtWidgets.QApplication(argv) # создаем объект QApplication
11. # передаем ему в качестве аргумента функцию argv
12. window = Resumaker() # создаем объект класса Resumaker
13. window.show() # вызываем функцию, показывающую окно программы
14. app.exec\_() # запускаем приложение
15. **if** \_\_name\_\_ == '\_\_main\_\_': # Если мы запускаем файл напрямую, а не импортируем
16. main() # вызываем функцию main(), запускающую приложение

Если мы запустим выполнение нашего кода, написав в консоли:

C:\Users\Chuck> python3 C:\School\_Project\Resumaker\resumaker\_main.py

мы можем видеть окно запустившейся программы. Пока что оно не реагирует ни на какие действия, кроме заполнения текстовых полей и выбора вариантов в выпадающем меню, потому что мы только создали класс, не описав логику работы самой программы, но начало созданию программы уже положено.

Теперь внутри класса “Resumaker” объявим функции “addphoto” и “browse\_folder”, которые будут вызываться при нажатии кнопок “Приложить личное фото” и “Сохранить резюме” соответственно. Внутри функции “\_\_init\_\_” вызовем принадлежащие объектам кнопок функции connect, передав аргументами только что созданные функции внутри класса “Resumaker”:

self.savebutton.clicked.connect(self.browse\_folder)

self.Addphoto.clicked.connect(self.addphoto)

Теперь мы можем контролировать поведение программы, так как при нажатии кнопок код внутри соответствующих им функций будет выполняться, а следовательно мы можем начать реализовывать логику работы программы.

При нажатии кнопки “Приложить личное фото” нам нужно всего лишь показать пользователю окно, в котором можно будет указать путь до фотографии, которую пользователь хочет вставить в резюме. Для этого внутри функции “addphoto" класса “Resumaker” нам нужно вставить функцию “PyQt5.QtWidgets.QFileDialog.getOpenFileName”, при вызове которой пользователю будет показано нужное окно. Результат выполнения функции запишем в переменную “photopath”. Таким образом, если переменная “photopath” будет пустой, это будет означать, что пользователь не выбрал фотографию, которую хочет вставить в резюме, следовательно, нужно приложить иконку, означающую, что фото не было выбрано. Если в переменной “photopath” будет указан путь до файла фотографии расширения “.jpg” или “.png” нужно вставить фотографию, расположенную по этому пути в резюме.

В итоге функция “addphoto” будет выглядеть следующим образом:

def addphoto(self):

photopath = QtWidgets.QFileDialog.getOpenFileName(self, "Приложить личное фото", getenv('USERPROFILE'), "Файлы изображений (\*.png \*.jpg )")

При нажатии кнопки “Сохранить резюме” нужно выполнить несколько операций:

* Показать пользователю окно, в котором можно будет выбрать путь, по которому итоговый файл резюме будет сохранен
* Если пользователь выбрал путь, по которому нужно сохранить файл:
  + Прочитать текст из всех текстовых полей и выпадающих меню
  + Создать файл резюме формата Microsoft Word (.docx)
  + Добавить в него заголовок “Резюме” и разделительную черту
  + Вставить ФИО, пол и дату рождения пользователя
  + Если фотография выбрана, вставить фотографию
  + Если фотография не выбрана, вставить иконку, означающую, что фотография не была выбрана
  + Добавить всю остальную информацию о пользователе в резюме: должность, контактные данные, уровень образования, персональные качества, профессиональные навыки и опыт работы в других компаниях
  + Сохранить резюме по пути, который указал пользователь
  + Сохранить общую информацию о пользователе в файл базы данных resumaker\_database.rdb для дальнейшего использования в программе ResuSearch

Для того, чтобы показать пользователю окно, в котором можно будет выбрать путь, по которому будет сохранен файл, внутри функции “browse\_folder” класса Resumaker нужно вставить чуть отличающуюся, но похожую на ту, что мы вставляли в “addphoto” функцию PyQt5.QtWidgets.QFileDialog.getSaveFileName. Результат выполнения функции также нужно записать в переменную, но на этот раз в переменную “directory”. Таким образом, если переменная directory будет содержать в себе какой-либо путь, это будет означать, что пользователь выбрал путь, значит, нужно скомпоновать резюме и сохранить его по выбранному пути.

Для работы с документами формата Microsoft Word для языка программирования Python существует модуль, который называется python-docx. [5] Установим его с помощью системы управления пакетами, поставляющейся с установщиком Python под названием pip (**p**ackage **i**nstaller for **p**ython). [22] Для этого в командной строке запустим утилиту, передав ей в качестве аргумента команду “install” и название пакета “python-docx”:

C:\Users\Chuck> pip install python-docx

Подключим сам модуль в коде программы с помощью функции import:

from docx import Document

from docx.shared import Inches

Теперь, создав объект класса Document принадлежащий модулю python-docx, и передав ему в качестве аргумента путь, по которому нужно создать файл, мы можем добавлять в документ любую информацию а также сохранять его, используя методы add\_heading (создать заголовок), add\_paragraph (создать новый абзац), add\_run (добавить текст), add\_picture (добавить изображение), save (сохранить файл).

База данных Resumaker будет являться простейшим текстовым файлом в формате CSV (англ. **C**omma-**S**eparated **V**alues - значения, разделенные запятыми), который является текстовым форматом, предназначенным для представления табличных данных [21]. Каждая строка таблицы в файле такого формата соответствует строке текста, которая, как следует из названия формата, содержит одно или несколько полей, разделенных запятыми. В нашем же случае каждая строка текста будет содержать информацию об одном пользователе, и в каждой из них будут последовательно идти имя, фамилия, профессия, возраст, уровень образования и путь до файла резюме, разделенные запятой. Например, база данных, состоящая из 7 пользователей, будет выглядеть следующим образом:

**resumaker\_database.rdb**

1. Данила,Зуев,Юрист,30,Средне специальное,C:\Users\Максим\Desktop\Резюме\Д. Зуев, 30.docx
2. Андрей,Сидоров,Пожарный,21,Среднее,C:\Users\Максим\Desktop\Резюме\А. Сидоров, 21.docx
3. Максим,Панов,Маляр,49,Среднее,C:\C:\Users\Максим\Desktop\Резюме\М. Панов, 49.docx
4. Алексей,Максимов,Программист,37,Высшее,C:\Users\Максим\Desktop\Резюме\А. Максимов, 37.docx
5. Данила,Попов,Столяр,23,Средне специальное,C:\Users\Максим\Desktop\Резюме\Д. Попов, 23.docx
6. Даниил,Панов,Фотограф,30,Неполное высшее,C:\Users\Максим\Desktop\Резюме\Д. Панов, 30.docx
7. Андрей,Лебедев,Маляр,46,Среднее,C:\Users\Максим\Desktop\Резюме\А. Лебедев, 46.docx

Для того, чтобы работать с текстовыми файлами в стандартной библиотеке Python есть функция “open”. Чтобы начать работу с файлами, нужно вызвать функцию, передав ей 3 аргумента: полный путь до файла, режим работы с файлом и кодировка [19]. Режимов работы с файлами бывает несколько:

|  |  |
| --- | --- |
| Режим | Обозначение |
| r | открытие на чтение (является значением по умолчанию). |
| w | открытие на запись, содержимое файла удаляется, если файла не существует, создается новый. |
| x | открытие на запись, если файла не существует, иначе исключение. |
| a | открытие на дозапись, информация добавляется в конец файла. |
| b | открытие в двоичном режиме. |
| t | открытие в текстовом режиме (является значением по умолчанию). |
| + | открытие на чтение и запись |

В данном случае при работе с файлом базы данных нам нужно сначала открыть его, чтобы понять, содержится ли в нем какая-то информация, а уже потом начать запись новой. Так как нам нужно одновременно и читать, и записывать файл следует выбрать режим работы с файлом “r+”, что означает, что мы хотим открыть файл на чтение с возможностью записи.

Указать кодировку файла нужно для того, чтобы программа могла правильно интерпретировать символы русского алфавита, т. к. в данный момент для представления текста на русском языке существует несколько кодировок. Вдаваться в подробности о том, какие кодировки текста существуют на данный момент не считаем нужным, так как за множество лет эволюции компьютеров их появилось очень много. Нужно лишь знать, что на данный момент самый популярный формат кодировки текста, который используется по умолчанию для кодировки русского текста - **UTF-8** (**U**nicode **T**ransformation **F**ormat, **8**-bit), поэтому последним аргументом мы будем передавать строковое значение “utf-8” [23].

Результатом выполнения функции “open” будет файловый объект, который нужно записать в любую переменную:

db = open('resumaker\_database.rdb', 'r+', encoding="utf-8")

Далее для чтения данных из файла, или записи данных в него нужно вызвать метод “read” или метод “write” соответственно.

Прежде чем записывать новую информацию в базу данных нужно проверить, содержится ли в ней какая-нибудь информация. Если в файле уже присутствуют какие-либо данные нужно просто записать новую информацию, начав с новой строки. Если в файле нет никакой информации, нужно не переводить строку, а начать ведение базы данных прямо с первой строки. Для этого напишем простую конструкцию if, внутри которой мы будем читать файл с помощью метода “read”. Если файл содержит любую информацию, сначала в конец файла запишем управляющий символ таблицы ASCII “\n”, который означает перевод строки [24]. Дальше запишем всю остальную информацию. Таким образом, если файл пустой, то новая информация будет записываться в файл с первой строки, а если нет, то сначала в файл запишется символ перевода строки, а уже потом новая информация. Данная конструкция будет выглядеть следующим образом:

if db.read() != "":

db.write("\n")

db.write(name + "," + second\_name + "," + post + "," +\

age + "," + edu\_level + "," + directory[0])

После окончания работы с файлом нужно вызвать метод “close”, который запишет все произведенные изменения в файл:

db.close()

Таким образом в ходе выполнения работы была полностью реализована логика работы программы Resumaker. Ниже представлен полный исходный код программы:

from sys import argv # модуль sys нужен для передачи argv в объект QApplication

from os import getenv # нужен для отображения окна выбора файла

from PyQt5 import QtWidgets # модуль PyQt

import resumaker\_ui # файл кода графического интерфейса

from docx import Document # модуль для работы с файлами Microsoft Word

from docx.shared import Inches

# создание класса Resumaker с наследованием классов QtWidgets.QMainWindow и resumaker\_ui.Ui\_MainWindow

class Resumaker(QtWidgets.QMainWindow, resumaker\_ui.Ui\_MainWindow):

# функция, которая вызывается при создании класса

def \_\_init\_\_(self):

super().\_\_init\_\_() # получим доступ к переменным и

# методам внутри родительского класса Ui\_MainWindow

self.setupUi(self)

# инициализируем интерфейс

self.savebutton.clicked.connect(self.browse\_folder)

self.Addphoto.clicked.connect(self.addphoto)

# привяжем нажатие кнопок к исполнению соответствующих им функций

# функция, которая вызывается при нажатии кнопки "Приложить личное фото"

def addphoto(self):

# покажем пользователю окно выбора файла фото и запишем путь до файла в переменную "photopath"

photopath = QtWidgets.QFileDialog.getOpenFileName(self, "Приложить личное фото", getenv('USERPROFILE'),

"Файлы изображений (\*.png \*.jpg )")

# функция, которая вызывается при нажатии кнопки "Сохранить резюме"

def browse\_folder(self):

# вызовем функцию, которая покажет пользователю окно, в котором можно будет

# выбрать путь, по которому будет сохранен файл резюме и запишем его в переменную "directory"

directory = QtWidgets.QFileDialog.getSaveFileName(self, "Сохранение резюме",

getenv('USERPROFILE') + r"\Desktop",

'Документ Word (\*.docx)')

# если переменная directory содержит путь:

if directory[0] != "":

# создать объект класса Document, чтобы начать работу с файлом

resume\_savefile = Document()

resume\_savefile.save(directory[0])

resume = Document(docx=directory[0])

# собрать информацию из всех текстовых полей

sex = self.sex\_comboBox.currentIndex()

sex\_text = self.sex\_comboBox.currentText()

second\_name = self.secondname.displayText()

name = self.name.displayText()

patronymic = self.patronymic.displayText()

post = self.post.displayText()

phone\_number = self.phoneNumber.displayText()

professional\_skills = self.professional\_skills.toPlainText()

email = self.email.displayText()

birth\_date = self.birthDate.date().toPyDate()

personal\_qualities = self.personal\_qualities.toPlainText()

edu\_level = self.edu\_level.currentText()

experience = self.experience.toPlainText()

birth\_year = int(str(birth\_date)[0:4])

birth\_month = str(birth\_date)[5:7]

birth\_day = str(birth\_date)[8:10]

birth\_date = birth\_day + "." + birth\_month + "." + str(birth\_year)

age = str(2019 - birth\_year)

if sex == 1:

sex = "Мужчина, "

else:

sex = "Женщина, "

# добавить заголовок "Резюме" и разделительную черту

resume.add\_heading('Резюме', level=0)

# добавить ФИО

resume.add\_heading(text=str(second\_name) + " " + str(name) + " " + str(patronymic), level=1)

# добавить пол и дату рождения

resume.add\_heading(sex + age + " лет, дата рождения: " + birth\_date, level=1)

# если выбрана фотография:

if photopath[0] != "":

# вставить ее в резюме

resume.add\_picture(photopath[0], width=Inches(1.25))

# если нет:

else:

# вставить иконку, означающую, что фотография не была выбрана

resume.add\_picture("dummy.png", width=Inches(1.25))

# добавить всю остальную информацию:

p = resume.add\_paragraph(text="", style=None)

p.add\_run('Должность: ').bold = True

p.add\_run(post)

if phone\_number != "":

mobile\_phone = resume.add\_paragraph(text="", style=None)

mobile\_phone.add\_run('Моб. телефон: ').bold = True

mobile\_phone.add\_run(phone\_number)

if email != "":

em = resume.add\_paragraph(text="", style=None)

em.add\_run('E-mail: ').bold = True

em.add\_run(email)

edu = resume.add\_paragraph(text="", style=None)

edu.add\_run('Уровень образования: ').bold = True

edu.add\_run(edu\_level)

if personal\_qualities != "":

personal\_qualities\_title = resume.add\_paragraph(text="", style=None)

personal\_qualities\_title.add\_run('Персональные качества: ').bold = True

resume.add\_paragraph(text=personal\_qualities, style=None)

if professional\_skills != "":

professional\_skills\_title = resume.add\_paragraph(text="", style=None)

professional\_skills\_title.add\_run('Профессиональные навыки: ').bold = True

resume.add\_paragraph(text=professional\_skills, style=None)

if experience != "":

experience\_title = resume.add\_paragraph(text="", style=None)

experience\_title.add\_run('Опыт работы:').bold = True

resume.add\_paragraph(text=experience, style=None)

# открыть файл базы данных

db = open('resumaker\_database.rdb', 'r+', encoding="utf-8")

# если файл бд содержит какую-либо информацию:

if db.read() != "":

# вставить символ перевода строки

db.write("\n")

# записать всю информацию в файл

db.write(name + "," + second\_name + "," + post + "," +\

age + "," + edu\_level + "," + directory[0])

# применить изменения в файле и закончить работу с ним

db.close()

# закончить компоновку резюме и заверишть работу с файлом

resume.save(directory[0])

def main(): # эта функция будет запускать приложение

app = QtWidgets.QApplication(argv) # создаем объект QApplication

# и передаем ему в качестве аргумента функцию argv

window = Resumaker() # создаем объект класса Resumaker

window.show() # вызываем функцию, показывающую окно программы

app.exec\_() # запускаем приложение

if \_\_name\_\_ == '\_\_main\_\_': # Если мы запускаем файл напрямую, а не импортируем

main() # вызываем функцию main(), запускающую приложение

Для тестирования программы был использован метод ручного системного тестирования белого ящика. Системное тестирование предполагает тестирование всего кода программы одновременно, тестирование “белого ящика” означает, что мы проводили тестирование, зная внутреннее устройство нашей программы [16]. В ходе тестирования никаких ошибок программного кода выявлено не было.

## Полученные результаты

В результате нашей работы мы создали 2 программы, которые могут использоваться либо по прямому назначению – для создания резюме с помощью **Resumaker (рис. 5)**

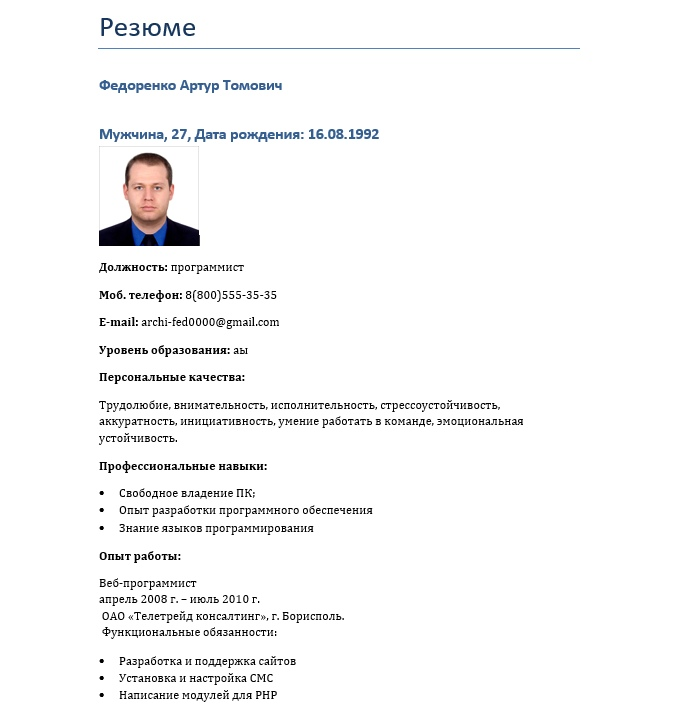


рис. 5

или сортировки созданных в **Resumaker** резюме с помощью **ResuSearch** (рис. 6)

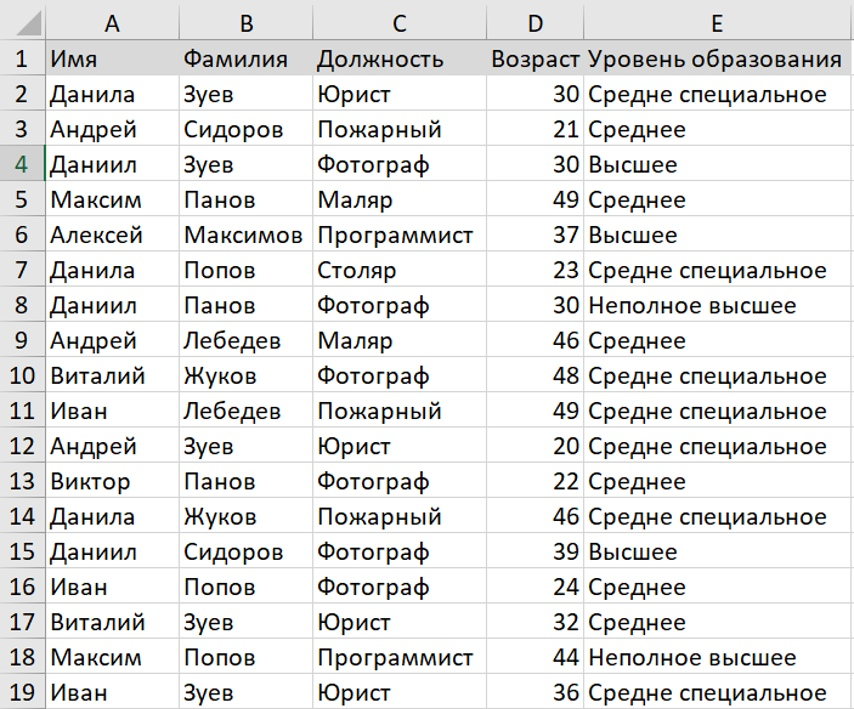


рис. 6

либо в качестве примеров программ с графическим интерфейсом для обучения других программированию на языке Python. **Но самый главный результат, который мы получили – это бесценный опыт**, без которого в наши дни практически ни один работодатель не берет к себе на работу, и, имея который теперь мы можем более уверенно судить о направлении нашего развития и выборе нашей дальнейшей профессии.

# Выводы

1. Мы остановились на ЯП Python, потому что этот язык один из самых популярных, востребованных и легких в освоении
2. Выбрав язык, мы приступили к изучению синтаксиса языка, функциям и возможностям
3. При помощи **Qt Designer** спроектировали интерфейс обеих программ
4. Разработали алгоритм программ
5. Реализовали алгоритм на языке Python
6. Для тестирования программы был использован метод ручного системного тестирования белого ящика.
7. Протестировали программы
8. В ходе тестирования не обнаружили ошибок, так что изменения в коде не потребовались

# Заключение

В ходе работы мы вошли в роль профессии программиста, и исходя из этого мы выяснили, что его работа часто вызывает фрустрацию, особенно в процессе поиска и устранения ошибок, но в целом программирование — это увлекательное занятие, и теперь, после выполнения данной проектной работы, мы уверены, что хотим развиваться в этом направлении дальше и, возможно, в ближайшем будущем превратить это небольшое хобби в занятие, которое будет кормить нас всю оставшуюся жизнь.

Исходный код обоих программ доступен в репозитории на сайте GitHub, так что любой желающий может просмотреть его или даже продолжить работу над данным проектом:

**Ссылка на репозиторий программы Resumaker**: <https://github.com/chuckythecat/Resumaker>

**Ссылка на репозиторий программы Resusearch:** <https://github.com/chuckythecat/Resusearch>

# Источники

1. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. - СПб.: Питер, 2017. - 496 с.
2. МакГрат М. Программирование на Python для начинающих. - Москва: Эксмо, 2015. - 192 с.
3. Прохоренок Н. Python 3 и PyQt 5. Разработка приложений / Н. Прохоренок, В. Дронов. - СПб.: БХВ-Петербург, 2017. - 832 с.
4. Статья о создании приложения с графическим интерфейсом на основе Python и PyQt на сайте издательства об актуальных новостях из мира IT [Электрон. ресурс] - URL: <https://tproger.ru/translations/python-gui-pyqt/>

(Дата обращения: 04.09.2020)

1. Документация на библиотеку для работы с документами формата Microsoft Word [Электрон. ресурс] - URL: <https://python-docx.readthedocs.io/en/latest/>

(Дата обращения: 07.09.2020)

1. Индекс TIOBE, оценивающий популярность языков программирования [Электрон. ресурс] - URL: <https://www.tiobe.com/tiobe-index/>

(Дата обращения: 11.09.2020)

1. Страница об языке программирования Python в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: <https://ru.wikipedia.org/wiki/Python>

(Дата обращения: 14.09.2020)

1. Статья о том, почему современные школьники не знают, кем быть на сайте «Беларусь сегодня» [Электрон. ресурс] – URL: <https://www.sb.by/articles/raskroy-v-sebe-talant.html> (Дата обращения: 21.09.2020)
2. Статья об профориентационном опросе школьников, проведенном фондом развития культуры и кинематографии «Страна» на сайте МИР24 [Электрон. ресурс] – URL: <https://mir24.tv/news/16412114/shkolniki-v-rossii-hotyat-stat-vrachami-rezhisserami-i-uchitelyami> (Дата обращения: 21.09.2020)
3. Статья о профориентации школьников на сайте «Российская Газета» [Электрон. ресурс] – URL: <https://rg.ru/2019/05/23/bolshinstvo-shkolnikov-ne-opredelilis-s-vyborom-professii.html> (Дата обращения: 21.09.2020)
4. Статья о результатах исследования «Траектории в образовании и профессии» на сайте национального исследовательского университета «Высшая школа экономики» [Электрон. ресурс] – URL: <https://ioe.hse.ru/news/300366606.html>
5. Статья «Типичные трудности и ошибки при выборе профессии у старшеклассников» на сайте научного журнала «Молодой Ученый» [Электрон. ресурс] – URL: <https://moluch.ru/archive/95/21337/>
6. Страница о программах, использующих фреймворк Qt в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: [https://ru.wikipedia.org/wiki/Категория:Приложения,\_использующие\_Qt](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F,_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D1%8E%D1%89%D0%B8%D0%B5_Qt) (Дата обращения: 22.09.2020)
7. Страница о программе Qt Designer в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: <https://ru.wikipedia.org/wiki/Qt_Designer> (Дата обращения: 26.09.2020)
8. Страница о документе “резюме” в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: [https://ru.wikipedia.org/wiki/Резюме](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D1%8E%D0%BC%D0%B5) (Дата обращения: 27.09.2020)
9. Страница о тестировании программного обеспечения в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: [https://ru.wikipedia.org/wiki/Тестирование\_программного\_обеспечения](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (Дата обращения: 01.10.2020)
10. Страница о интегрированных средах разработки в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: [https://ru.wikipedia.org/wiki/Интегрированная\_среда\_разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) (Дата обращения: 07.10.2020)
11. Сравнение различных языков программирования в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: [https://ru.wikipedia.org/wiki/Сравнение\_языков\_программирования](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) (Дата обращения: 15.10.2020)
12. Статья о работе с файлами на языке программирования Python на сайте самоучителя по Python 3 [Электрон. ресурс] - URL: <https://pythonworld.ru/tipy-dannyx-v-python/fajly-rabota-s-fajlami.html> (Дата обращения: 24.10.2020)
13. Страница о ИСР PyCharm в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: <https://ru.wikipedia.org/wiki/PyCharm> (Дата обращения: 27.10.2020)
14. Страница о текстовом формате CSV в онлайн-энциклопедии “Википедия” [Электрон. ресурс] - URL: <https://ru.wikipedia.org/wiki/CSV> (Дата обращения: 03.11.2020)
15. Страница о системе управления пакетами pip в онлайн-энциклопедии “Википедия” [Электрон. ресурс] URL: [https://ru.wikipedia.org/wiki/Pip\_(менеджер\_пакетов)](https://ru.wikipedia.org/wiki/Pip_(%D0%BC%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%B5%D1%80_%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D0%BE%D0%B2)) (Дата обращения: 04.11.2020)
16. Страница о стандарте кодирования символов UTF-8 в онлайн-энциклопедии “Википедия” [Электрон. ресурс] URL: <https://ru.wikipedia.org/wiki/UTF-8> (Дата обращения: 08.11.2020)
17. Страница о таблице для кодировки текста ASCII [Электрон. ресурс] URL: <https://ru.wikipedia.org/wiki/ASCII> (Дата обращения: 16.11.2020)