**4.3 Разработка алгоритмов работы программы и программного обеспечения**

В данном разделе будут произведены разработки алгоритмов функционирования программы, разработан интерфейс программы, реализованы основные функции программы.

**4.3.1 Разработка программного обеспечения**

Программное обеспечение разработано в среде PyCharm на языке Python. Для реализации пользовательского интерфейса использовалась библиотека Tkinter. Исходный код представлен в приложении А.

В структуре данной программы присутствует импорт модулей: tkinter, random. Tkinter – библиотека для создания графического пользовательского интерфейса, используется для создания окон, кнопок и других виджетов. Модуль random используется для выбора загаданного компьютером числа.

Функция generate\_secret\_number() генерирует секретное четырехзначное или пятизначное число. Исходный код функции generate\_secret\_number() представлен на рисунке 1.

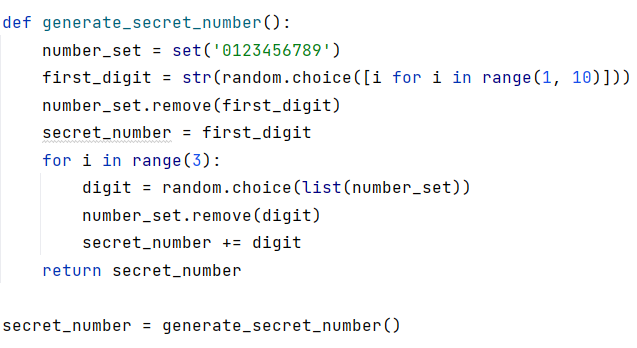


Рисунок 1 – Исходный код функции generate\_secret\_number()

check\_input\_number(number) – функция, которая проверяет введенное пользователем число на соответствие правилам игры. Она проверяет длину числа (должно быть 4 цифры на Уровне 1 и 5 цифр на Уровне 2), отсутствие повторяющихся цифр, а также начало числа не с нуля. Если число не соответствует правилам, функция отображает окно с сообщением об ошибке. Исходный код функции check\_input\_number(number) представлен на рисунке 2.

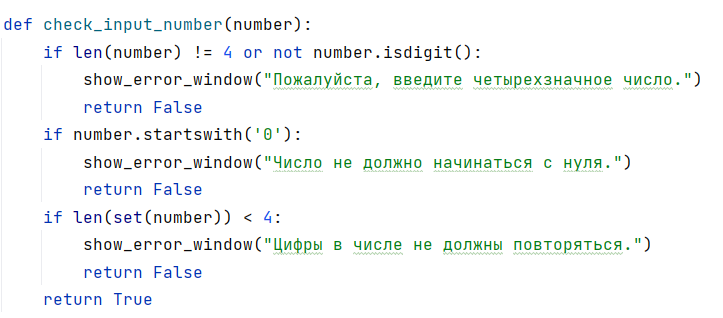


Рисунок 2 – Исходный код функции check\_input\_number(number)

start\_game() – функция, которая запускает логику игры. Пользовательский ввод проверяется с помощью check\_input\_number(), затем вычисляется количество быков и коров в числе пользователя, обновляется история ходов и проверяется условие победы или конца игры. Исходный код функции start\_game() представлен на рисунках 3 и 4.

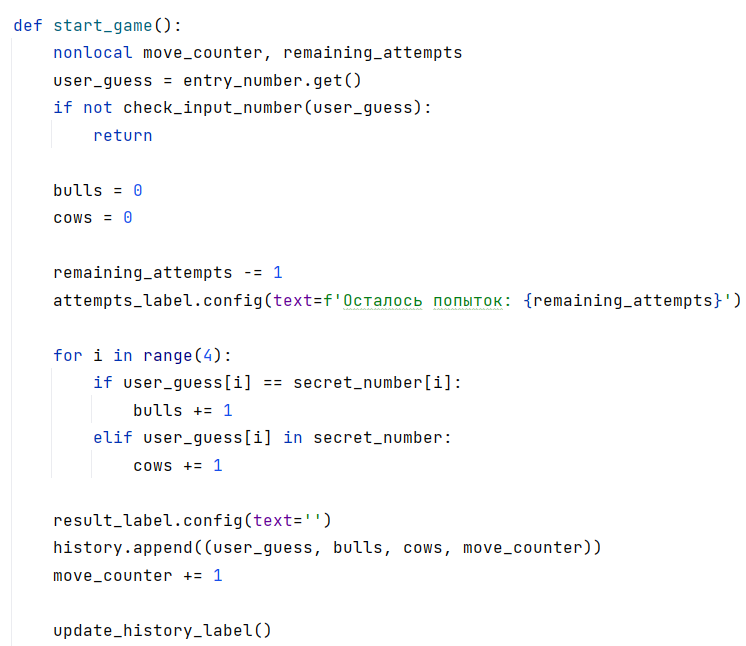


Рисунок 3 – Исходный код функции start\_game()

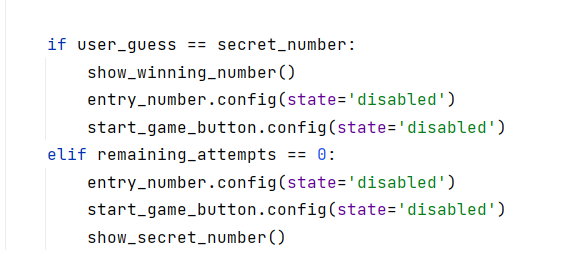


Рисунок 4 – Исходный код функции start\_game()

update\_history\_label() – функция, которая обновляет отображение истории ходов на экране. Для каждого хода отображает введенное пользователем число, количество быков и коров, а также номер хода. Данная функция обновляет информацию на экране после каждого хода. Исходный код функции update\_history\_label() представлен на рисунке 5.

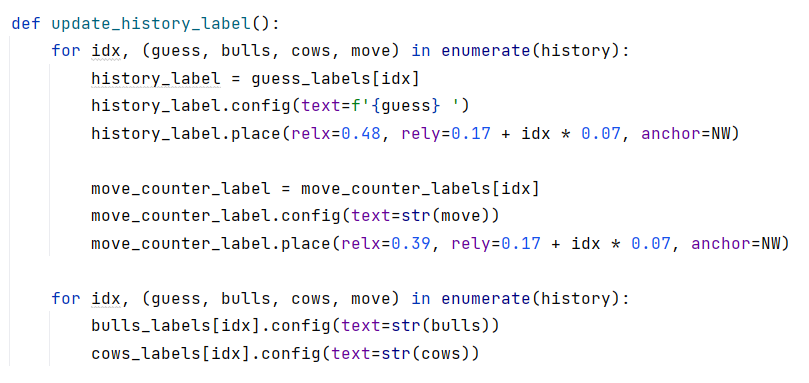


Рисунок 5 – Исходный код функции update\_history\_label()

show\_error\_window(message) – функция, которая отображает окно с сообщением об ошибке. В случае некорректного ввода числа пользователем, эта функция позволяет показать сообщение об ошибке. Исходный код функции show\_error\_window(message) представлен на рисунке 6.

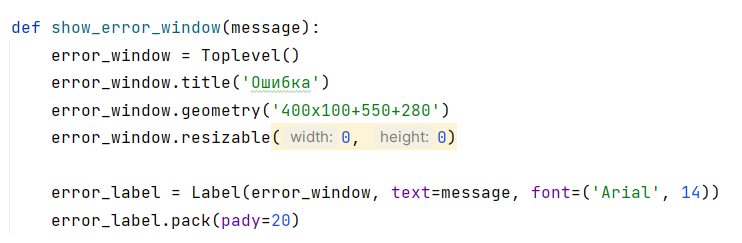


Рисунок 6 – Исходный код функции show\_error\_window(message)

reset\_game() – функция, которая сбрасывает игру в начальное состояние. Она обнуляет счётчики ходов и оставшихся попыток, генерирует новое секретное число, очищает историю ходов и возвращает поля и кнопки к исходному состоянию. Исходный код функции reset\_game() представлен на рисунке 7.

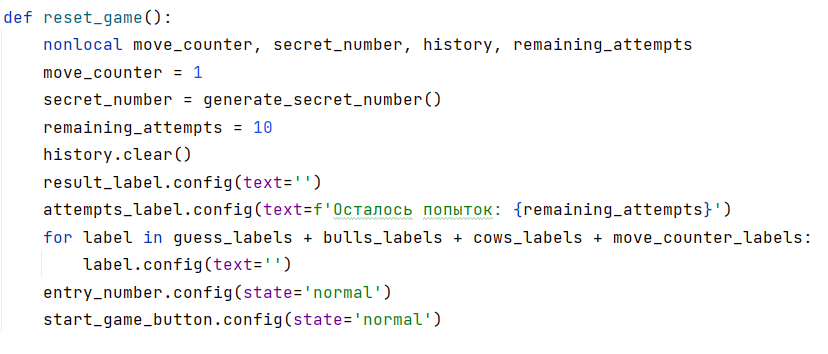


Рисунок 7 – Исходный код функции reset\_game()

show\_secret\_number() – функция, которая отображает окно с сообщением об окончании игры, если пользователь исчерпал все попытки угадывания и проиграл. В окне отображается секретное число, которое было загадано. Исходный код функции представлен на рисунке 8.

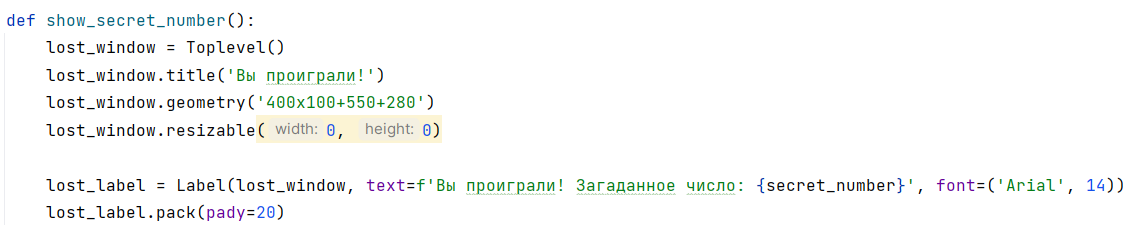


Рисунок 8 – Исходный код функции show\_secret\_number()

show\_winning\_number() – функция, отображающая окно с сообщением о победе игрока, если пользователь угадал секретное число. Исходный код функции представлен на рисунке 9.

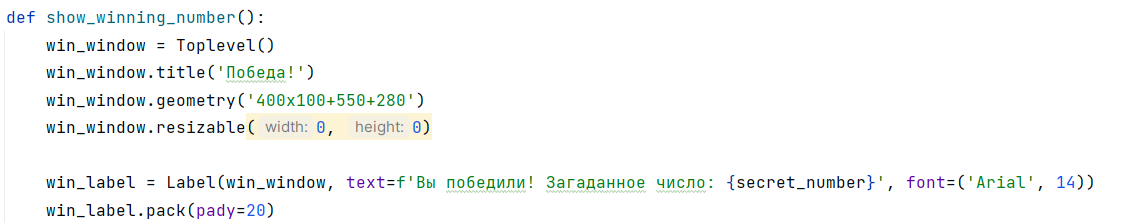


Рисунок 9 – Исходный код функции show\_winning\_number()

**4.3.2 Разработка пользовательского интерфейса**

На рисунке 10 представлен главный экран интерфейса приложения «Быки и коровы».



Рисунок 10 – Главный экран интерфейса

Для создания главного окна приложения используется класс Tk() из библиотеки Tkinter. С помощью команды title() создается заголовок «Быки и коровы» (также используется для создания заголовка «Выбор уровня сложности» рис. 11, «Уровень 1» рис. 12 и «Уровень 2» рис. 13), geometry() устанавливает размеры и позицию окна на экране (также используется для интерфейса рис. 11, рис. 12 и рис. 13).

Класс PhotoImage используется для загрузки изображения для фона интерфейса, Label отображает изображение на экране (PhotoImage и Label также используются для интерфейса рис. 11, рис. 12 и рис. 13).

Button cоздает кнопки с заданными параметрами текста, цвета фона, шрифта и функции, которая будет выполнена при нажатии. К кнопке «Выбор уровня сложности» привязана функция open\_difficulty\_window (используется для кнопок «Уровень 1» и «Уровень 2» рис. 11), с помощью которой открывается другое окно. К кнопке «Выход из игры» привязана функция exit\_game, которая обеспечивает завершение работы программы. resizable(0, 0) (также используется для интерфейса рис.11, рис. 12 и рис. 13) блокирует изменение размеров окна по ширине и высоте.

На рисунке 11 представлен экран выбора уровня сложности интерфейса приложения «Быки и коровы».

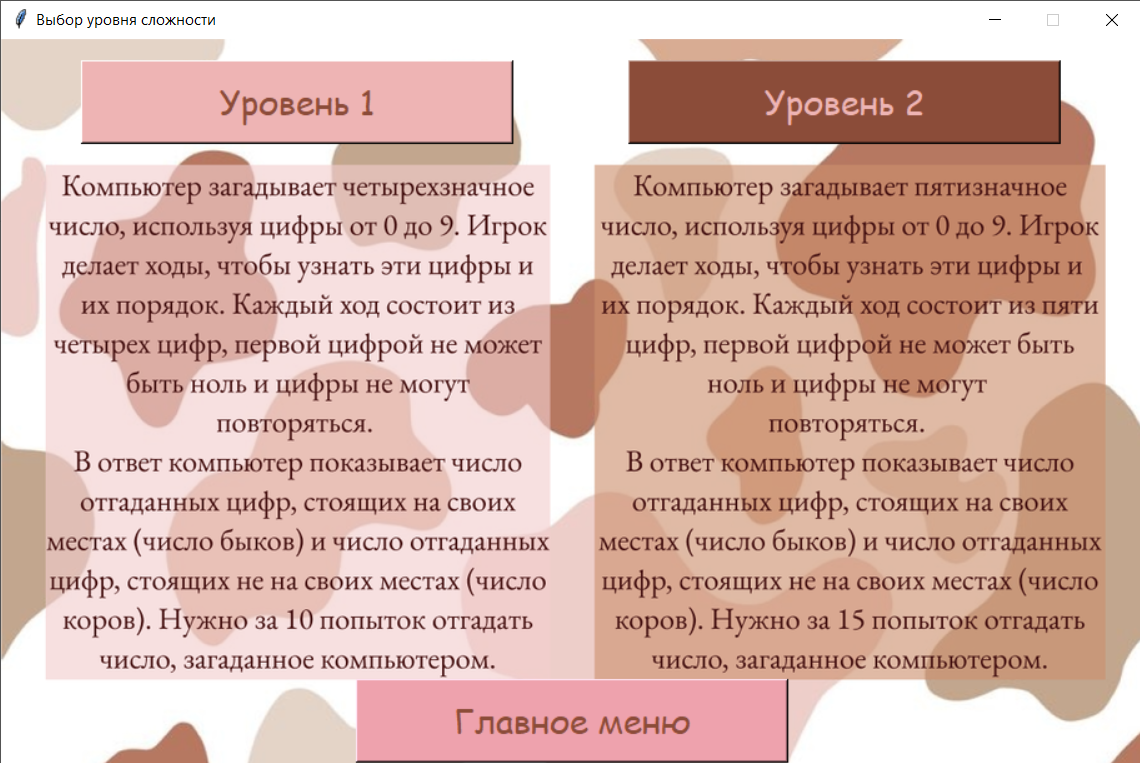


Рисунок 11 – Экран выбора уровня сложности

Для создания экрана выбора уровня сложности используется класс Toplevel() из библиотеки Tkinter (также используется для окон «Уровень 1» рис. 12 и «Уровень 2» рис. 13), создающий дочернее окно приложения. Команда root.withdraw() в Tkinter используется для скрытия главного окна приложения. Окно становится невидимым для пользователя, но при этом остаётся активным в оперативной памяти и может быть в дальнейшем восстановлено с помощью команды .deiconify().

К кнопке «Главное меню» привязана функция go\_back(difficulty\_window), предназначенная для возврата к предыдущему окну путем уничтожения текущего окна и повторного отображения главного окна (которое было скрыто с помощью root.withdraw()), используя команду root.deiconify().

На рисунках 12 и 13 представлен экран игрового окна интерфейса приложения «Быки и коровы».

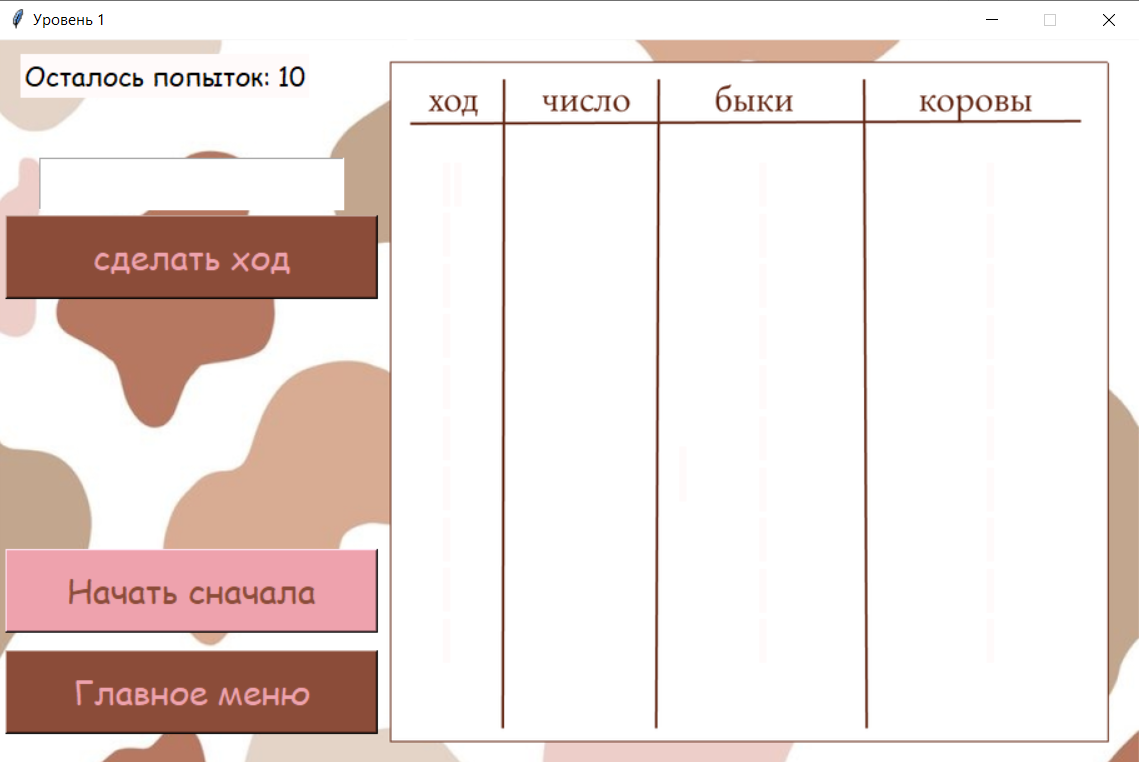


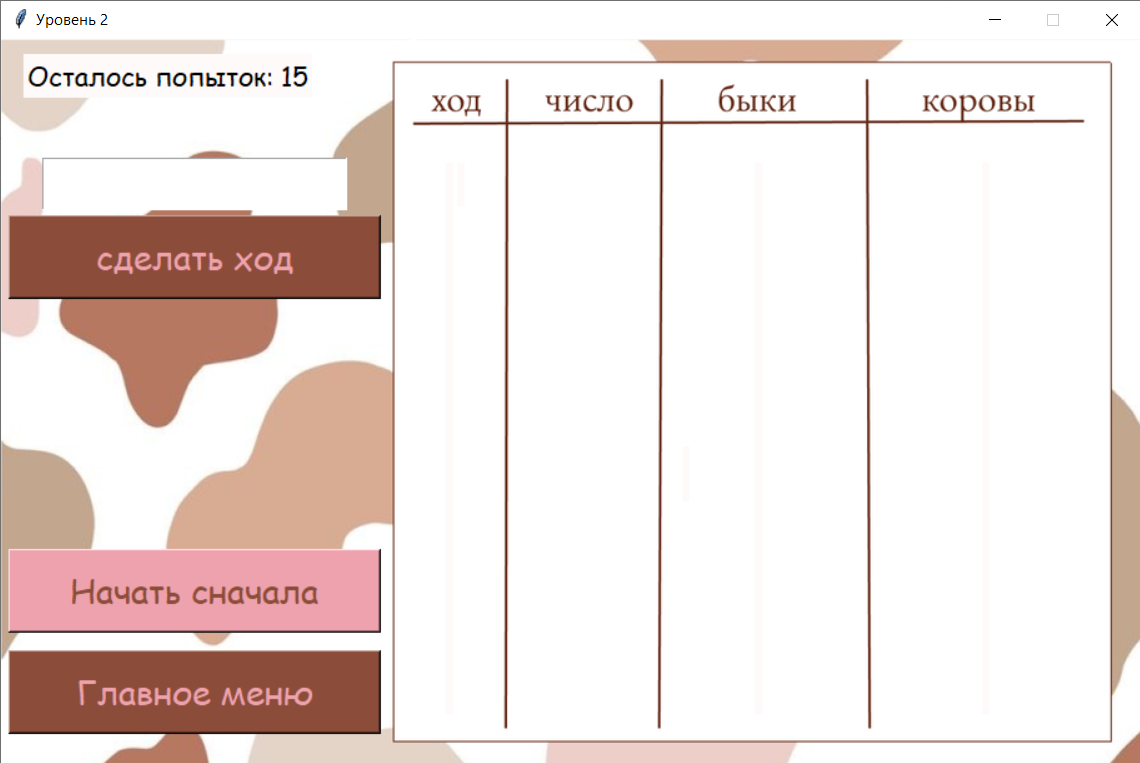
Рисунок 12 – Экран игрового окна Уровень 1

Рисунок 13 – Экран игрового окна Уровень 2

К кнопке «сделать ход» привязана функция start\_game(), которая запускает логику игры, обрабатывает введенные данные из entry\_number (поле для ввода числа), проверяет правильность хода и выводит результат (быков и коров) в result\_label. К кнопке «Главное меню» привязана функция go\_back\_to\_main(new\_window), которая отвечает за возвращение к начальному экрану приложения. Она скрывает текущее окно `new\_window` и восстанавливает главное окно. К кнопке «Начать сначала» привязана функция reset\_game(), которая перезапускает игру с нуля, сбрасывая все результаты и начальные значения.

На экране есть поле для ввода, созданное с помощью entry\_number, предназначенное для того, чтобы пользователь мог вводить числа, необходимые для игры.

С помощью attempts\_label создается виджет Label, который представляет собой текстовую метку, отображаемую на окне. Текст в метке включает информацию о количестве оставшихся попыток при угадывании числа. Переменная remaining\_attempts используется для динамического обновления количества попыток в тексте.

**Вывод**

Таким образом, были изучены различные ресурсы по программированию на языке Python, благодаря чему мы поняли как правильно и грамотно писать код, познакомились с различными полезными функциями. Так же для реализации приложения было необходимо создать графический интерфейс пользователя, для этого мы изучили разные библиотеки доступные на Python. Наш выбор пал на библиотеку Tkinter, при использовании которой был получен навык создания GUI. Так же при написании кода применялся модуль random, благодаря чему приобретен навык использования функций модуля для разных целей.