

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Московский институт электроники и математики им. А.Н. Тихонова**

**ЗАДАНИЕ**

**на выполнение междисциплинарной курсовой работы**

студенту группы БИВ186 Заякиной Анастасии Михайловне

1. Тема работы

|  |
| --- |
| Разработка системы автоматической генерации тестов для дисциплины "Информатика" |

1. Требования к работе

2.1 Общие требования

Система взаимодействия с учеником должна быть проработана во избежание ошибок в оценивании и непонимания со стороны пользователя.

2.2 Требования к процессу разработки и итоговой системе

- Процесс взаимодействия пользователя с системой тривиален или созданы соответствующие пояснения

- Система позволяет создавать задания различных типов и уровней сложности

- Формируются индивидуальные файлы заданий для ученика и преподавателя

3. Содержание работы

- Обзор аналогов

- Описание ПО

- Описание функционала

- Результаты тестирования ПО на целевой аудитории

4. Сроки выполнения

Первая версия ПО предоставлена до **28 февраля 2020 г**

Предварительная версия отчета предоставлена **до 5 апреля 2020 г**

Конечная версия ПО предоставлена до **15 мая 2020 г**

Итоговый отчет (оформленный по ГОСТу) предоставлен до **31 мая 2020 г**

|  |  |  |
| --- | --- | --- |
| Задание выдано студенту | «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / И. И. Романова |
| Задание принято к исполнению студентом | «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А. М. Заякина |

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Московский институт электроники и математики им. А.Н. Тихонова**

**График сдачи этапов**

**Междисциплинарной курсовой работы бакалавра**

студента *Заякиной Анастасии Михайловны*  группы БИВ186

Тема работы:

Разработка системы автоматической генерации тестов для дисциплины «Информатика»

|  |  |  |
| --- | --- | --- |
| Дата представления первого варианта МКР | «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_\_ г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.О. Фамилия *подпись руководителя* |
| Дата представления итогового варианта МКР | «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_\_ г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.О. Фамилия *подпись руководителя* |

Аннотация

**Цель курсовой работы**: Разработать систему автоматической генерации тестов по дисциплине «Информатика» и реализовать выдачу тестовых материалов в текстовом редакторе Word.

В дальнейшем планируется внедрение этой системы в учебный процесс и использование для генерации тестов учащимся по дисциплине «Информатика».

Передо мной были поставлены следующие задачи:

1. Обзор аналогов систем генерации тестов
2. Выбор программного обеспечения;
3. Разработка эргономичного интерфейса для пользователей
4. Программирование тестов по заданным разделам
5. Выдача тестовых материалов в документе Word
6. Выдача ответов к тестовым материалам

В процессе выполнения курсовой работы осуществлено более углублённое изучение языка Python, освоены графическая библиотека Tkinter для оконного интерфейса и модуль python-docx для работы с документами Word. Кроме того, получен опыт в создании User Friendly интерфейса, а также в группировке отдельных тем заданий в смысловые разделы.

Каждое из заданий прошло тестирование; ответы, предложенные системой, оказались правильными.

Разработанная система имеет следующие преимущества по сравнению с существующими аналогами:

* *Адресность* - ориентированность на задания, используемые именно в НИУ ВШЭ для изучения дисциплины «Информатика»;
* *Поддержка пользователя -* Cведение возможности ошибки пользователя к минимуму;
* *Дружественность* - максимальная простота и удобство интерфейса;
* *Многовариантность* - возможность получения практически неограниченного количества вариантов тестов, благодаря использованию функции выбора случайных чисел;
* *Редактируемость* - выдача вариантов для тестирования в удобной форме для правки;
* *Полнота* - выдача ответов ко всем полученным вариантам.

Объем данной работы составляет 27 страниц, включая 12 рисунков и 1 таблицу.

Annotation

The main goal of course work:

To develop a system for automatic generation of tests in the discipline "Informatics" and implement the issuance of test materials in a text editor Word.

Also there are plans to introduce this system into the educational process and use students to study computer science in order to generate tests.

The following tasks were set for me:

1. Overview of test generation system analogues

2. The choice of software

3. Ergonomic user interface development

4. Programming tests on specified sections

5. Issue of test materials in a Word document

6. Issuing answers to test materials

The course work process allows to learn more in-depth Python programming language, learn how to work with Tkinter graphic library for the window interface and the python-docx module for working with Word documents. In addition, experience in creating a User Friendly interface was gained, as well as in grouping individual task topics into semantic sections.

Each of the tasks was tested; The answers suggested by the system turned out to be correct.

The developed system has the following advantages compared to existing analogues:

* Targeting - focus on tasks used specifically at the Higher School of Economics for the study of the discipline "Computer Science";
* User support - Minimizing the possibility of user error;
* User friendly - the maximum simplicity and convenience of the interface;
* Multivariance - the ability to get an almost unlimited number of test options, thanks to the use of the random number selection function;
* Editable - the issuance of options for testing in a convenient form for editing;
* Completeness - issuing answers to all received options.

This work consists of 27 pages, including 12 pictures and 1 table.

Содержание

[Введение 9](#_Toc41842838)

[Анализ существующих технических решений 9](#_Toc41842839)

[Темы, включённые в систему автоматической генерации 12](#_Toc41842840)

[Анализ технических решений объекта разработки 13](#_Toc41842841)

[1. Выбор языка программирования для создания системы 13](#_Toc41842842)

[2. Используемые встроенные библиотеки и конкретные задачи, ими выполняемые 14](#_Toc41842843)

[3. Описание принципа работы программы 15](#_Toc41842844)

[Описание создания программного интерфейса 20](#_Toc41842845)

[Описание работы ключевых фрагментов программы 23](#_Toc41842846)

[Заключение 25](#_Toc41842847)

[Список источников 25](#_Toc41842848)

# Введение

Педагогический тест - это инструмент, предназначенный для измерения знаний учащегося, состоящий из системы тестовых заданий и стандартизованной процедуры проведения, обработки и анализа результатов.

Контроль знаний студентов является одной из проблем высшего образования, а тестирование позволяет дать предварительную оценку знаний обучающегося, как студенту (для самоконтроля), так и преподавателю.

Выполнение данной курсовой работы даст возможность ознакомиться с различными системами генерации тестов, позволит провести их сравнительный анализ, а также углубить знание языков программирования и получить опыт и навыки в разработке практически значимых систем.

Тема является актуальной, поскольку тестовый контроль знаний в настоящее время является в высшем образовании одной из самых распространенных форм оценки знаний обучаемых и, согласно нормативным документам в образовании и соответствующим рабочим планам изучаемых дисциплин, является обязательным компонентом освоения практически всех изучаемых дисциплин. Система автоматической генерации тестов будет не только хорошим помощником преподавателю, позволив снизить объем его рутинной работы в проверке знаний студентов, но и поможет студентам в изучении учебного материала.

# Анализ существующих технических решений

Рассмотрим различные аналоги генератора тестов.

1. **Knowing1(рис. 1)**

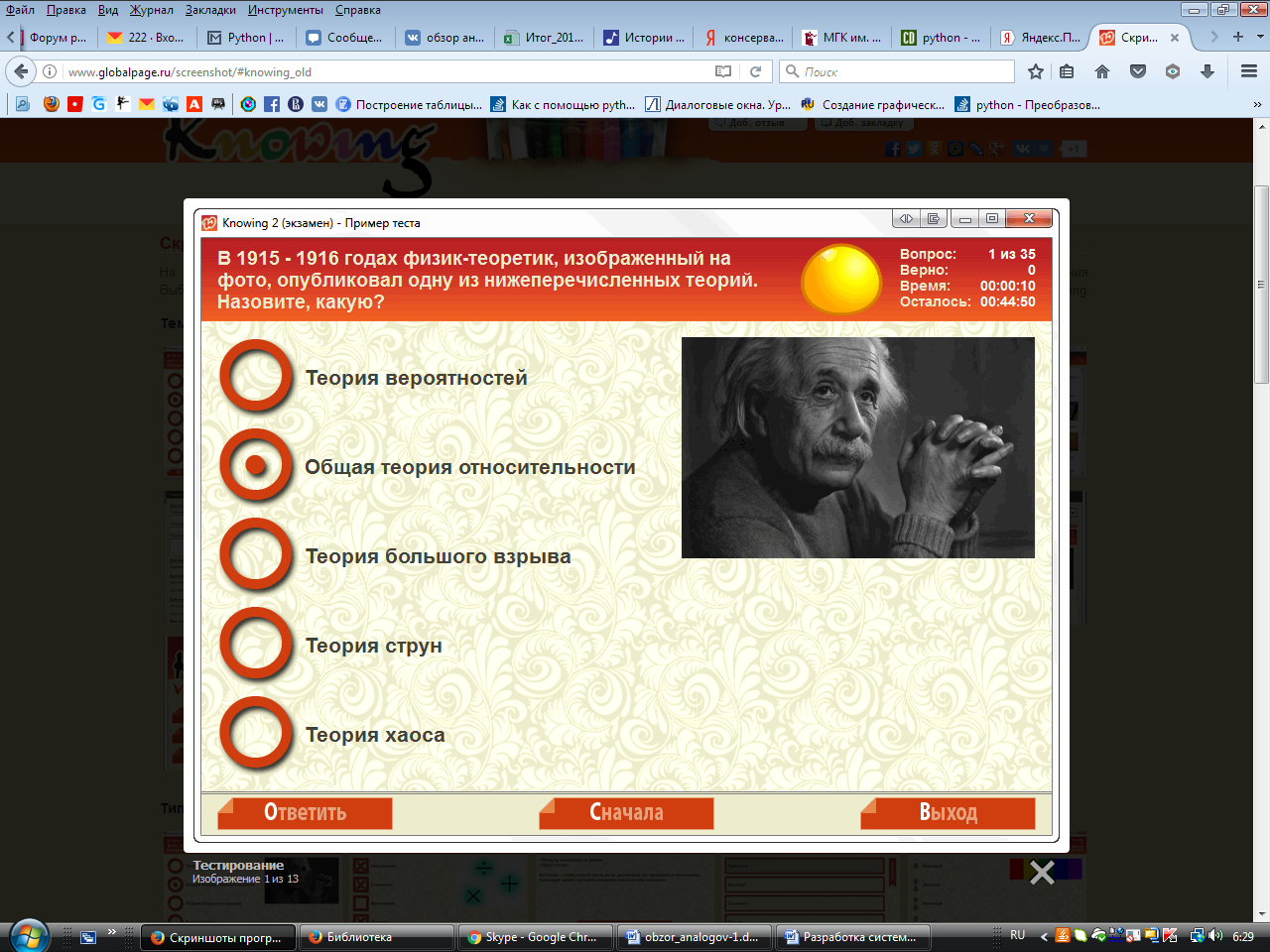
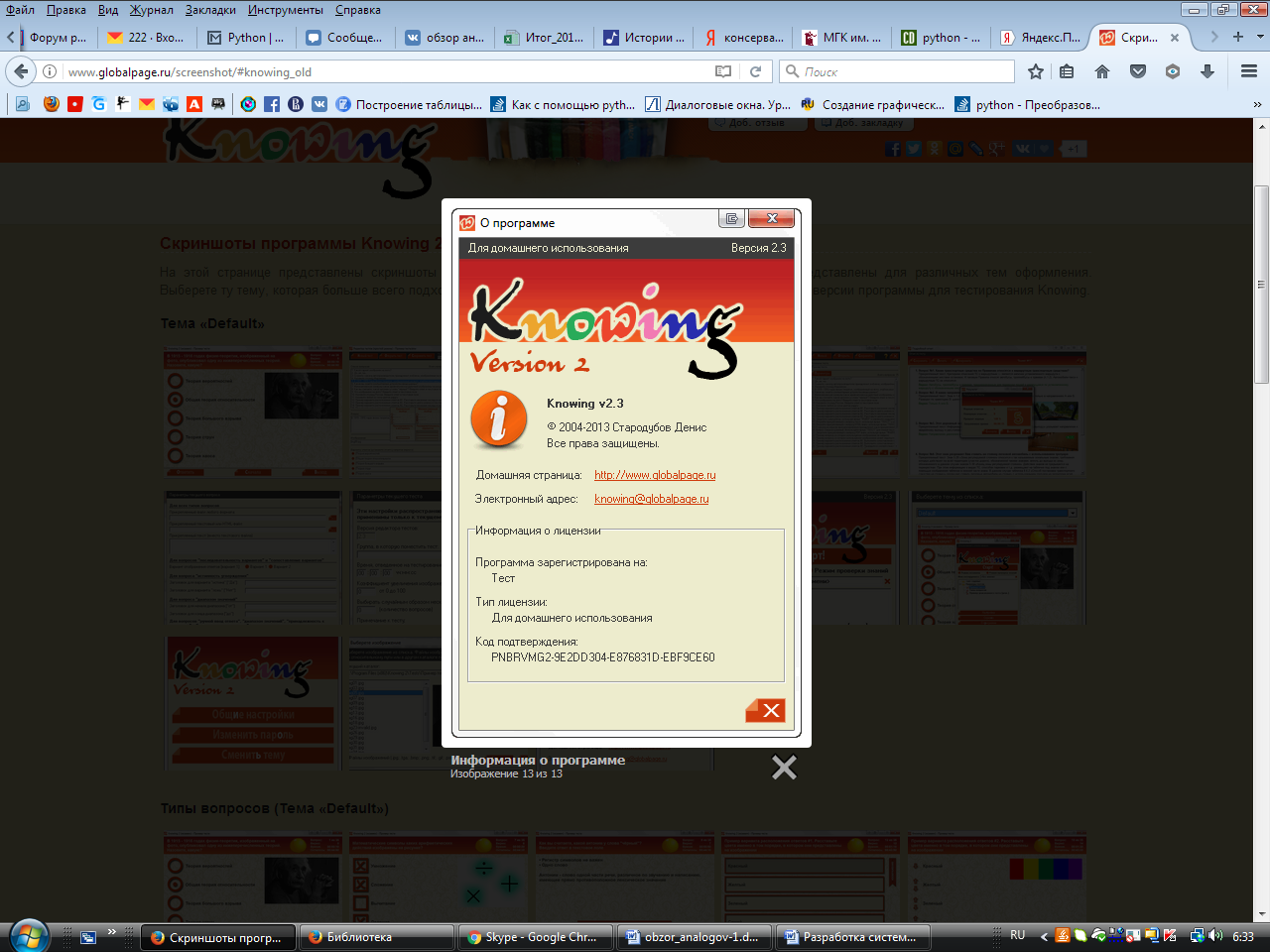


Рис. 1. - Внешний вид программы Knowing.

Knowing – программа для создания тестов. Данная программа может использоваться для создания тестов по любым темам. Выделим основные особенности:

1. Два режима создания тестов:

* Простой (вопросы добавляются поочередно, изменить или добавить пропущенный вопрос можно только по завершении работы)
* Текстовый (вопросы добавляются исходя из желаний создателя, прямо в процессе создания каждый из вопросов и его параметры доступны к редактированию)

2. Типы вопросов:

* Единичный выбор
* Множественный выбор
* Ручной ввод ответа
* Диапазон значений
* Принадлежность к интервалу
* Последовательность вариантов
* Сопоставление вариантов
* Истинность утверждения

3. Присутствует возможность добавления картинок или таблиц

4. Выставление оценки в заданном диапазоне после окончания

5. Защита тестов и их настроек от изменения паролем

1. **ADTester2 (рис. 2)**

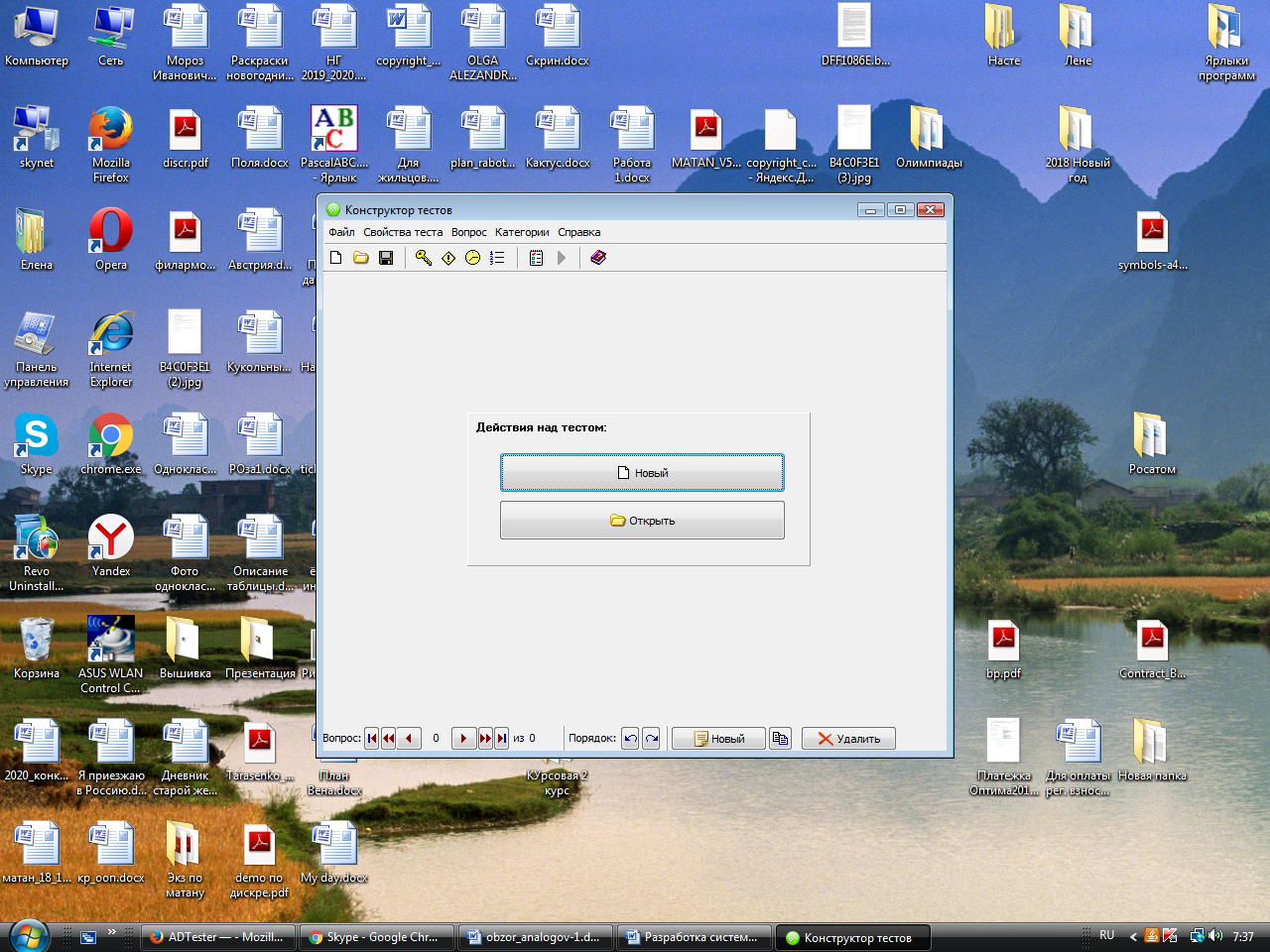
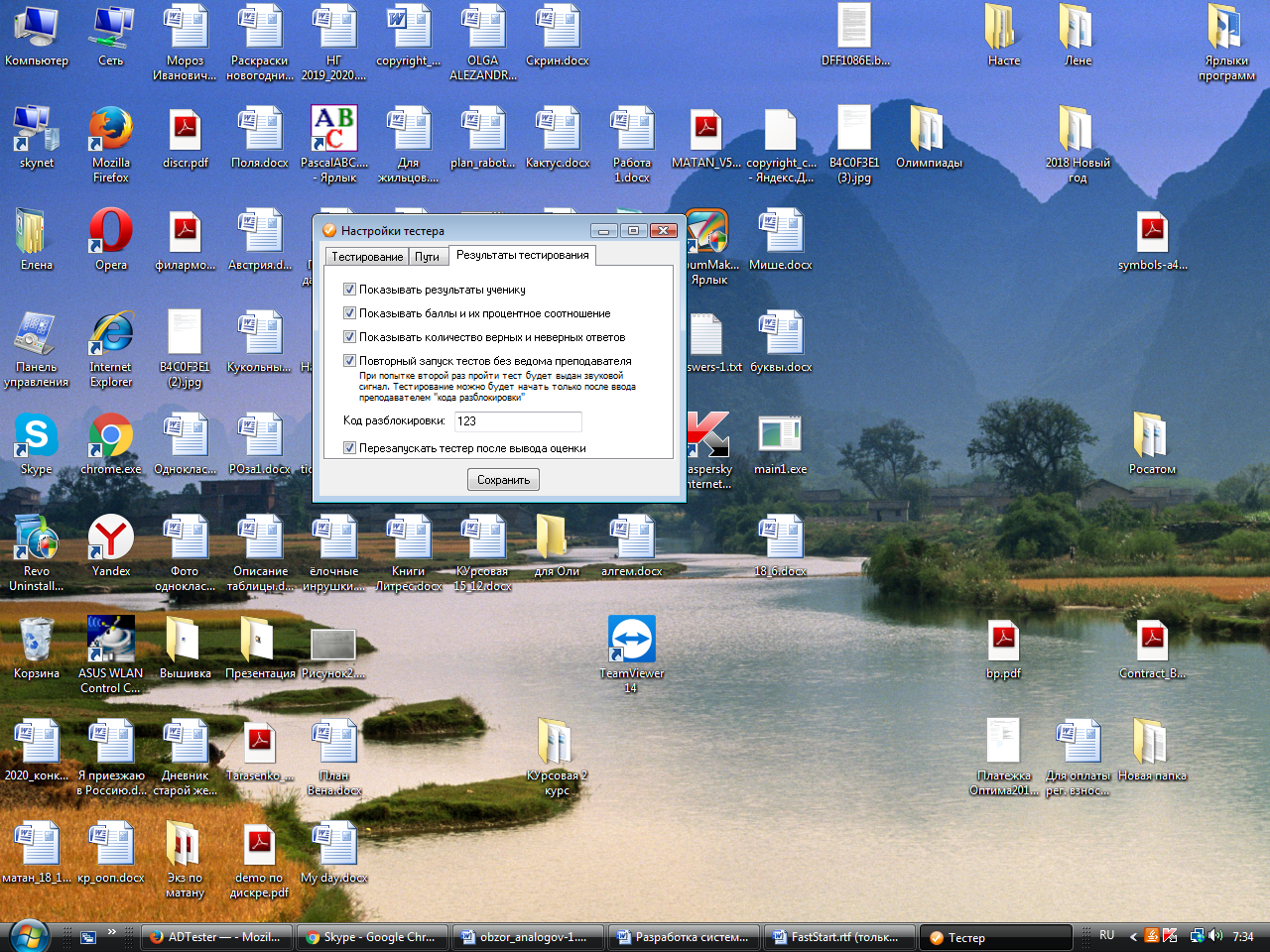


Рис. 2. - Внешний вид программы ADTester.

Программы пакета:

* Конструктор тестов - программа для создания тестов. Позволяет создавать и редактировать тесты;
* Тестер - программа для проведения тестирования. Имеет простой интерфейс и параметры проведения тестирования;
* Админ панель - программа администрирования. Позволяет производить манипуляции с пользователями и группами пользователей, строить матрицы правильности и производить анализ результатов тестирования.

В тестах можно использовать различные шрифты, формулы, схемы, таблицы, HTML документы и любые OLE-объекты. Вопросы и ответы можно полноценно форматировать - различные шрифты, абзацы, списки и т.д.

Возможности пакета:

Достаточно большое возможное количество вопросов в тесте и вариантов ответа.

Типы вопросов:

* одиночный выбор;
* множественный выбор;
* ввод ответа с клавиатуры;
* соответствие;
* порядок.

Возможно установить режим тестирования: контроль и обучение. В режиме обучения при неправильном ответе можно посмотреть подсказку по данному вопросу, результаты тестирования не учитываются. В режиме контроля ведется статистика прохождения теста. Каждый вопрос имеет свой «вес» (цену в баллах). Каждый вопрос может сопровождаться подсказкой для тестируемого. Редактирование теста может быть защищено паролем. Тестирование может быть ограничено по времени.

1. **Moodle3 (рис.3)**

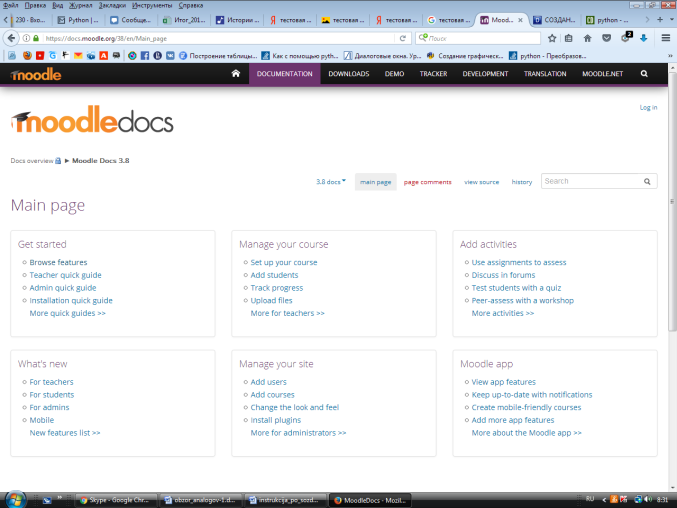


Рис. 3. - Внешний вид системы Moodle.

Moodle (Modular Object-Oriented Dynamic Learning Environment) - бесплатная система электронного обучения, ориентированная прежде всего на организацию взаимодействия между преподавателем и учениками, подходит и для организации традиционных дистанционных курсов, и для поддержки очного обучения.

Несмотря на то, что создание тестов не является целевой задачей платформы, оно реализовано как одна из возможностей для дистанционного курса. Тест Moodle невозможно создать без привязки к банку заданий, то есть генерация теста возможна только после заполнения банка вопросов. Для создания вопроса необходимо заполнить такие поля как: категория вопроса, название вопроса, текст вопроса, балл по умолчанию (то есть можно создавать тесты с разными по сложности вопросами, что будет учитываться в оценивании) и штраф за неправильную попытку.

# Темы, включённые в систему автоматической генерации

По предложению руководителя для курсовой были выбраны следующие темы:

1. Математические основы ВТ (Действия с числами в разных системах счисления)
   1. Перевод из одной системы счисления в другую
      1. Перевод целых чисел из 10-ичной системы в 2-ичную систему счисления
      2. Перевод целых чисел из 2-ичной системы в 10-ичную систему счисления
      3. Перевод целых чисел из одной системы счисления в другую
      4. Перевод чисел с плавающей запятой
   2. Перевод целых чисел в дополнительный код
   3. Двоичная арифметика с целыми числами
2. Логические основы ВТ (Алгебра логики)
   1. Приведение к совершенной нормальной форме
      1. Приведение к совершенной конъюнктивной нормальной форме (СКНФ)
      2. Приведение к совершенной дизъюнктивной нормальной форме (СДНФ)
   2. Приведение к минимальной нормальной форме
      1. Приведение к минимальной конъюнктивной нормальной форме (МКНФ)
      2. Приведение к минимальной дизъюнктивной нормальной форме (МДНФ)
   3. Операции булевой алгебры
3. Основы теории информации
   1. Применение формулы сочетания без повторений
   2. Применение формулы сочетания с повторениями
   3. Применение формулы размещения без повторений
   4. Применение формулы размещения с повторениями

Задания для генерации были выбраны в соответствии с ПУД по дисциплине «Информатика».

# Анализ технических решений объекта разработки

## Выбор языка программирования для создания системы

Язык программирования Python был выбран по нескольким существенным критериям:

* Имеется существенный задел в изучении этого языка программирования (его основы были изучены на 1-м курсе), что позволяет ускорить процесс решения задачи;
* Реализуется возможность углубить полученные знания в программировании на языке Python;
* Python имеет достаточное количество библиотек, которые облегчают работу и делают код более читаемым и понятным;
* Python неприхотлив к платформе запуска, что упрощает тестирование работы ПО;
* Графическая библиотека Tkinter делает процесс создания интерфейса достаточно удобным;
* Имеется возможность выдавать результаты в файлы docx c помощью специального модуля;
* Максимальная читаемость и понятность кода на языке Python позволяет в дальнейшем редактировать его в зависимости от изменения потребностей.

## Используемые встроенные библиотеки и конкретные задачи, ими выполняемые

Для каждого из разделов используются свои специфические библиотеки. Ниже приведён их список.

|  |  |  |
| --- | --- | --- |
| **Для какого раздела** | **Название** | **Что делает** |
| Выдача результатов | python-docx | Работа с документами Word |
| Оконный интерфейс | Tkinter | Графическая библиотека |
| Основы теории информации | Itertools4 | Библиотека для работы с различными формулами комбинаторики |
| Функции рандомной генерации | Random | Библиотека, позволяющая получить псевдорандомные числа |
| Создание датафрейма, содержащего значения переменных a, b, c и значение функции F (применяется в разделе «Алгебра логики») | Pandas | Высокоуровневая библиотека, необходимая для анализа данных |
| Создание типа numpy array в заданиях на создание МКНФ, МДНФ, СКНФ и СДНФ | Numpy | Наиболее часто используемая Python библиотека |
| Связь программных файлов 3 разделов | Os | Модуль, позволяющий работать с операционной системой |

## Описание принципа работы программы

Приложение состоит из двух оконных форм. Первая форма используется для выбора темы разделов, по которой необходимо сформировать варианты задания (рис. 4).

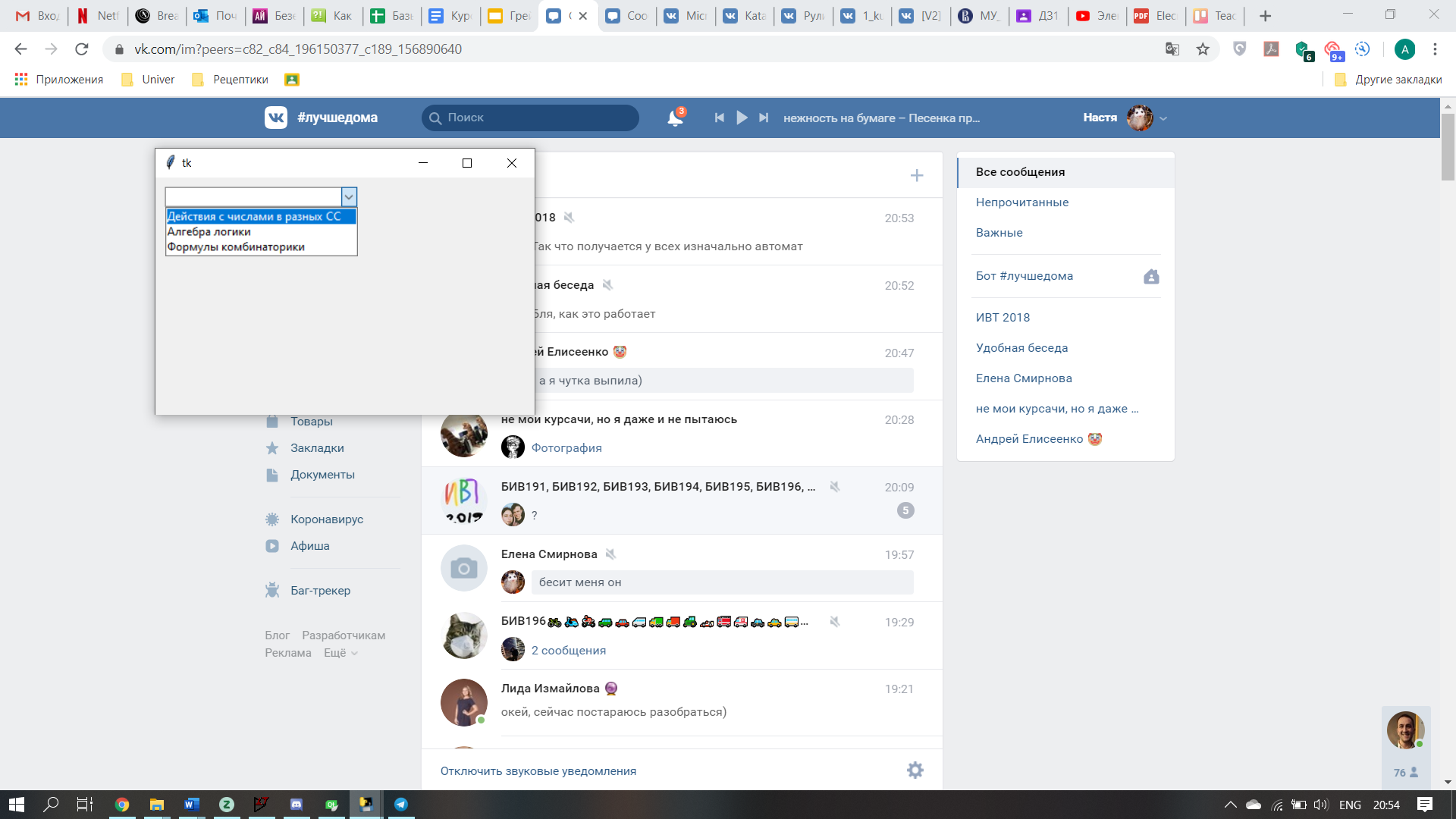


Рис. 4. Оконная форма выбора темы раздела.

Вторая оконная форма имеет три различных варианта, зависящих от выбора на рис. 4: оконная форма для формирования заданий по теме «Действия с числами в разных системах счисления» (рис.5), оконная форма для формирования заданий по теме «Алгебра логики» (рис.6), оконная форма для формирования заданий по теме «Формулы комбинаторики» (рис.7)

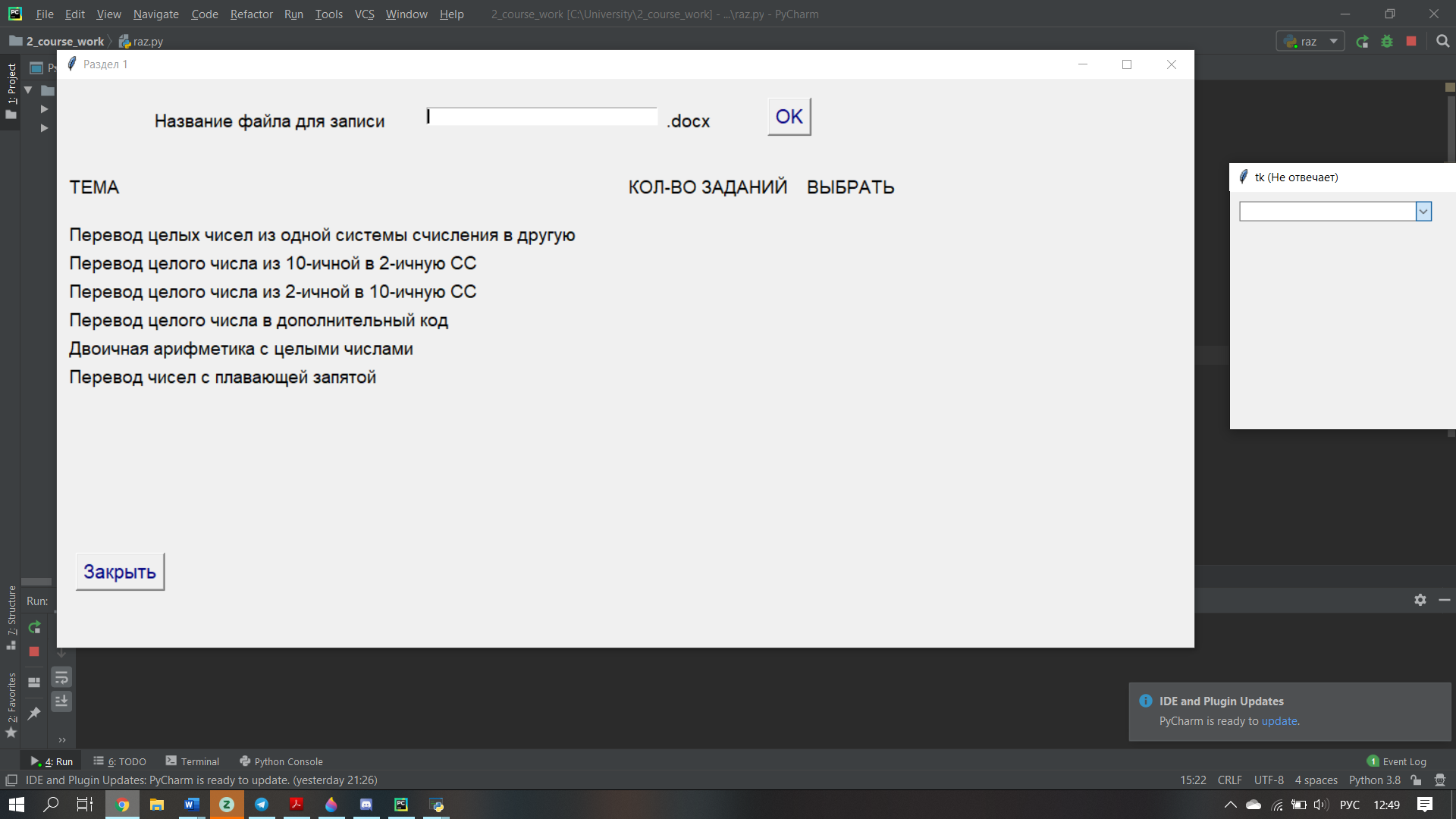


Рис. 5. Оконная форма для формирования заданий по теме

«Действия с числами в разных системах счисления»

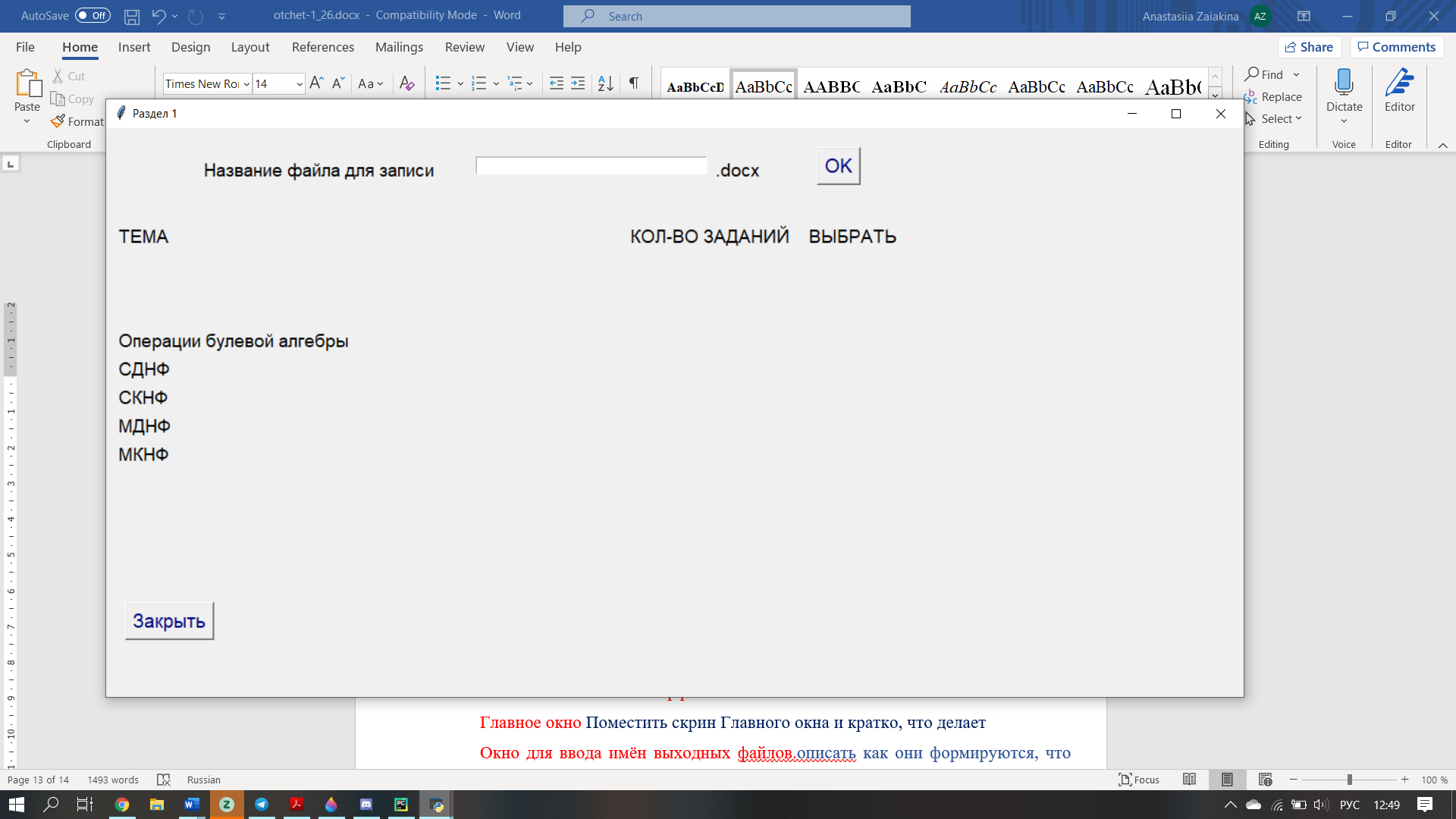


Рис. 6. Оконная форма для формирования заданий по теме «Алгебра логики».

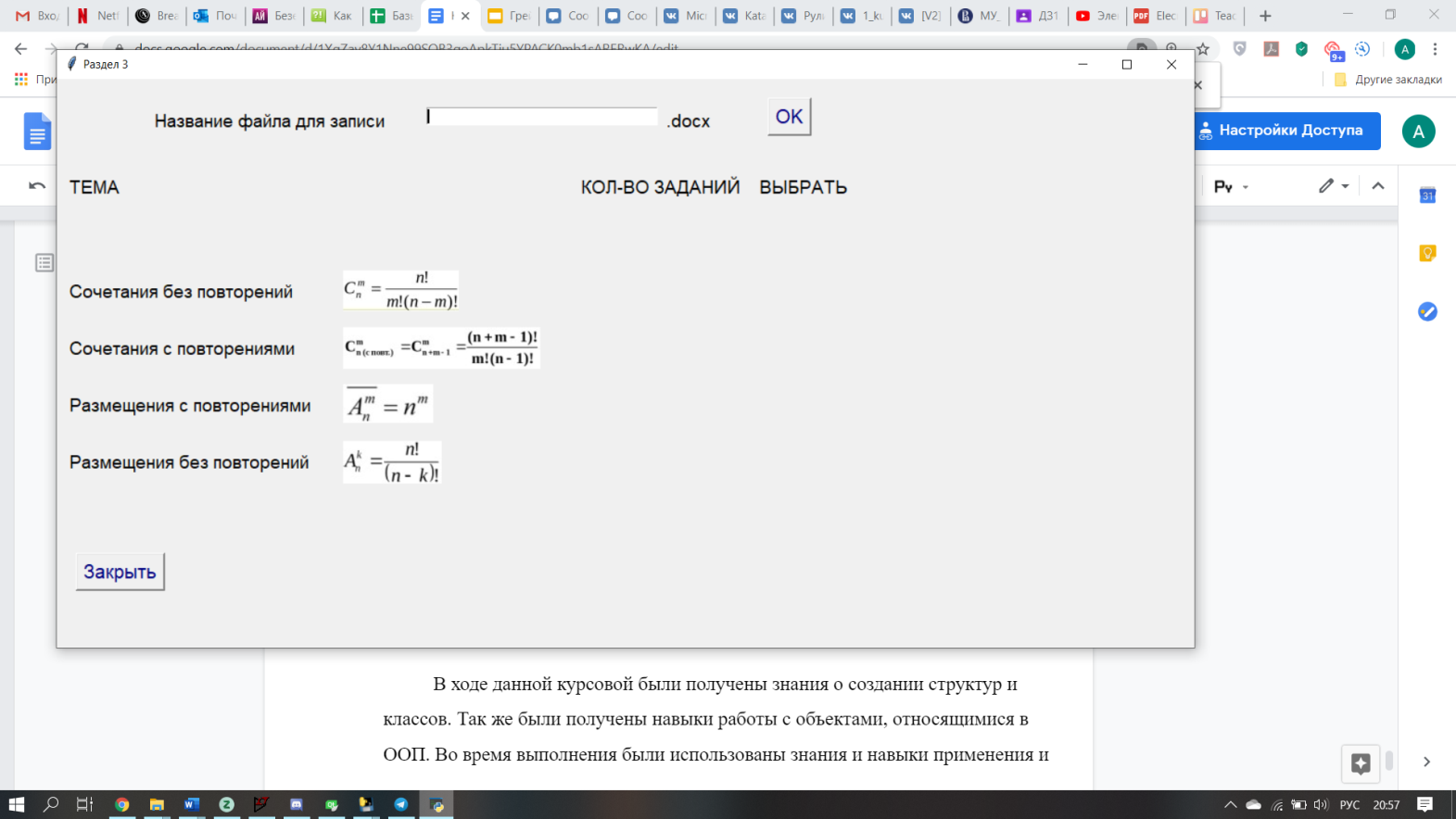


Рис 7. Оконная форма для формирования заданий по теме

«Формулы комбинаторики».

Каждый из вариантов может содержать более одного задания по необходимой теме. Количество заданий определяется пользователем во время формирования вариантов. Для ввода количества заданий необходимо вписать натуральное число в предназначенную для этого ячейку (рис. 8).

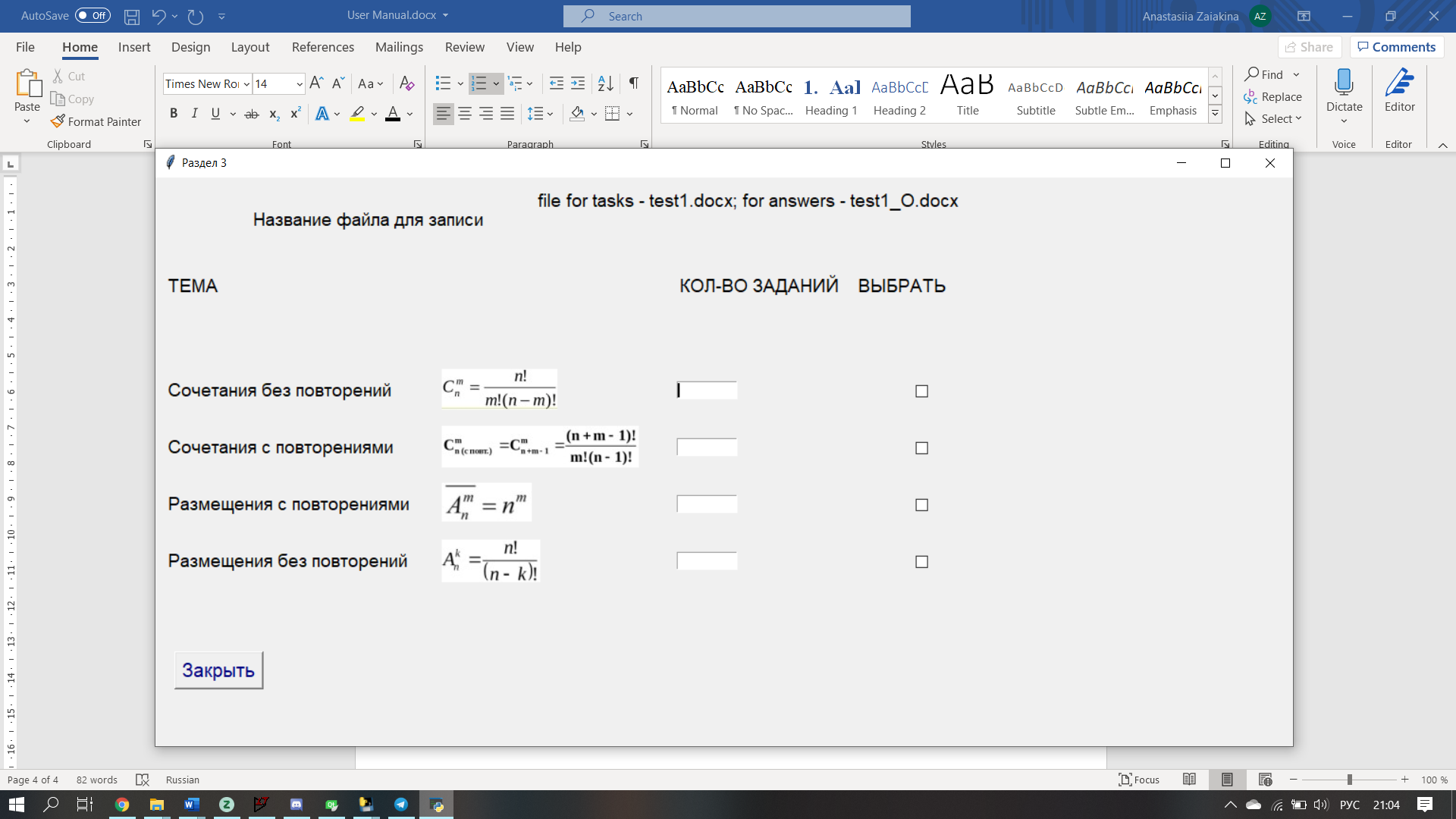


Рис. 8. Оконная форма, готовая для ввода количества заданий

Для подтверждения выбора и начала формирования вариантов пользователю необходимо активировать checkbutton в графе «Выбрать». После записи числа заданий, указанного в графе «Количество заданий», на месте checkbutton появится надпись «Выполнено» (рис. 9).

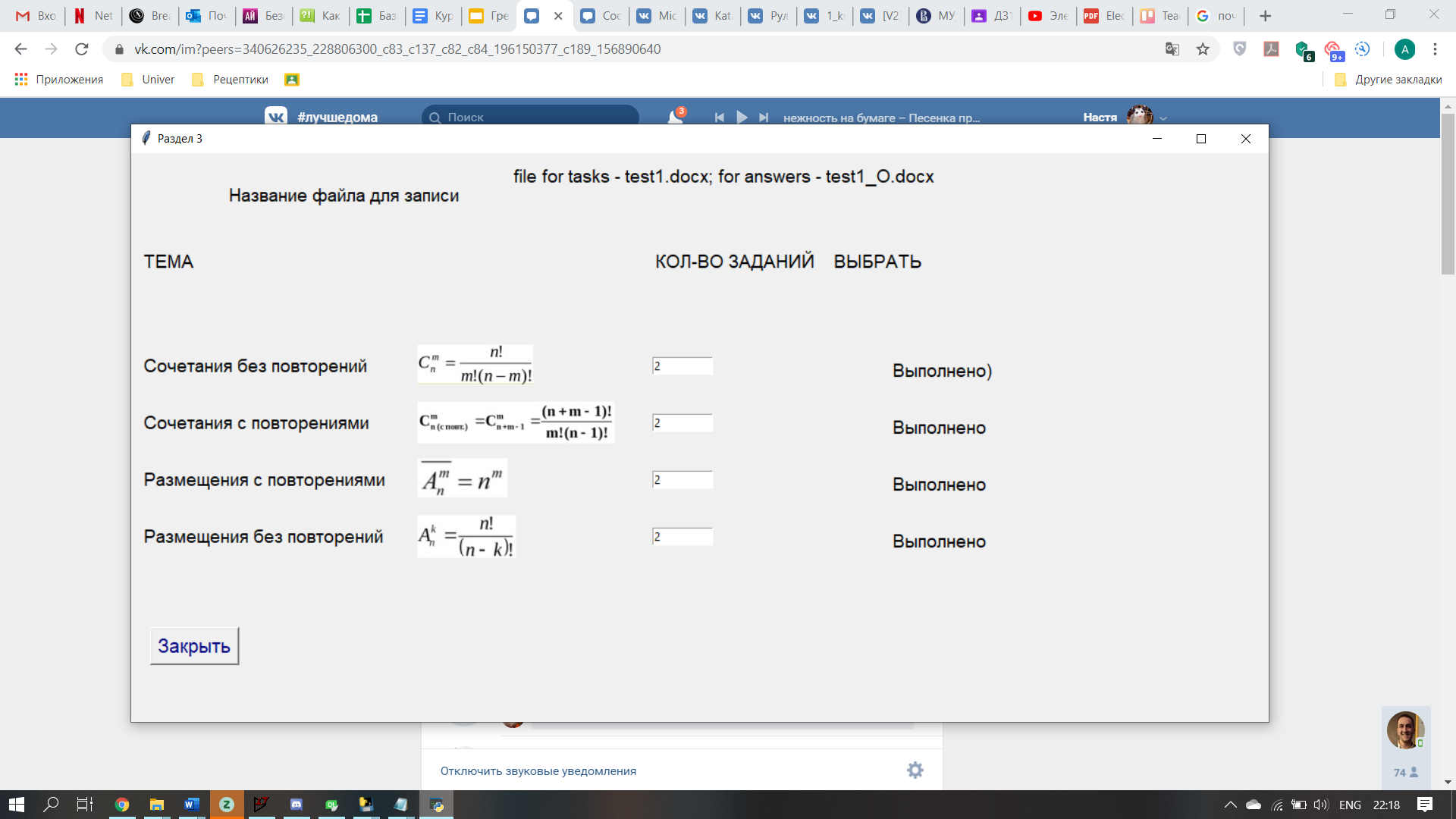


Рис. 9. Вид оконной формы после завершения записи заданий в файл

Для выполнения одного из описанных выше требований к системе, при активации checkbutton производится проверка введенного числа заданий на допустимость. Допустимыми для ввода считаются только натуральные числа. Поскольку это требование является естественным, дополнительная подсказка пользователю до ввода не является необходимой. В случае ошибки при введении количества заданий запись заданий в файл не производится и на экран выводится текст «Only integer numbers are available» (рис. 10).

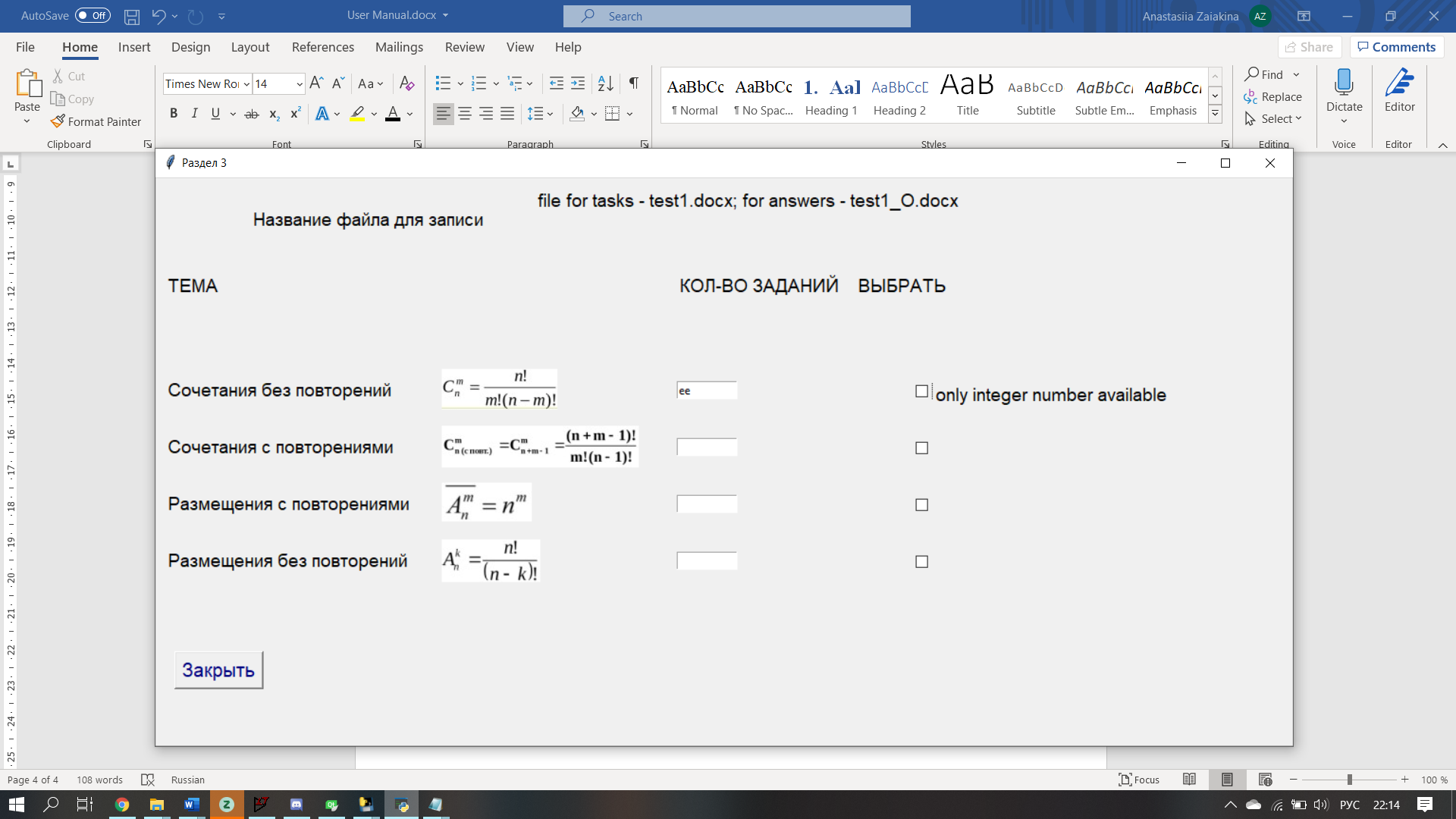


Рис. 10. Проверка на допустимость введённых значений

**Ввод имени файла для тестовых вариантов**

Для начала выбора необходимого задания и записи его в файл пользователю необходимо ввести название файла. По согласованию с научным руководителем курсовой работы было принято решение о том, что система будет формировать пять различных подобных вариантов. Дополнительно формируются файлы ответов к каждому из вариантов.

Название файлов с заданиями для учащегося строится по следующему принципу: < введённое пользователем название > + \_<номер варианта по порядку> + .docx

Название файлов с заданиями для преподавателя строится по следующему принципу: < введённое пользователем название > + <номер варианта по порядку> + \_О.docx

Файл с заданиями для преподавателя (рис. 11) идентичен файлу с заданиями для студента (рис. 12) с одним исключением. Для преподавателя так же выводятся ответы для каждого из заданий.

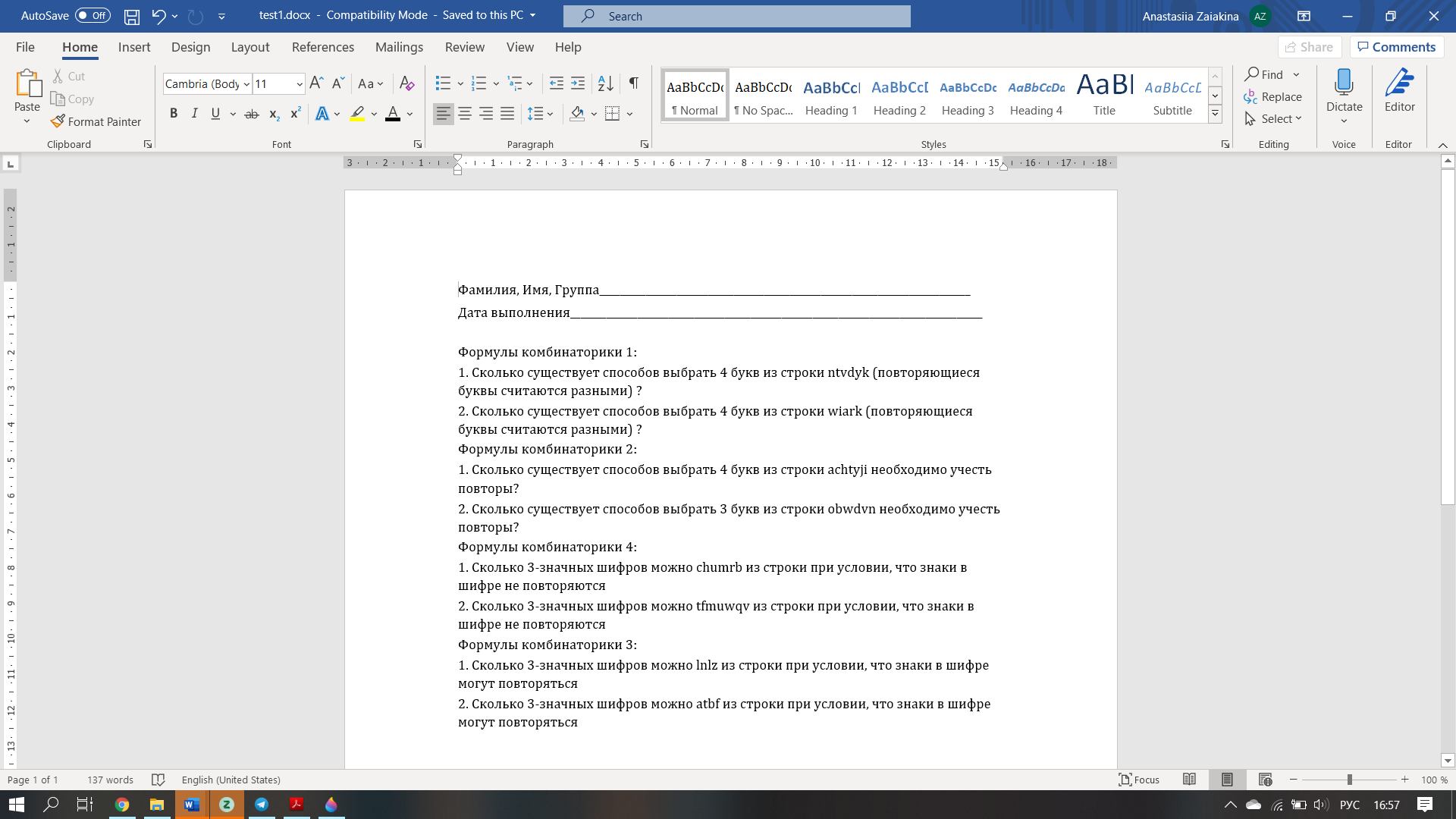


Рисунок 11. Файл с заданиями для студентов

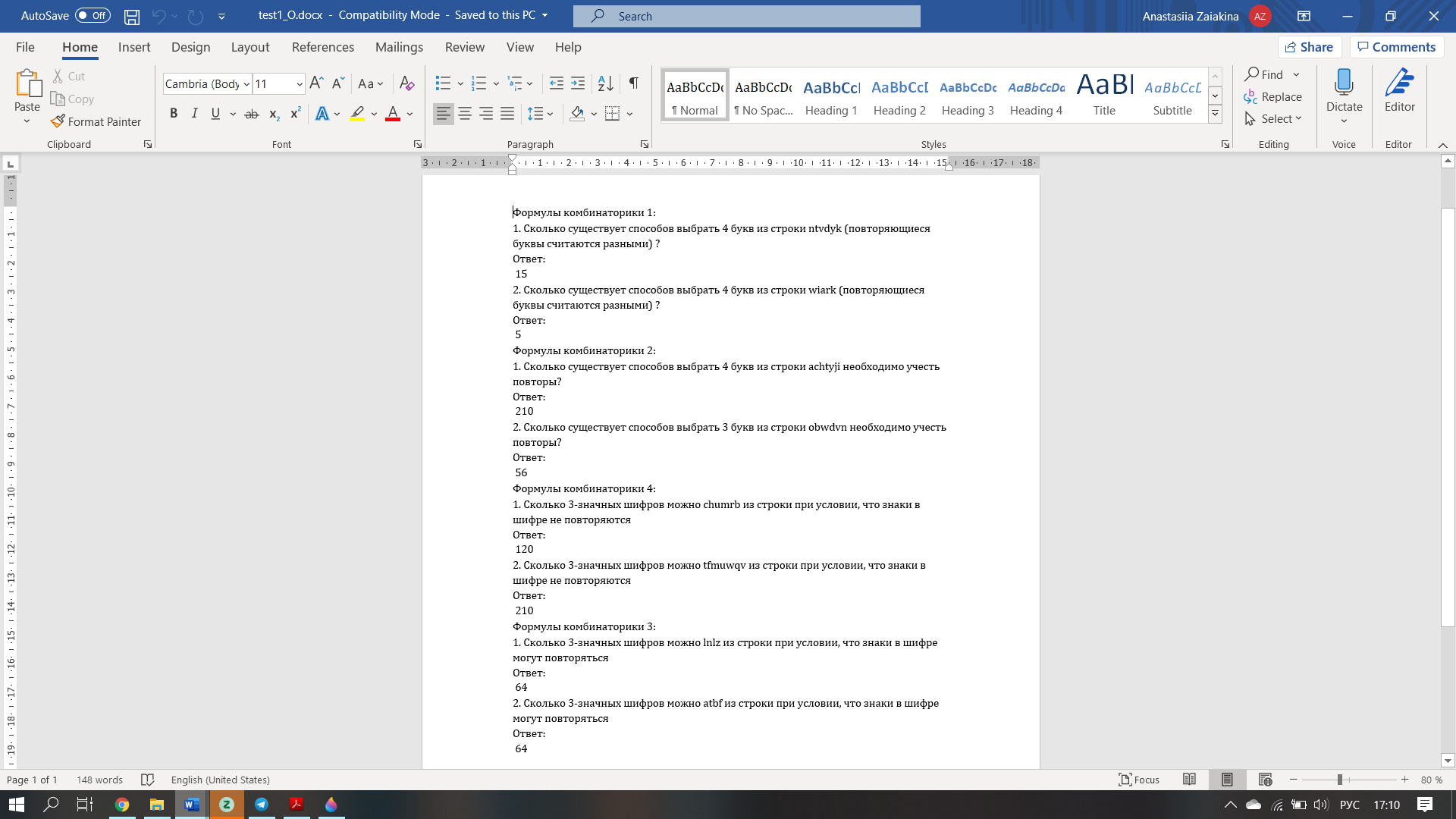


Рис. 12. Файл с заданиями для преподавателя

# Описание создания программного интерфейса

1. Главное окно для выбора раздела

window = Tk()

window.geometry('400x250')

combo = Combobox(values=["Действия с числами в разных СС ", "Алгебра логики", "Формулы комбинаторики"], width=30)

combo.bind('<<ComboboxSelected>>', get\_selected)

combo.place(x=10, y=10)

mainloop()

Основной часть главного окна является Combobox, позволяющий пользователю выбрать раздел из 3 представленных вариантов. После выбора начинает работу функция get\_selected, описанная в следующем разделе.

1. Общие параметры окон для формирования заданий по каждому из разделов

window1.geometry('1200x600')

btnputinfile = Button(window1, text="OK", font=14, command=put\_filename, foreground="#000080")

btnputinfile.place(x=750, y=20)

btnClose=Button(window1, text="Закрыть", font=14, сommand= window1.destroy, foreground="#000080")

btnClose.place(x=20, y=500)

lbl2 = Label(window1, font=("Arial Bold", 14), text="ТЕМА ")

lbl2.place(x=10, y=100)

lbl3 = Label(window1, font=("Arial Bold", 14), text="КОЛ-ВО ЗАДАНИЙ ВЫБРАТЬ")

lbl3.place(x=600, y=100)

window1.mainloop()

Приведённый выше код создаёт общий вид оконной формы, он универсален для всех трех разделов.

1. Ввод имени файла

fn = Entry(window1, width=40)

fn.focus\_set()

fn.place(x=390, y=30)

lbl8 = Label(window1, font=("Arial Bold", 14), text="Название файла для записи")

lbl8.place(x=100, y=30)

lbl9 = Label(window1, font=("Arial Bold", 14), text=".docx")

lbl9.place(x=640, y=30)

Данный код создаёт место, куда пользователь будет вводить название файла. Так же сразу после места для ввода указано расширение, чтобы показать, что его можно не вводить, а оно будет задано автоматически.

1. Добавление названия заданий (приведён пример для третьего раздела)

lbl4 = Label(window1, font=("Arial Bold", 14), text="Сочетания без повторений")

lbl5 = Label(window1, font=("Arial Bold", 14), text="Сочетания c повторениями")

lbl6 = Label(window1, font=("Arial Bold", 14), text="Размещения с повторениями")

lbl7 = Label(window1, font=("Arial Bold", 14), text="Размещения без повторений")

lbl4.place(x=10, y=210)

lbl5.place(x=10, y=270)

lbl6.place(x=10, y=330)

lbl7.place(x=10, y=390)

В данном случае создается 4 надписи, для которых затем задается расположение в окне.

1. В третьем разделе помимо названия задания используется формула, которая в нём используется. Приведён пример вставки формулы в окно.

path4 = 'rbp.jpg'

img4 = ImageTk.PhotoImage(Image.open(path4))

panel4 = tk.Label(window1, image=img4)

panel4.pack(side="bottom", fill="both", expand="yes")

panel4.place(x=300, y=380)

Для вставки формулы было принято решение использовать изображение из-за максимальной простоты работы с ним. При необходимости поместить другое изображение, размещенное на компьютере пользователя, достаточно изменить path4.

# Описание работы ключевых фрагментов программы

Структурно программа состоит из 3 файлов, создающих задания для каждого из разделов, и файла, объединяющего и связывающего из работу.

Обратим внимание на ключевые фрагменты, определяющие работоспособность системы.

1. Функция добавления задания и ответа к нему в соответствующие файлы

def input\_file\_docx(str\_write, str\_answer):

paragraph = dti.add\_paragraph(str\_write)

paragraph\_format = paragraph.paragraph\_format

paragraph\_format.space\_after = Pt(1.0)

paragraph = dti1.add\_paragraph(str\_answer)

paragraph\_format = paragraph.paragraph\_format

paragraph\_format.space\_after = Pt(1.0)

Данная функция вызывается в конце работы функций, автоматически генерирующих задания с ответами по заданной теме. Она взаимодействует непосредственно с файлом .docx, записывая в него строки, поданные на вход функции.

1. Функция, добавляющая графу для записи имени, группы и даты проведения работы в файл, создающийся для учащегося

def fio\_add(dti):

paragraph = dti.add\_paragraph('Фамилия, Имя, Группа' + '\_' \* 72)

paragraph\_format = paragraph.paragraph\_format

paragraph\_format.space\_after = Pt(3.0)

paragraph = dti.add\_paragraph('Дата выполнения' + '\_' \* 80 + '\n')

paragraph\_format = paragraph.paragraph\_format

paragraph\_format.space\_after = Pt(2.0)

Данная функция вызывается сразу после ввода пользователем имени файла и созданием программой файлов, взаимодействует непосредственно с файлами .docx, предназначенными для учащегося.

1. Функция, регистрирующая нажатие на Checkbutton и запускающая соответствующую ему команду.

def flag(i, com, xi, yi):

chk = Checkbutton(window1, variable=list\_cb[i], command=com)

chk.focus()

chk.place(x=xi, y=yi)

return

Данная функция используется для связи расположения checkbutton и команды, которая должна запускаться при его активации.

1. Функции, обеспечивающие корректность вводимых данных
   1. Проверка является ли введенная строка натуральным числом

def only\_int(p):

if p.isdigit():

return True

return False

На вход функции передаётся строка, которая проверяется на то, является ли она натуральным числом. На выход функции подаётся ложь или истина в зависимости от строки.

* 1. Функция, выводящая подсказку для пользователя о введении неверного значения строки

def num\_check(xi, yi, li):

"""

Function put on a screen text if user try to put anything besides integer number

lb\_f = Label(window1, font=("Arial Bold", 14), text='only integer number available')

lb\_f.place(x=xi, y=yi)

list\_cb[li].set(0)

return

Функция активируется в случае, если на выходе функции only\_int оказалось значение False. Подсказка выводится рядом с Checkbutton, соответствующему месту ввода неверного значения.

1. Функция, определяющая какой из 3 файлов программ необходимо запустить для продолжения работы.

def get\_selected(param):

icombo = combo.current() + 1

if icombo == 1:

os.system('python part\_1\_otl\_1.py')

elif icombo == 2:

os.system('python part\_2\_otl\_1.py')

else:

os.system('python part\_3\_otl\_1.py')

Фиксируется номер по порядку выбранного раздела в combobox, и в зависимости от него запускается необходимая программа.

1. Связующая функция для каждого из разделов. Приведён пример для третьего раздела.

def combin():

"""

Function reads number from question\_amount and launches the desired function

:return:

"""

question\_amount\_1.place(x=550, y=215)

question\_amount\_2.place(x=550, y=275)

question\_amount\_3.place(x=550, y=335)

question\_amount\_4.place(x=550, y=395)

flag(1, comb\_n\_k, 800, 215)

flag(2, comp\_n\_k\_rep, 800, 275)

flag(3, accom\_n\_k\_rep, 800, 335)

flag(4, accom\_n\_k, 800, 395)

Данная функция считывает числа, введённые в поля question\_amount и последовательно вызывает функцию flag для каждого из заданий. В ходе выполнения данной функции в файлы заданий для учащихся и преподавателя добавляются выбранные задания, если значения, введёные в question\_amount являются натуральными числами, иначе для каждой ошибочно введённой строки необходимо повторить ввод.

# Заключение

В процессе выполнения курсовой работы было выполнено ознакомление с тестовыми системами, усовершенствование навыков программирования на языке программирования Python, освоение принципов использования библиотек этого языка, получила опыт в создании пользовательского интерфейса, изучены основные принципы тестирования программ. Была выполнена работа, которая будет иметь практическое использование.

Процесс работы над курсовой дал возможность применить, полученные в ходе обучения, знания на практике, приобретен опыт взаимодействия с постановщиком задачи.

# Список источников

1. Knowing [Электронный ресурс]. – Режим доступа : <http://www.globalpage.ru/> (Дата обращения 15.11.2019)
2. AdTester [Электронный ресурс]. – Режим доступа: <https://www.adtester.org/> (Дата обращения 16.11.2019)
3. Moodle [Электронный ресурс]. – Режим доступа: <https://moodle.org/> (Дата обращения 16.11.2019)
4. Python 3.8.3 documentation [Электронный ресурс]. – Режим доступа <https://docs.python.org/> (Дата обращения 03.01.2020 – 05.05.2020)
5. Matthes, Python Crash Course //No Starch Press, 2016
6. Обучение Tkinter [Электронный ресурс]. – Режим доступа: <https://younglinux.info/tkinter/tkinter.php> (Дата обращения 03.01.2020 - 05.05.2020)