МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИИ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛЬ – ХОРЕЗМИ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Кафедра «Системное и прикладное программирование»

Индивидуальный проект

По дисциплине «Системное программное обеспечение»

На тему: «Разработка генератора календаря»

Выполнил: студент 3 курса группы 321-20ДИФ

Исмагилов Дамир _____

ОЦЕНКА: _____

ТАШКЕНТ – 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном обществе, где эффективное использование времени играет важную роль, календари становятся неотъемлемой частью нашей повседневной жизни. Они позволяют нам планировать наши дела, устанавливать важные сроки, отслеживать события и встречи. Однако, ручное создание и ведение календарей может быть трудоемким и подвержено ошибкам. В таких случаях программное решение для генерации и отображения календарей становится необходимостью.

Целью данной индивидуальной работы является разработка программы для генерации и отображения календаря с использованием графического интерфейса. В рамках работы будет реализована программа на основе библиотеки Tkinter, позволяющая пользователю вводить год и месяц, а затем отображать соответствующий календарь на экране.

Для достижения данной цели необходимо решить следующие задачи:

1. изучение предметной области генерации календарей;
2. анализ существующих рынков электронных календарей;
3. Составление задачи и целей для разработки генератора календаря;
4. Проведение реализация программы, включающая в себя взаимодействие с пользователем, генерацию календаря и отображение дат.
5. Проведение тестирования разработанного программного решения для проверки его работоспособности и корректности работы.

Результатом выполнения данной индивидуальной работы будет готовая программа, способная генерировать календарь и отображать даты выбранного пользователем года и месяца. Также будет проведен анализ результатов тестирования.

Данная индивидуальная работа актуальна, поскольку представляет практическую ценность для широкого круга пользователей, которым

необходимо эффективное планирование времени. Программа может быть полезна как для индивидуального использования, так и для организаций, которые нуждаются в удобном и надежном инструменте для работы с календарями.

В следующей главе работы будет проведен обзор предметной области, включающий изучение существующих методов генерации календарей и отображения дат, анализ рынка электронных календарей.

ОСНОВНАЯ ЧАСТЬ

ГЛАВА 1: ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Изучение понятий календаря, генерации календарей

Календарь -это систематический способ организации и измерении времени, который используется для определения дат, дней недели, месяцев и лет. Календарь помогает людям сориентироваться во времени, планировать события, управлять расписаниями и отслеживать время прошедшее и предстоящее.

Календари обычно основаны на различных системах и циклах, которые учитывают движение Земли, Луны, Солнца и других астрономических факторов. Они могут включать в себя такие элементы, как годы, месяцы, недели, дни, часы и минуты.

Помимо того, существуют несколько типов календарей, таких как грегорианский, юлианский, лунный и т.д. Каждый из них имеет свои особенности и правила расчета дней и месяцев.

Календари также могут включать в себя дополнительные информации, такие как как праздники, памятные даты, дни недели и другие события.

Генерация календарей - это процесс, связанный с созданием календарных дней на компьютере или другом устройстве, которые представляют собой расписание дней, недель, месяцев и лет в определенном временном интервале. Генерация календарей может выполняться вручную или автоматически с использованием программного обеспечения.

Существуют несколько шагов генерации календаря:

- Определение параметров: необходимо определить параметры для создания календаря, такие как год, месяц, формат отображения и другие настройки.
- Расчет дней и недель: используя определенные параметры, производится расчет дней и недель для соответствующего периода. Это может включать определение первого дня недели, определение високосных лет, учет праздников и других особенностей календаря.
- Форматирование и отображение: сгенерированные данные календаря форматируются и отображаются в удобочитаемом виде. Это может включать размещение дней и недель в таблицу, использование различных цветов и стилей для выделения определенных дат и отображение другим цветом, такие как праздники или события.
- Обработка особых случаев: во время генерации календарей могут возникать особые случаи, такие как високосные годы, переход между месяцами или учет праздничных дней. Эти случаи должны быть учтены

и обработаны соответствующим образом.

Для генерации календарей в программе обычно используются алгоритмы, основанные на стандартных календарных функциях и правилах. Библиотеки программирования часто предоставляют функции для работы с календарями, которые позволяют определить количество дней в месяце, день недели для конкретной даты, проверить високосный год и другие календарные операции.

1.2. Отличие от бумажных календарей

Главное отличие программы от бумажных календарей заключается в ее интерактивности и возможности генерировать календари для любого заданного года и месяца.

В отличие от бумажных календарей, где все даты уже напечатаны заранее, программа позволяет пользователю выбрать конкретный год и месяц, для которого он хочет увидеть календарь. Это дает гибкость и удобство при планировании событий или просмотре календаря на определенный период и нет необходимости иметь отдельные бумажные календари для каждого года или ждать выхода нового календаря на следующий год.

Программа отмечает выходные дни и праздники разными цветами, что помогает быстро определить эти даты при просмотре календаря. Это удобно для планирования отпусков, праздников или других событий, связанных с выходными и праздничными днями.

Кроме того, программа также предоставляет функции перехода к предыдущему и следующему месяцу, что позволяет быстро просматривать календарь для любого месяца и года без необходимости перелистывания страниц или поиска нужной даты и легко переключаться по календарю без необходимости иметь отдельные страницы для каждого месяца, как в случае с бумажными календарями. Все даты отображаются на экране сразу, и пользователь может легко найти нужную информацию.

Использование программы для генерации календаря вместо покупки бумажного календаря помогает снизить потребление бумаги и сократить отходы. Это важно с точки зрения экологической ответственности и сохранения природных ресурсов.

В целом, программа предлагает удобный, гибкий и интерактивный способ использования календаря, с преимуществами в доступности, обновлении и дополнительных функциях по сравнению с бумажными календарями.

1.3. Преимущества и недостатки программы

Преимущества программы:

1. Гибкость: Программа позволяет пользователю выбрать любой год и месяц для генерации календаря. Это обеспечивает гибкость в планировании и просмотре событий на определенные периоды.

2. Интерактивность: Программа предоставляет графический интерфейс, что делает ее более удобной в использовании и навигации. Пользователь может взаимодействовать с элементами интерфейса и получать наглядную информацию о днях недели, датах, выходных и праздничных днях.

3. Отметка выходных и праздничных дней: Программа отмечает выходные и праздничные дни различными цветами, что позволяет пользователю быстро и легко определить эти даты при просмотре календаря.

4. Возможность перехода к предыдущему и следующему месяцу: Программа позволяет пользователю легко переключаться между месяцами, что облегчает просмотр и планирование на разные периоды времени.

5. Обновление и модификация: Программа может быть обновлена или модифицирована для добавления новых функций или исправления ошибок. Это позволяет сделать ее более удобной и функциональной с течением времени.

Недостатки программы:

1. Ограниченность функциональности: Программа ограничена функционалом генерации и отображения календаря. В ней нет дополнительных функций, таких как напоминания о событиях, интеграция с другими приложениями или возможность синхронизации с другими устройствами.

2. Отсутствие сохранения данных: Программа не сохраняет данные о событиях или заметках пользователя между сеансами работы. При закрытии программы все введенные данные будут потеряны.

3. Ограниченность визуального оформления: Внешний вид календаря и интерфейса программы ограничен предоставленными элементами и стилями. Нет возможности полностью настроить внешний вид календаря в соответствии с предпочтениями пользователя.

4. Отсутствие автоматических обновлений праздников: В программе заданы определенные праздничные дни.

5. Зависимость от компьютера: Программа требует наличия компьютера и запуска в операционной системе Windows, что ограничивает ее доступность и удобство использования в сравнении с физическими календарями и запуска в других операционных системах как Linux, MacOs.

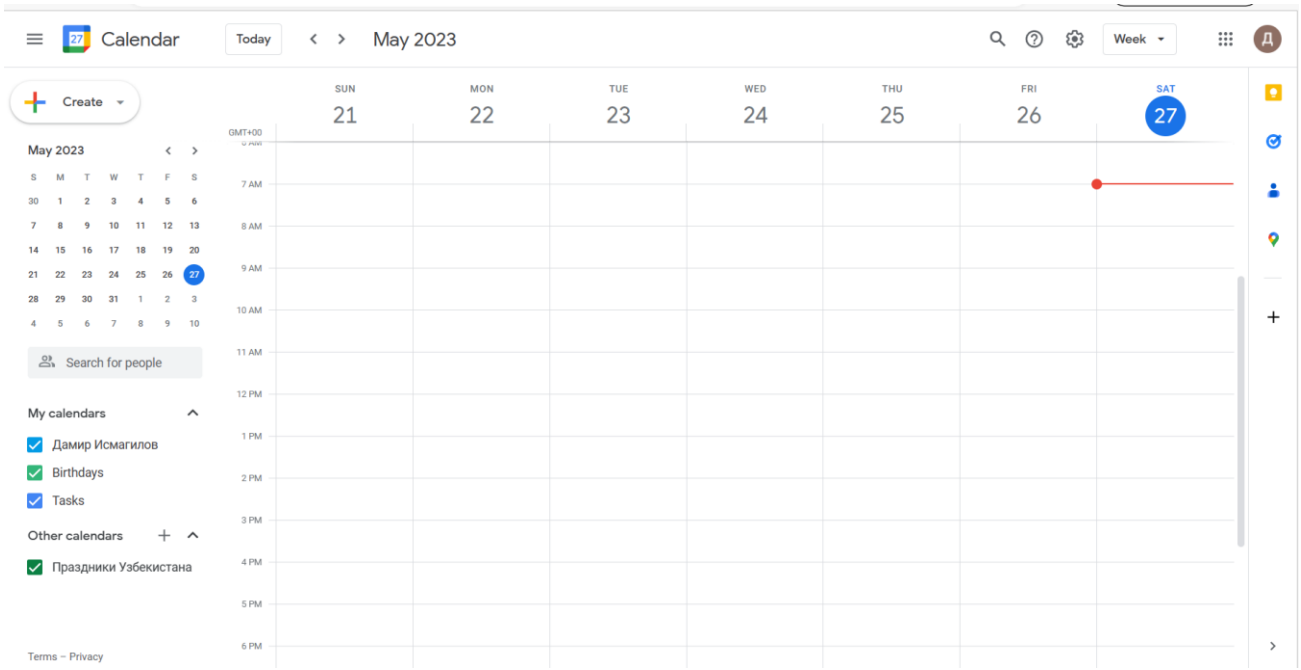
6. Отсутствие физической формы: Программа не имеет физического наличия в виде бумажных страниц, что может быть неудобно для тех, кто предпочитает традиционные бумажные календари или нуждается в их использовании в офлайн-режиме.

7. Требования к навыкам использования компьютера: для использования программы пользователю может потребоваться базовое понимание работы с компьютером и интерфейсом программы, что может быть проблематично для тех, кто не имеет опыта в использовании компьютера

1.4. Анализ рынка электронных календарей

Существует множество программных решений для генерации и отображения календарей, включая как простые приложения для индивидуального использования, так и более сложные системы планирования и управления событиями для организаций.

1) Google Calendar



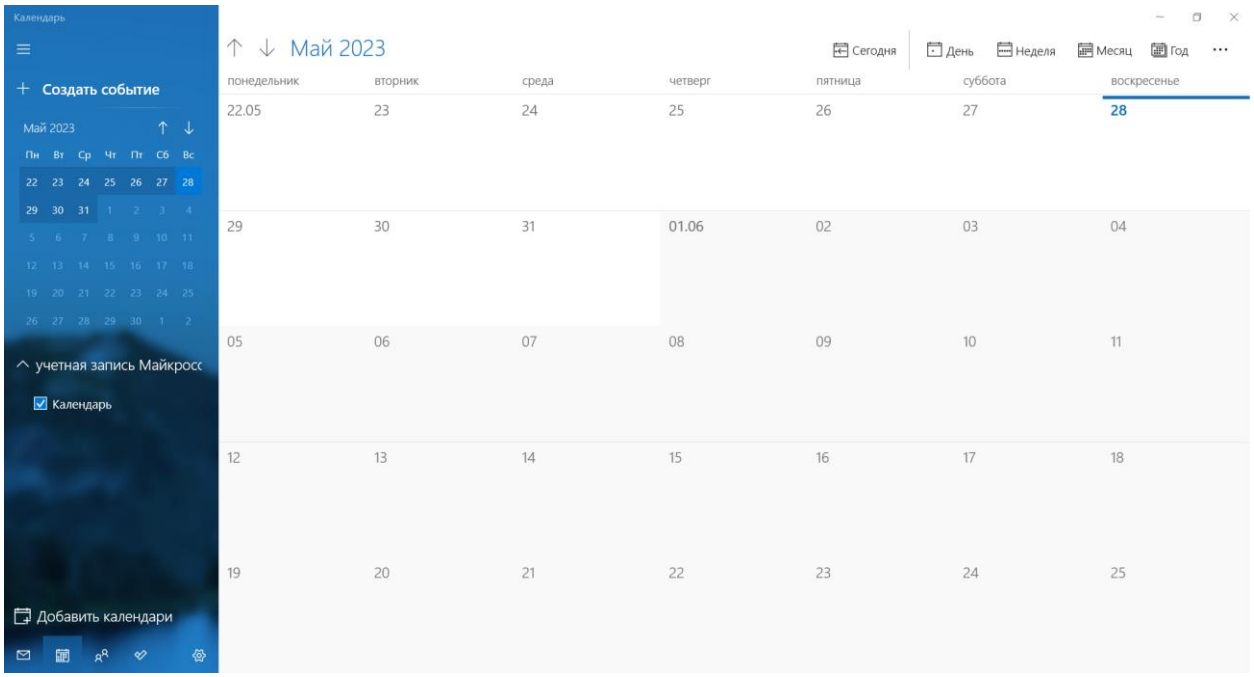
Google Календарь является одним из самых популярных онлайн-календарей, предлагаемых Google. Он обладает множеством функций, таких как создание событий, приглашение других пользователей, напоминания и синхронизация с другими устройствами.

Программа, которая будет разработана, будет ограничена локальным использованием и не доступна через Интернет. Google Календарь, с другой стороны, является веб-приложением, доступным через веб-браузер на различных платформах, таких как компьютеры, смартфоны и планшеты. Он предоставляет доступ к календарю через Интернет и позволяет синхронизировать данные между устройствами.

Преимущество Google Календаря заключается в его интеграции с другими

сервисами Google, такими как Gmail, и возможности синхронизации с мобильными устройствами.

2) Календарь Windows

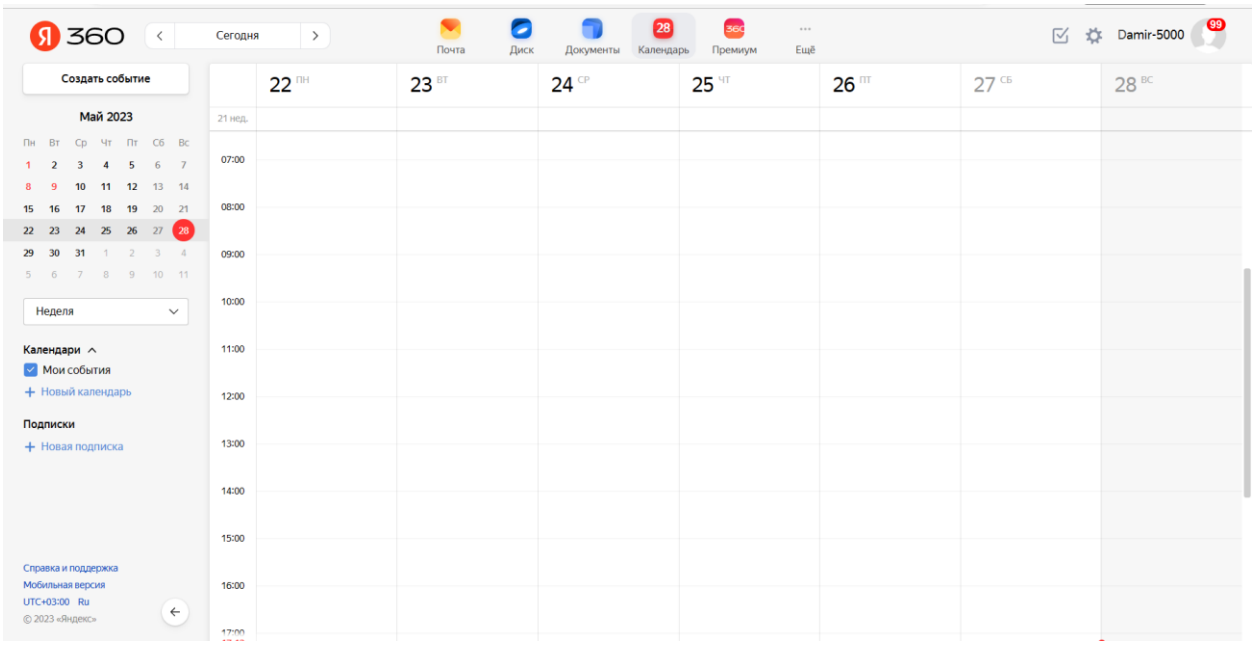


Календарь Windows является предустановленным приложением в операционных системах Windows 10 и Windows 11. Он предоставляет удобный способ для генерации, управления и просмотра календарей прямо на компьютере.

Также предоставляет функции генерации и управления календарями, включая создание событий, напоминаний, приглашение участников и синхронизацию с учетной записью Microsoft. Преимущество Календаря Windows состоит в его интеграции с операционной системой и простоте использования.

Календарь Windows предоставляет удобный и легко доступный способ управления и просмотра календарей на компьютере под управлением Windows. Он может быть полезным для пользователей, которым требуется базовая функциональность календаря с интеграцией с учетной записью Microsoft и другими приложениями Windows.

3) Yandex календарь



Яндекс.Календарь (Yandex.Calendar) -это онлайн-сервис календаря, предоставляемый Яндексом. Он предназначен для создания, управления и синхронизации событий и задач в удобном и интуитивно понятном интерфейсе.

Яндекс.Календарь предоставляет удобные инструменты для управления событиями и задачами. Он особенно полезен для пользователей, которые уже используют другие сервисы Яндекса и хотят интегрировать календарь с ними.

Преимущества программы относительно аналогов:

- Простота и легкость использования: Программа, описанная в коде, имеет простой и интуитивно понятный интерфейс, что делает ее легко доступной для пользователей без опыта работы с другими программами.
- Отсутствие необходимости в интернет-соединении: В отличие от некоторых онлайн-сервисов, программа может работать офлайн, что позволяет пользователю использовать ее без доступа в Интернет.
- Гибкость и удобство: Программа позволяет пользователю генерировать календари для любого года и месяца, что обеспечивает гибкость при планировании событий и просмотре дат. Она также позволяет быстро

переключаться между месяцами с помощью соответствующих кнопок.

- Возможность настройки стиля и цветов: В коде программы можно легко настроить стиль и цвета элементов календаря, включая выходные дни, праздники и дни месяца. Это позволяет адаптировать календарь под индивидуальные предпочтения пользователя.

1.5. Польза программы для широкого круга пользователей

Данная программа календаря может быть полезной для широкого круга пользователей, включая:

1. Людей, нуждающихся в организации своего расписания: Программа поможет им вести записи о встречах, событиях, важных сроках и напоминаниях.

2. Людей, работающих в офисе: они могут использовать программу для планирования рабочих встреч, совещаний, задач и проектов, а также для отслеживания своего рабочего графика.

3. Школьников и студентов: они смогут использовать программу для планирования учебных занятий, сроков сдачи заданий и проектов, а также для установления напоминаний о важных академических событиях, таких как экзамены или контрольные работы.

4. Родителей: они могут использовать программу для организации расписания своих детей, включая занятия, спортивные тренировки, встречи с друзьями и другие важные события.

5. Людей, ведущих активный образ жизни: Такие люди могут использовать программу для планирования и отслеживания своих хобби, спортивных мероприятий, путешествий и других активностей.

В конечном счете, программа имеет простой и интуитивно понятный интерфейс, который делает ее удобной в использовании для разных категорий

пользователей. Также стоит отметить, что программа предлагает функции генерации календаря с помощью введенных пользователем данных, а также отображение месяца, дней недели и дней месяца с различными цветами, отражающими выходные и праздничные дни. Это позволяет пользователю легко и наглядно визуализировать свое расписание и быть в курсе важных событий.

В следующей главе работы будет написаны цели и задачи, которые нужны для разработки программы, использование технологии, описана конкретная реализация программы для генерации и отображения календаря с использованием библиотеки и тестирование.

ГЛАВА 2: ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Цели и задачи разработки программы

Задачей данной индивидуальной работы является разработка программы для генерации и отображения календарей с использованием библиотеки Tkinter и будучи написанным на высокоуровневом языке программирования Python. Программа должна предоставлять пользователю возможность ввода года и месяца, после чего генерировать календарь для указанного периода. Интерфейс строен максимально понятным и удобным способом чтобы у пользователя не возникал никаких вопросов. Календарь должен содержать информацию о днях недели и датах, а также отображать праздничные дни, выделенные зеленым цветом, а выходные дни розовым цветом. Помимо этого, календарь должен отображать месяц над календарем.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработка графического интерфейса: необходимо создать окно приложения с элементами интерфейса, такими как поля ввода для года и месяца, кнопки для генерации календаря и переключения между месяцами. Графический интерфейс должен быть интуитивно понятным и удобным в использовании.
2. Обработка пользовательского ввода: Программа должна корректно обрабатывать введенные пользователем значения года и месяца. Необходимо проверять корректность ввода и предупреждать пользователя в случае ошибочных данных.
3. Генерация календаря: По введенным пользователем значениям года и месяца необходимо сгенерировать календарь для указанного периода. Календарь должен быть представлен в виде сетки, содержащей информацию о днях недели и датах.

4. Отображение календаря: Сгенерированный календарь должен быть отображен в графическом интерфейсе. Дни недели и даты должны быть отформатированы и расположены в соответствии с календарной сеткой. Праздничные дни должны быть выделены зеленым цветом.

5. Маркировка праздничных дней: В календаре необходимо отметить праздничные дни, определенные заранее. Для этого нужно создать список праздничных дней, который будет проверяться при генерации календаря. Праздничные дни должны быть выделены зеленым цветом.

6. Навигация по месяцам: Пользователь должен иметь возможность переключаться между месяцами, как вперед, так и назад. Для этого необходимо реализовать функционал кнопок "Previous Month" и "Next Month", которые будут обновлять отображение календаря с учетом выбранного месяца и года.

7. Обработка выходных дней: помимо праздничных дней, необходимо учесть выходные дни, которые приходятся на субботу и воскресенье. В программе нужно определить выходные дни и выделить их розовым цветом или стилем, чтобы пользователь мог легко их распознать.

В результате выполнения поставленных задач будет разработана программа, позволяющая генерировать календари с отображением дней недели и дат, а также выделением праздничных дней. Пользователь сможет легко переключаться по месяцам и получать актуальную информацию о календарных данных для выбранного периода.

2.2. Требования к программе

1) Функциональные требования

1. Отображение главного окна программа;
2. Наличие функционала кнопок;
3. Отображение календаря;
4. Возможность перехода на следующий или предыдущий месяц.

2) Требования к удобству и простоте использования

1. Интерфейс должен иметь простую структуру;
2. Не должно занимать много времени для генерации календаря;

3) Требование к производительности

Не должны быть задержки при генерации календаря и использования кнопок при переключении месяца;

4) Ограничения проектирования

1. Операционные системы Windows 7, Windows 8, Windows 8.1, Windows 10, Windows 11;
2. Среда разработки PyCharm;
3. Язык программирования Python;
4. Библиотека: Tkinter
5. Память: 8.39 МБ

2.3. Применение технологии: PyCharm, Python и Tkinter

PyCharm Community - это бесплатная интегрированная среда разработки (IDE) для языка программирования Python. Разработанная компанией JetBrains, она предлагает множество функций и инструментов, которые помогают разработчикам создавать, отлаживать и управлять проектами на Python.

PyCharm Community обладает удобным и интуитивно понятным пользовательским интерфейсом, что делает его доступным даже для новичков в программировании. В IDE доступны различные режимы просмотра и редактирования кода, включая режимы редактирования, отображения дерева проекта и поиска файлов.

Одной из ключевых особенностей PyCharm Community является его способность предоставлять автодополнение кода, что значительно увеличивает производительность разработчиков. IDE также обладает мощным отладчиком, который позволяет обнаруживать и исправлять ошибки в коде.

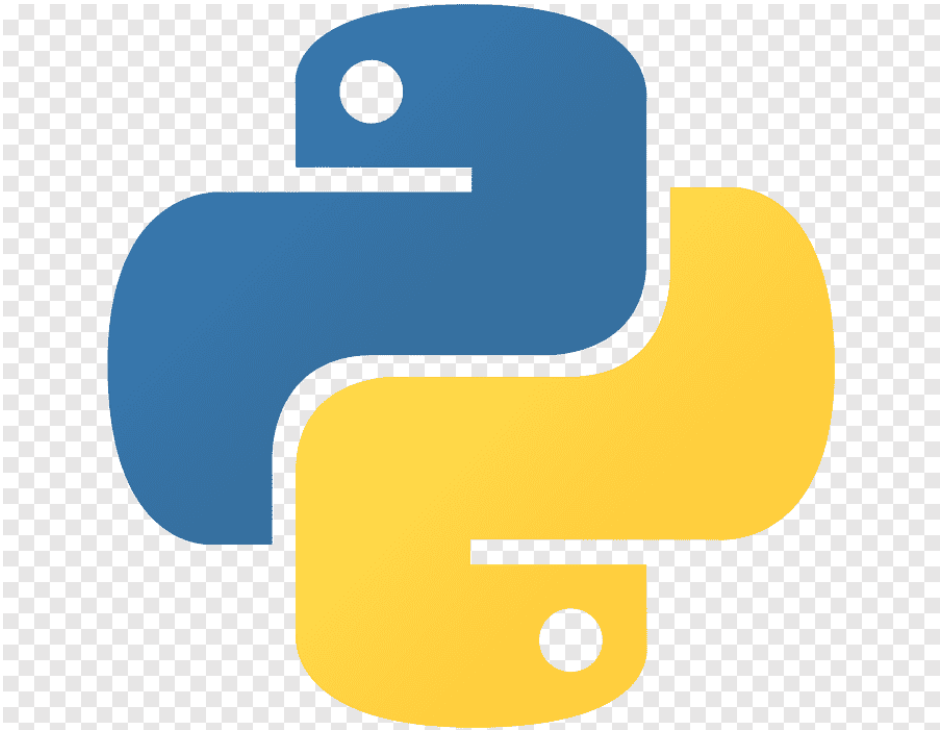
PyCharm Community предлагает интеграцию с системами контроля

версий, такими как Git, что облегчает управление и отслеживание изменений в проектах. Он также обладает возможностями управления пакетами и виртуальными средами Python, что позволяет легко устанавливать и управлять зависимостями проекта.

Кроме того, PyCharm Community предлагает инструменты для тестирования кода, анализа производительности и профилирования приложений Python. Он также поддерживает интеграцию с различными фреймворками и библиотеками, такими как Django, Flask, NumPy, Tkinter и другими.

Благодаря своей бесплатной лицензии, она доступна для всех и является популярным выбором среди разработчиков Python.

Python- это высокоуровневый, интерпретируемый язык программирования, который изначально был разработан Гвидо ван Россумом в конце 1980-х годов. Он отличается простым и понятным синтаксисом, который облегчает чтение и написание кода. Python является интерпретируемым языком, что означает, что программы на Python выполняются построчно, без необходимости предварительной компиляции.

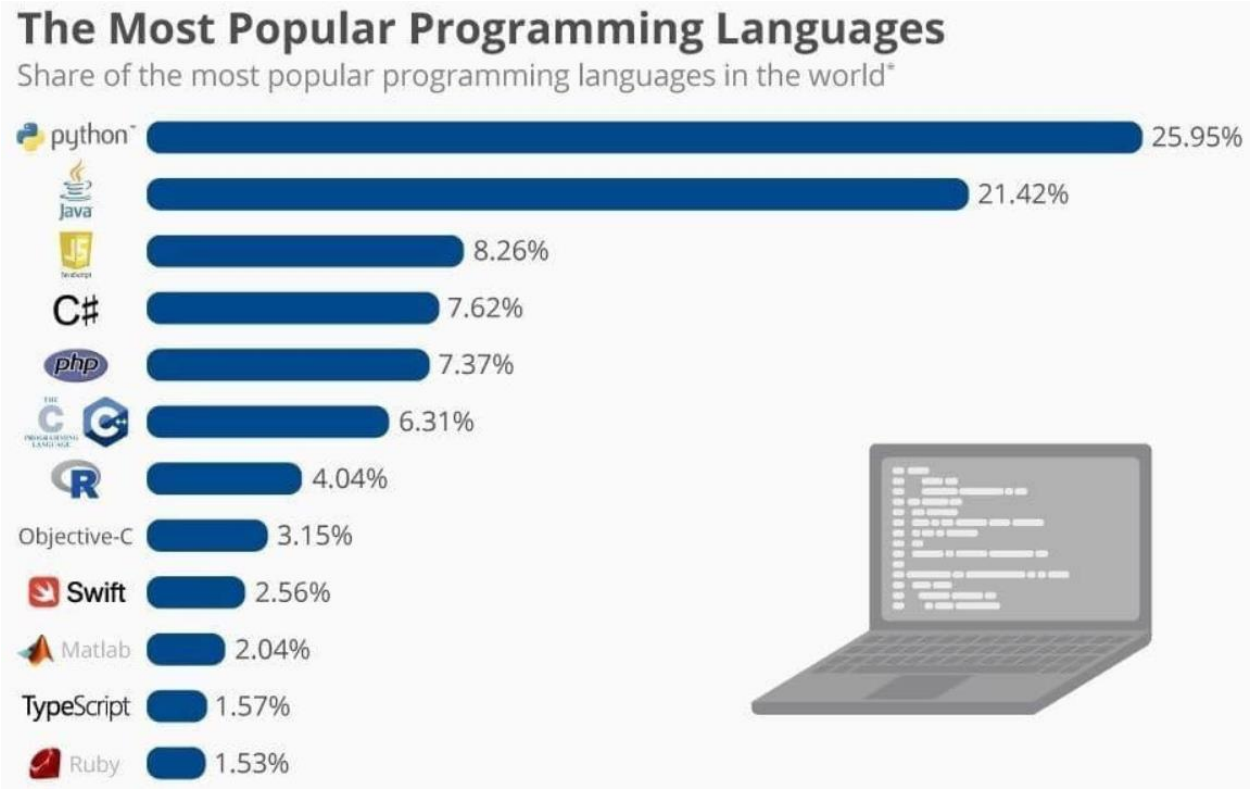


Python является многоцелевым языком программирования, который поддерживает различные стили программирования, включая процедурное, объектно-ориентированное и функциональное программирование. Он предлагает обширную стандартную библиотеку, которая включает в себя различные модули и инструменты, упрощающие разработку приложений.

Одной из главных особенностей Python является его философия "читаемости кода". Синтаксис языка ставит акцент на понятности и лаконичности, что делает код на Python легко читаемым и понятным. Это способствует улучшению сотрудничества между разработчиками и ускоряет процесс разработки.

Python обладает обширным набором сторонних библиотек и фреймворков, которые расширяют его возможности и упрощают разработку

приложений в различных областях, таких как веб-разработка, научные вычисления, анализ данных, искусственный интеллект и другие.



Основные преимущества Python включают простоту и читаемость кода, быструю разработку, мощные возможности работы с данными, поддержку многих платформ и операционных систем, а также активное сообщество разработчиков, которое постоянно развивает язык и предлагает новые инструменты и библиотеки.

Python широко используется в различных областях, включая веб-разработку, научные исследования, разработку игр, автоматизацию задач, системное администрирование и многое другое. Благодаря своей простоте и мощности, Python стал одним из самых популярных языков программирования в сообществе разработчиков.

Tkinter - это стандартная библиотека для создания графического пользовательского интерфейса (GUI) в языке программирования Python. Она предоставляет различные виджеты, инструменты и методы, позволяющие разработчикам создавать интерактивные приложения с помощью графического интерфейса.

Основным компонентом Tkinter является библиотека Tk, которая представляет собой графический фреймворк, написанный на языке программирования Tcl. Tkinter обеспечивает связь между языком Python и библиотекой Tk, позволяя разработчикам создавать и управлять различными элементами GUI, такими как окна, кнопки, метки, поля ввода и многое другое.

С использованием Tkinter можно создавать окна и размещать в них виджеты для взаимодействия с пользователем. Библиотека предоставляет различные методы для настройки внешнего вида и поведения виджетов, а также для обработки событий, таких как нажатие кнопки или ввод текста.

Tkinter также поддерживает использование различных компоновщиков (layout managers), которые позволяют управлять расположением виджетов в

окне. Некоторые из наиболее популярных компоновщиков в Tkinter включают grid, pack и place.

Одним из главных преимуществ Tkinter является его наличие в стандартной библиотеке Python, что означает, что он доступен по умолчанию без необходимости установки дополнительных пакетов. Это упрощает разработку и распространение приложений, так как не требуется дополнительных зависимостей.

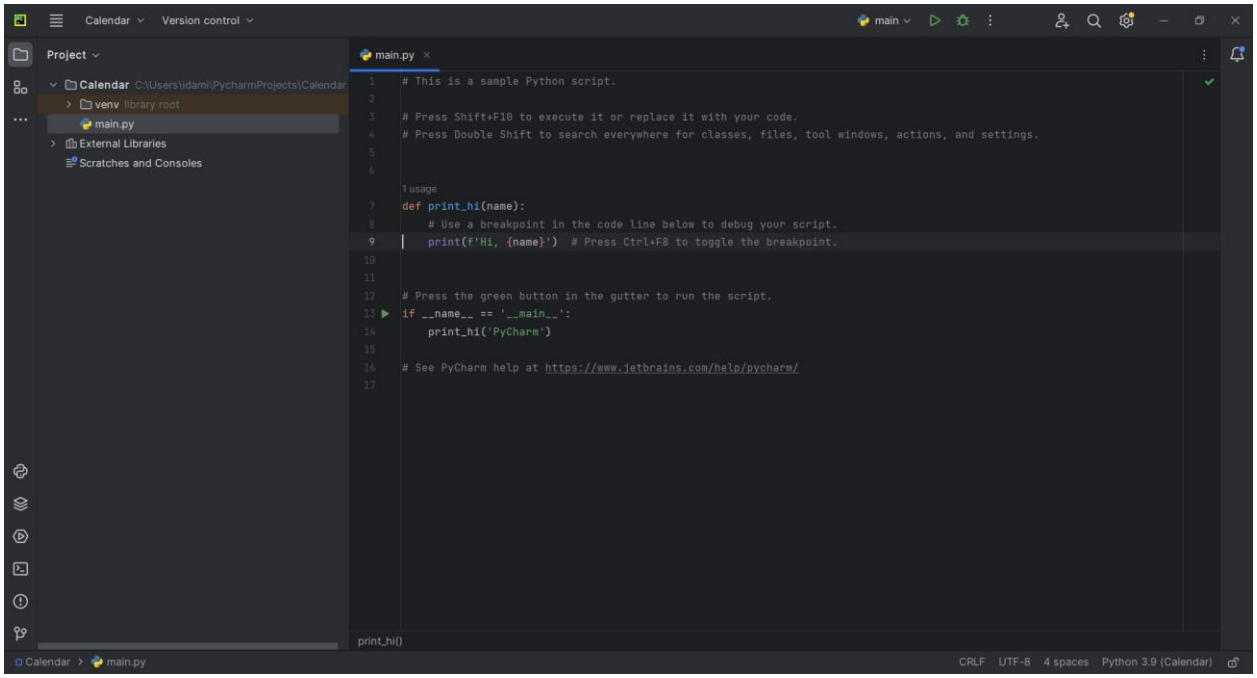
Tkinter также предлагает простой и интуитивно понятный интерфейс для создания GUI-приложений. Он поддерживает различные стили оформления, позволяя разработчикам создавать приложения с уникальным внешним видом и ощущением.

Хотя Tkinter может быть использован для создания простых интерфейсов, он также способен обрабатывать более сложные задачи и предоставляет разработчикам широкий набор возможностей для создания функциональных приложений с отзывчивым пользовательским интерфейсом.

Tkinter является популярным выбором для разработки GUI-приложений на Python благодаря своей простоте

2.3. **Начало разработки. Подготовка интегрированной среды (PyCharm) для разработки программы**

Весь процесс разработки программы начинается в интегрированной среде разработки PyCharm Community. Открываем программу и в появившемся окне пишем название проекта, для моего проекта будет название “Calendar”, указываем папку, где будет храниться проект и выбираем «Создать проект». После этого программа создаст нам базовый шаблон программы типа «Пустой проект» в котором уже содержится готовых исходный код, который после компиляции выводит “Hi, PyCharm”. Нам это не нужно, так как наша цель разработать генератор календаря, а не программу, которая выводит слово. Следовательно, в этой IDE и будем создавать программу.

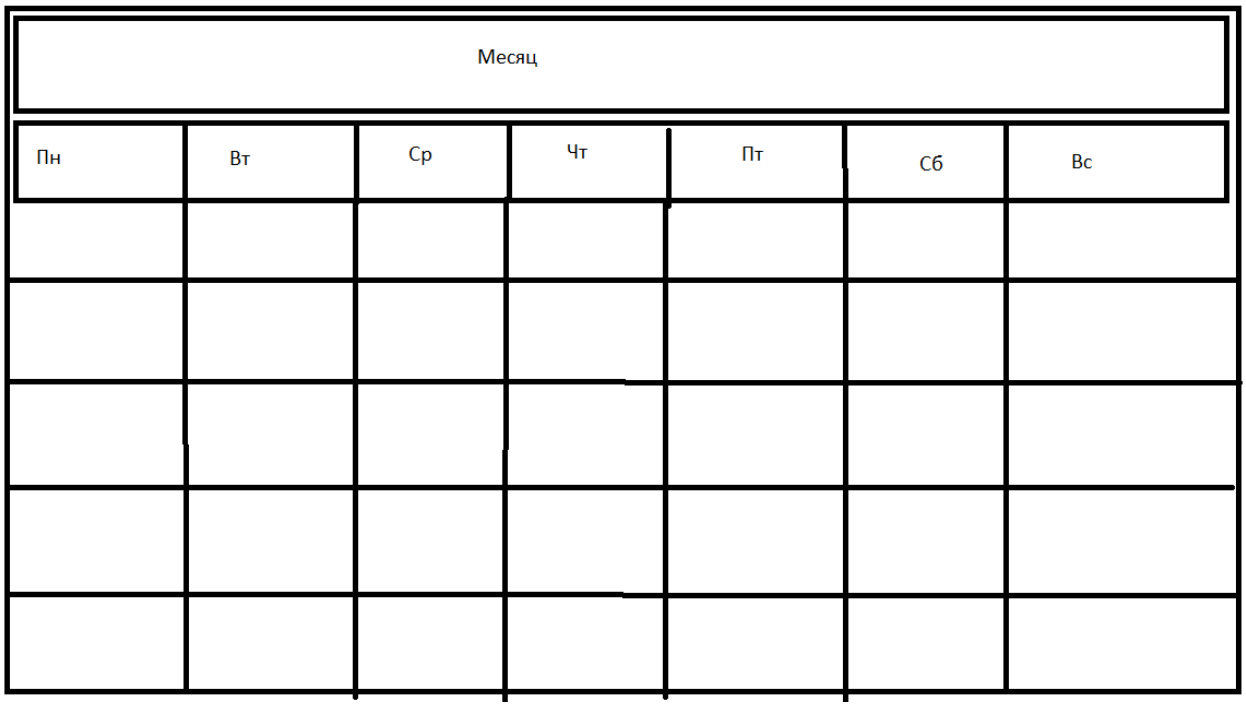


В этом то и удобства использование PyCharm: данный IDE позволяет сэкономить время и автоматически создает базовый шаблон, на ручное создание которого уходило бы более 45 минут без помощи таких программ.

2.4. Создание пользовательского интерфейса

Пользовательский интерфейс создается с использованием библиотеки Tkinter с помощью языка программирования Python в среде PyCharm

Эскиз главного экрана



Эскиз календаря (отображение чисел будет генерироваться в зависимости от

выбранного месяца)

1. Главное окно:

- Заголовок окна установлен как "Генератор календаря".

- Размер окна установлен на 1400x600+0+0 пикселей, где 1400 ширина, 600 длина. “0+0” заданы для того, чтобы программа запускалась сверху, а не в середине окна Windows.

- Цвет фона окна задан как "#99CCCC" (светло-серый цвет).

```
# Создаем главное окно
root = tk.Tk()
root.title("Генератор календаря")
root.geometry('1400x600+0+0')
root.configure(bg="#99CCCC")
```

2. Элементы интерфейса:

- `year_label`: Метка "Год:" для указания пользователю, что нужно ввести год.

  - Текст “Год: ”.

  - Шрифт и размер текста: “Arial”, 12.

  - Цвет установлен как у главного экрана.

Метод pack() используется для упаковки (размещения) виджета в главном окне программе.

```
# Создаем элементы интерфейса
year_label = tk.Label(root, text="Год:", font=("Arial", 12),
bg="#99CCCC")
year_label.pack()
```

- `year_entry`: Поле ввода для ввода года.

  - Шрифт и размер вводимого текста: “Arial”, 12.

  - Ширина ячейки: 45.

```
year_entry = tk.Entry(root, font=("Arial", 12), width=45)
year_entry.pack()
```

- `month_label`: Метка "Месяц:" для указания пользователю, что нужно ввести месяц.

  - Текст “ Месяц: ”.

  - Шрифт и размер текста: “Arial”, 12.

  - Цвет установлен как у главного экрана.

```
month_label = tk.Label(root, text="Месяц:", font=("Arial", 12),
bg="#99CCCC")
month_label.pack()
```

- `month_entry`: Поле ввода для ввода месяца.

  - Шрифт и размер вводимого текста: “Arial”, 12.

  - Ширина ячейки: 45.

```
month_entry = tk.Entry(root, font=("Arial", 12), width=45)
month_entry.pack()
```

- `generate_button`: Кнопка "Сгенерировать календарь" для запуска генерации календаря на основе введенных значений года и месяца.

  - Текст “ Сгенерировать календарь: ”.

  - Шрифт и размер текста: “Arial”, 12.

  - Цвет кнопки: Голубой.

  - Метод: “generate_calendar”. Будет использоваться для генерации календаря на основе введенных пользователем значений года и месяца.

  - Ширина и высота кнопки меняются в зависимости от размера экрана

  - Установления кнопки влево

```
generate_button = tk.Button(controls_frame, text="Сгенерировать
календарь", font=("Arial", 12), bg='blue',
command=generate_calendar, width=width_button,
height=height_button)
generate_button.pack(side=tk.LEFT)
```

- `prev_button`: Кнопка "Предыдущий месяц" для перехода к предыдущему месяцу от текущей выбранной даты.

- Текст “ Предыдущий месяц: ”.

- Шрифт и размер текста: “Arial”, 12.

- Цвет кнопки: Зеленый.

- Метод: “previous_mohth”. Будет использоваться для перехода к предыдущему месяцу отображаемого календаря.

- Ширина и высота кнопки меняются в зависимости от размера экрана

- Установления кнопки влево

```
prev_button = tk.Button(controls_frame, text="Предыдущий месяц",
font=("Arial", 12), bg='green', command=previous_month,
width=width_button, height=height_button)
prev_button.pack(side=tk.LEFT)
```

- `next_button`: Кнопка "Следующий месяц" для перехода к следующему месяцу от текущей выбранной даты.

- Текст “ Следующий месяц : ”.

- Шрифт и размер текста: “Arial”, 12.

- Цвет кнопки: Желтый.

- Метод: “next_month”. Будет использоваться для перехода к следующему месяцу отображаемого календаря.

- Ширина и высота кнопки меняются в зависимости от размера экрана

- Установления кнопки влево

```
next_button = tk.Button(controls_frame, text="Следующий месяц",
font=("Arial", 12), bg='yellow', command=next_month,
width=width_button, height=height_button)
next_button.pack(side=tk.LEFT)
```

3. Рамка календаря (`calendar_frame`):

- Фон установлен как белый.

- Толщина границы рамки задана как 2 пикселя.

- Стиль границы рамки установлен как "solid".

- Аргументы pady=50 и padx=50 указывают на количество пикселей отступа (padding) в вертикальном (pady) и горизонтальном (padx) направлениях соответственно.Таким образом, данная инструкция устанавливает отступы в размере 50 пикселей вверху и внизу (pady) и 50 пикселей слева и справа (padx) для фрейма calendar_frame.

```
calendar_frame = tk.Frame(root, bg="white", bd=2,
relief="solid")
calendar_frame.pack(pady=50, padx=50)
```

4. Отображение календаря:

- При нажатии на кнопку "Сгенерировать календарь" вызывается метод `generate_calendar()`.

- В методе `generate_calendar()` происходит проверка корректности введенных значений года и месяца.

- Если значения некорректны, выводится сообщение об ошибке.

- Предыдущий календарь, если был отображен, очищается.

- Затем в рамке `calendar_frame` отображается месяц, дни недели и дни месяца в соответствии с выбранным годом и месяцем.

- Дни недели отображаются в первом ряду рамки `calendar_frame`.

- Дни месяца отображаются в последующих рядах рамки `calendar_frame`.

- Дни месяца окрашиваются в разные цвета в зависимости от выходных дней и праздничных дней.

5. Навигация по месяцам:

- При нажатии на кнопку "Предыдущий месяц" вызывается функция `previous_month()`.

- В функции `previous_month()` происходит уменьшение значения текущего месяца на 1 и обновление полей ввода года и месяца.

- Если текущий месяц является январем, то значение года уменьшается на 1, а месяц устанавливается как декабрь.

- При нажатии на кнопку "Следующий месяц" вызывается функция `next_month()`.

- В функции `next_month()` происходит увеличение значения текущего месяца на 1 и обновление полей ввода года и месяца.

- Если текущий месяц является декабрем, то значение года увеличивается на 1, а месяц устанавливается как январь.

6. Равномерное распределение ячеек в рамке:

- Чтобы ячейки календаря заполняли рамку равномерно, применяется цикл `for` для настройки равномерного распределения ячеек по строкам и столбцам рамки `calendar_frame`. Метод grid() используется для размещения элементов в виде таблицы.

```
# Задаем равномерное распределение ячеек в рамке
rows = 8
columns = 7
for row in range(rows):
    calendar_frame.grid_rowconfigure(row, weight=1)
for col in range(columns):
    calendar_frame.grid_columnconfigure(col, weight=1)
```

7. Главный цикл обработки событий:

- После создания всех элементов интерфейса, запускается главный цикл обработки событий методом `root.mainloop()`.

```
# Запускаем главный цикл обработки событий
root.mainloop()
```

- Этот цикл позволяет пользователю взаимодействовать с интерфейсом и откликаться на события, такие как нажатие кнопок или ввод данных в поля.

2.5. Реализация методов

Реализация методов включает в себя следующие шаги:

1. Импорт (подключения) необходимых модулей для разработки программы:

- `calendar` - для работы с календарными данными.
- `tkinter` - для создания графического интерфейса.
- `messagebox` - для вывода сообщений об ошибках.

```
import calendar
import tkinter as tk
from tkinter import messagebox
```

2. Метод `generate_calendar()`, которая будет вызываться при нажатии кнопки "Generate Calendar". Эта функция будет отвечать за генерацию и отображение календаря на основе введенных пользователем значений года и месяца. Она выполняет следующие шаги:

- Получение значений года и месяца из соответствующих полей ввода “year_entry” и “month_entry”.
- Проверка корректности введенных данных (например, проверка на отрицательные значения и диапазон месяца). Если год отрицательный или месяц находится вне диапазона от 1 до 12, вывести сообщение об ошибке и прекратить выполнение метода.
- Генерация календаря с помощью функции `calendar.monthcalendar(year, month)` для указанного года и месяца и сохранить результат в переменную “cal_data”.
- Очистка предыдущего отображения календаря, если оно уже существует, удалив все виджеты внутри “calendar_frame”.
- Отображение дней недели в верхней части календаря.
- Отображение дней месяца с учетом цветового кодирования для выходных и праздничных дней.

- Проверка и отметка праздничных дней в соответствии с заданными праздниками.

- Отображение маркера для праздничных дней.

```
def generate_calendar():
    # Получаем значения даты
    year = int(year_entry.get())
    month = int(month_entry.get())

    # Проверяем корректность введенных данных
    if year < 0 or month < 1 or month > 12:
        messagebox.showerror("Ошибка", "Пожалуйста, введите
корректные значения года и месяца в числах.")
        return

    # Генерируем календарь
    cal_data = calendar.monthcalendar(year, month)

    # Очищаем предыдущий календарь, если он уже отображен
    for widget in calendar_frame.winfo_children():
        widget.destroy()

    # Отображаем  месяц
    month_names = ['Январь', 'Февраль', 'Март', 'Апрель', 'Май',
'Июнь', 'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь',
'Декабрь']
    month_label = tk.Label(calendar_frame,
text=month_names[month - 1], font=("Arial", 16, "bold"),
width=30, height=12)
    month_label.grid(row=0, column=0, columnspan=7, pady=10,
sticky="nsew")

    # Отображаем дни недели
    days = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Вс']
    for i, day in enumerate(days):
        label = tk.Label(calendar_frame, text=day,
bg="lightgray", borderwidth=1, fg="black", relief="solid",
font=("Arial", 10, "bold"), width=30, height=20)
        label.grid(row=1, column=i, sticky="nsew")

    # Отображаем дни месяца
    for week, week_data in enumerate(cal_data, start=2):
        for day, date in enumerate(week_data):
            if date == 0:
                continue
            # Определяем цвета для выходных дней
            if day >= 5:
                bg_color = "lightpink"
                fg_color = "black"
            else:
```

```
                bg_color = "white"
                fg_color = "black"

            # Проверяем и отмечаем праздничные дни
            is_holiday = False
            holidays = [(1, 1), (2, 1), (3, 1), (8, 3), (21, 3),
(9, 5), (1, 9), (1, 10), (8, 12)]  # Праздничные дни
            if (date, month) in holidays:
                bg_color = "lightgreen"
                fg_color = "black"
                is_holiday = True

            # Отображаем дни месяца с цветами и стилем
            label = tk.Label(calendar_frame, text=str(date),
bg=bg_color, fg=fg_color, relief="solid", bd=1, width=30,
height=20)
            label.grid(row=week, column=day, sticky="nsew")
```

Описание **метода`generate_calendar()`**

Вначале метод получает значения года и месяца из полей ввода `year_entry` и `month_entry` с помощью метода `get()` и преобразует их в целочисленный формат.

Затем происходит проверка корректности введенных данных: если год отрицательный или месяц находится вне диапазона от 1 до 12, то выводится сообщение об ошибке с помощью `messagebox.showerror()` и метод прекращает своё выполнение с помощью оператора `return`.

Если введенные данные корректны, то вызывается функция `calendar.monthcalendar(year, month)`, которая возвращает двумерный список `cal_data` с данными календаря для указанного года и месяца.

Далее происходит очистка предыдущего календаря, если он уже отображен, с помощью цикла `for widget in calendar_frame.winfo_children(): widget.destroy()`. `calendar_frame` будет отображать календарь.

Затем создается и отображается месяц с помощью виджета `Label` (`month_label`). Текст берется из списка `month_names`, где каждый элемент списка соответствует названию месяца. Размер и стиль шрифта задаются с помощью аргументов `font` в конструкторе `Label`, а ширина и высота - с помощью аргументов `width` и `height`.

Далее создается и отображается строка с днями недели (`days`). Каждый день недели отображается в виде виджета `Label` с соответствующим текстом и стилем.

Затем с помощью вложенных циклов происходит отображение дней месяца. Если значение `date` равно 0, то пропускается итерация, так как это пустая ячейка. Иначе определяются цвет фона и текста для ячейки в зависимости от того, является ли день выходным или праздничным. Если день является праздником, то его цвет меняется на "lightgreen". Затем создается и отображается ячейка (`label`) с указанными цветами и стилем.

3. Создание методов `previous_month()` и `next_month()`, которые будут вызываться при нажатии кнопок "Previous Month" и "Next Month" соответственно. Эти функции будут обновлять отображение календаря с учетом выбранного месяца и года. Они выполняют следующие шаги:

- Получение текущих значений года и месяца из соответствующих полей ввода.

- Изменение значений года и месяца в зависимости от выбранного действия (предыдущий месяц или следующий месяц).

- Обновление значений года и месяца в полях ввода.

- Вызов функции `generate_calendar()` для обновления отображения календаря.

```
def previous_month():
    current_year = int(year_entry.get())
    current_month = int(month_entry.get())

    if current_month == 1:
        year_entry.delete(0, tk.END)
        year_entry.insert(tk.END, str(current_year - 1))
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, "12")
    else:
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, str(current_month - 1))

    generate_calendar()
```

Алгоритм метода `previous_month()`:

1) Получить текущие значения года и месяца из полей ввода `year_entry` и `month_entry`.

2) Проверить, если текущий месяц равен 1 (январь).

- Если это так, значит нужно перейти к предыдущему году.

- Удалить текущее значение в поле `year_entry` с помощью метода `delete(0, tk.END)`.

- Вставить новое значение года (текущий год минус 1) в поле `year_entry` с помощью метода `insert(tk.END, str(current_year - 1))`.

- Удалить текущее значение в поле `month_entry` с помощью метода `delete(0, tk.END)`.

- Вставить значение "12" (декабрь) в поле `month_entry` с помощью метода `insert(tk.END, "12")`.

- В противном случае, значит можно перейти к предыдущему месяцу в текущем году.

- Удалить текущее значение в поле `month_entry` с помощью метода `delete(0, tk.END)`.

- Вставить новое значение месяца (текущий месяц минус 1) в поле `month_entry` с помощью метода `insert(tk.END, str(current_month - 1))`.

3) Вызвать метод `generate_calendar()` для обновления календаря с учетом нового года и месяца.

```
def next_month():
    current_year = int(year_entry.get())
    current_month = int(month_entry.get())

    if current_month == 12:
        year_entry.delete(0, tk.END)
        year_entry.insert(tk.END, str(current_year + 1))
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, "1")
    else:
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, str(current_month + 1))

    generate_calendar()
```

Алгоритм метода `next_month()`:

1) Получить текущие значения года и месяца из полей ввода `year_entry` и `month_entry`.

2) Проверить, если текущий месяц равен 12 (декабрь).

- Если это так, значит нужно перейти к следующему году.

- Удалить текущее значение в поле `year_entry` с помощью метода `delete(0, tk.END)`.

- Вставить новое значение года (текущий год плюс 1) в поле `year_entry` с помощью метода `insert(tk.END, str(current_year + 1))`.

- Удалить текущее значение в поле `month_entry` с помощью метода `delete(0, tk.END)`.

- Вставить значение "1" (январь) в поле `month_entry` с помощью метода `insert(tk.END, "1")`.

- В противном случае, значит можно перейти к следующему месяцу в текущем году.

- Удалить текущее значение в поле `month_entry` с помощью метода `delete(0, tk.END)`.

- Вставить новое значение месяца (текущий месяц плюс 1) в поле `month_entry` с помощью метода `insert(tk.END, str(current_month + 1))`.

3) Вызвать метод `generate_calendar()` для обновления календаря с учетом нового года и месяца.

2.6. Тестирование

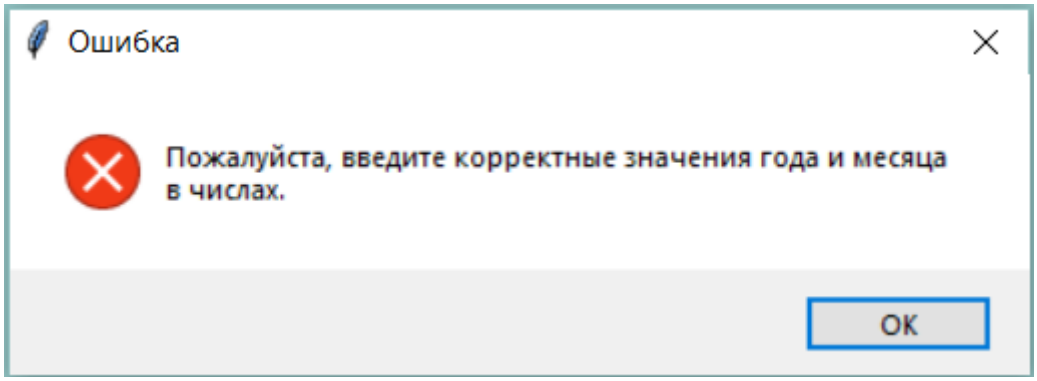
Тестирование программы может включать следующие шаги:

1. Ввод корректных значений года и месяца и нажатие кнопки "Generate Calendar" для генерации календаря.

После того, как нажали на кнопку на “Сгенерировать календарь”, то календарь отобразился правильно, соответствующие дни выходных и праздничные дни окрашены в соответствующие цвета, и маркеры для праздничных дней отображаются корректно.

2. Ввод некорректных значений года и месяца, таких как отрицательные значения или недопустимые значения месяца, и нажатие кнопки "Generate Calendar".

Если пользователь ввел неправильные значения, то программа выводит сообщение об ошибке и не позволяет сгенерировать некорректный календарь.



3. Нажатие кнопок "Предыдущий месяц" и "Следующий месяц" для переключения ме жду месяцами.

После нажатия кнопок "Предыдущий месяц" и "Следующий месяц", то отображение календаря обновляется правильно и соответствует выбранному месяцу.

4. Изменение значения года и/или месяца в полях ввода и нажатие кнопки "Generate Calendar".

Календарь обновляется с учетом новых значений года и месяца.

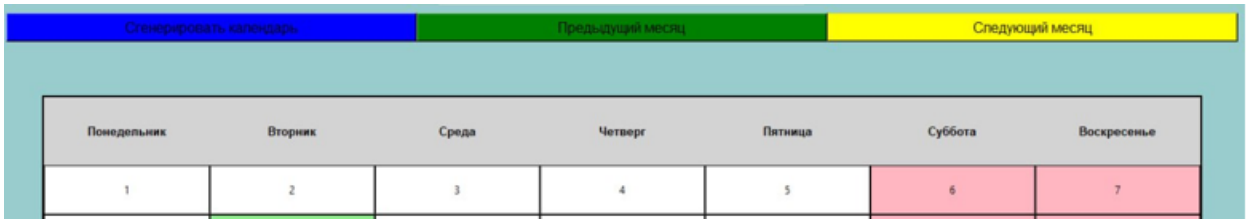
5. Проверка работы программы с различными праздниками и их датами.

Праздничные дни правильно отображаются и окрашиваются зеленым цветом.

<u>Ошибки, которые возникали при разработке программы:</u>

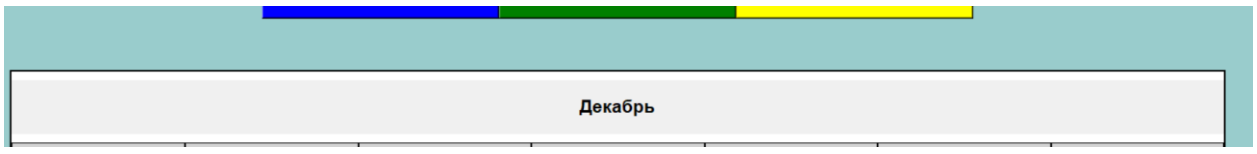
В ходе тестирования возникли следующие проблемы и их решения:

- **Проблема**: Некорректное отображение календаря. Если нажать на кнопку “Сгенерировать календарь”, то отображался календарь, но не показывал месяц.



- **Решение:** в метод “generate_calendar” добавили массив с месяцами, для того чтобы при нажатии на “Сгенерировать календарь отображался месяц”

```
# Отображаем  месяц
month_names = ['Январь', 'Февраль', 'Март', 'Апрель', 'Май',
'Июнь', 'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь',
'Декабрь']
month_label = tk.Label(calendar_frame, text=month_names[month -
1], font=("Arial", 16, "bold"), width=30, height=12)
month_label.grid(row=0, column=0, columnspan=7, pady=10,
sticky="nsew")
```



- **Проблема**: между неделями не проведена черта.

-Решение: в метод “generate_calendar” в цикле, который выводит недели, в свойствах Label добавили “relief=’solid’”

```
label = tk.Label(calendar_frame, text=day, bg="lightgray",
borderwidth=1, fg="black", relief="solid", font=("Arial", 10,
"bold"), width=30, height=20)
```

ЗАКЛЮЧЕНИЕ

В заключение индивидуального проекта по можно отметить следующее:

В ходе создания программы были рассмотрены основные этапы разработки, которая состоит из двух частей:

Теоретическая часть: анализ предметной области, отличие электронных календарей от бумажных, плюсы и минусы программы, анализ рынка электронных календарей, польза программы для широкого круга пользователей.

Практическая часть: Цели и задачи разработки программы, применение технологии: Python, Tkinter, Pycharm, подготовка к реализации, создание пользовательского интерфейса, реализация методов и тестирование.

Было проведен анализ существующих приложений-конкурентов.

В ходе реализации программы были использованы язык программирования Python и библиотека Tkinter для создания графического интерфейса и интегрированная среда разработки (IDE) PyCharm Community. Код программы был структурирован и организован в функции, обеспечивая модульность и повторное использование кода. Была реализована возможность генерации календаря для заданного года и месяца, а также навигация по месяцам.

В процессе тестирования программы были проведены различные тесты, включая ввод корректных и некорректных значений, переключение между месяцами, проверку отображения выходных и праздничных дней. Результаты тестирования показали, что программа работает корректно и соответствует ожиданиям, отображая календарь с правильными днями и цветами для выходных и праздничных дней. Помимо того, в ходе тестирования программы были исправлены некоторые недочеты, связанные с отображением календаря, такие как:

1) Календарь отображал дни, недели, но не месяцы;
2) Не была проведена черта, разделяющая недели;

Следовательно, в ходе анализа результатов тестирования были выявлены

и исправлены некоторые ошибки, что позволило улучшить качество и надежность программы. Программа успешно выполняет свою основную задачу - генерацию и отображение календарных данных - и предоставляет пользователям удобный инструмент для просмотра календаря на заданный период.

В целом, разработанная программа является эффективным и полезным инструментом, который может быть использован в различных сферах, требующих работы с календарными данными. Программа представляет практическую и полезную утилиту, а ее реализация отражает навыки и знания, полученные в процессе обучения и обладает потенциалом для дальнейшего расширения и улучшения, добавления новых функций и возможностей с целью удовлетворения потребностей пользователей.

В будущем, можно улучшить программу, добавив новые функции, например, функцию добавление задач, событие и функцию напоминание (Task Manager), также можно расширить функционал путем добавления списков покупок и т.д.

В процессе создания этой программы, я приобрел новые навыки в разработке графических интерфейсов, использовании библиотеки Tkinter и организации кода. Проект также позволил применить знания в области работы с датами и календарями, а также основные принципы программирования и тестирования. Уверен, что в будущем все знания который я получил в ходе выполнения индивидуального проекта будут вручать меня и помогать в моей карьере.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. "Python GUI Programming with Tkinter" by Alan D. Moore
2. "Python Cookbook" by David Beazley and Brian K. Jones
3. "Python Crash Course" by Eric Matthes
4. Modern Tkinter for busy python developer. Mark Roseman
5. Изучаем программирование на Python. Бэрри Пол
6. Простой Python. Современный стиль программирования. 2-е изд. Любанович Б
7. Программирование на Python в примерах и задачах. Васильев А.Н.
8. https://metanit.com/python/tutorial/?ysclid=lia3nyoe5b827436672
9. https://metanit.com/python/tkinter/
10. https://stepik.org/course/58852/promo
11. http://www.tkdocs.com/index.html
12. http://younglinux.info/book/export/html/48
13. https://pythonpip.ru/osnovy/uchebnik-po-tkinter-v-python-rukovodstvo-i-primery?ysclid=lia4n0ept652139830
14. https://pythonru.com/?ysclid=lia4utr4jc490915536
15. https://python.ru/?ysclid=lia4uoz4p793176472
16. https://www.wikipedia.org/
17. https://ziyonet.uz/
18. https://calendar.google.com/calendar/u/0/r
19. https://calendar.yandex.ru/week?uid=780191798
20. https://www.jetbrains.com/?ysclid=lia51iugg8724663075
21. https://ottisk.com/news/electronic-vs-print-calendars/?ysclid=lia51bvfds279416093
22. https://nabiraem.ru/blogs/job/53266/?ysclid=lia53f7d8u359320642
23. https://docs.python.org
24. https://docs.python.org/3/library/tkinter.html
25. https://studfile.net/

ПРИЛОЖЕНИЕ

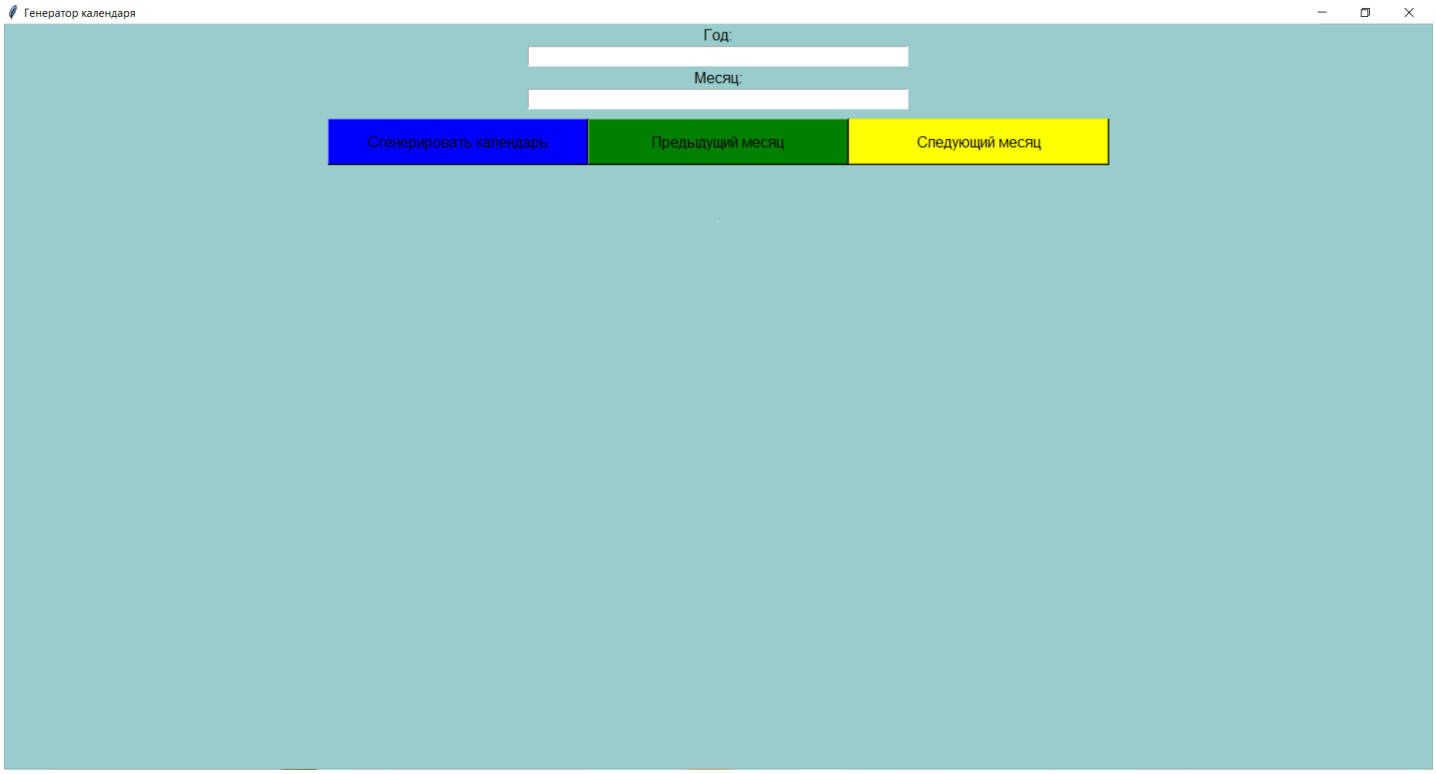
Код программы:

```
import calendar
import tkinter as tk
from tkinter import messagebox

def generate_calendar():
    # Получаем значения даты
    year = int(year_entry.get())
    month = int(month_entry.get())

    # Проверяем корректность введенных данных
    if year < 0 or month < 1 or month > 12:
        messagebox.showerror("Ошибка", "Пожалуйста, введите
корректные значения года и месяца в числах.")
        return

    # Генерируем календарь
    cal_data = calendar.monthcalendar(year, month)

    # Очищаем предыдущий календарь, если он уже отображен
    for widget in calendar_frame.winfo_children():
        widget.destroy()

    # Отображаем  месяц
    month_names = ['Январь', 'Февраль', 'Март', 'Апрель', 'Май',
'Июнь', 'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь',
'Декабрь']
    month_label = tk.Label(calendar_frame,
text=month_names[month - 1], font=("Arial", 16, "bold"),
width=30, height=12)
    month_label.grid(row=0, column=0, columnspan=7, pady=10,
sticky="nsew")

    # Отображаем дни недели
    days = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Вс']
    for i, day in enumerate(days):
        label = tk.Label(calendar_frame, text=day,
bg="lightgray", borderwidth=1, fg="black", relief="solid",
font=("Arial", 10, "bold"), width=30, height=20)
        label.grid(row=1, column=i, sticky="nsew")

    # Отображаем дни месяца
    for week, week_data in enumerate(cal_data, start=2):
        for day, date in enumerate(week_data):
            if date == 0:
                continue
            # Определяем цвета для выходных дней
            if day >= 5:
                bg_color = "lightpink"
                fg_color = "black"
```

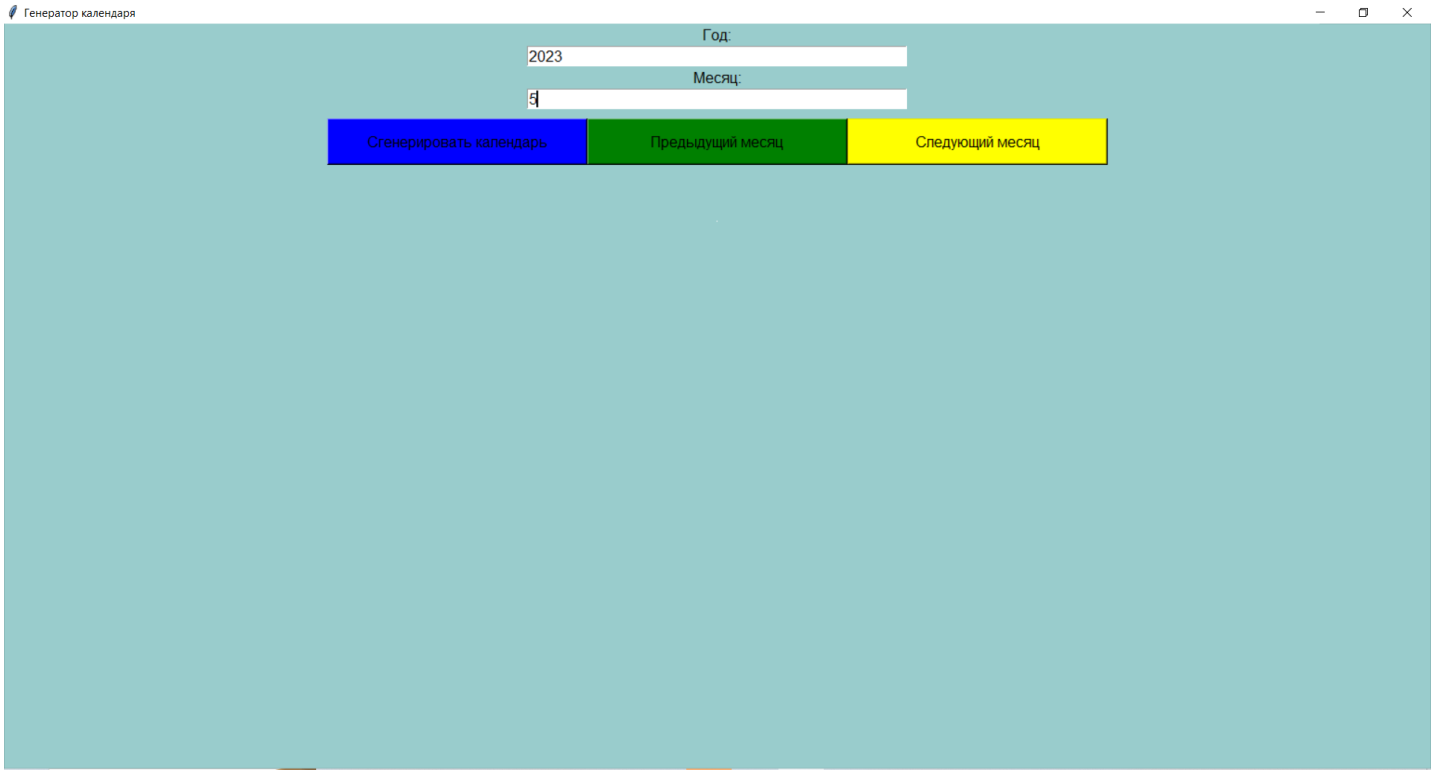
```
            else:
                bg_color = "white"
                fg_color = "black"

            # Проверяем и отмечаем праздничные дни
            is_holiday = False
            holidays = [(1, 1), (2, 1), (3, 1), (8, 3), (21, 3),
(9, 5), (1, 9), (1, 10), (8, 12)]  # Праздничные дни
            if (date, month) in holidays:
                bg_color = "lightgreen"
                fg_color = "black"
                is_holiday = True

            # Отображаем дни месяца с цветами и стилем
            label = tk.Label(calendar_frame, text=str(date),
bg=bg_color, fg=fg_color, relief="solid", bd=1, width=30,
height=20)
            label.grid(row=week, column=day, sticky="nsew")

def previous_month():
    current_year = int(year_entry.get())
    current_month = int(month_entry.get())

    if current_month == 1:
        year_entry.delete(0, tk.END)
        year_entry.insert(tk.END, str(current_year - 1))
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, "12")
    else:
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, str(current_month - 1))

    generate_calendar()

def next_month():
    current_year = int(year_entry.get())
    current_month = int(month_entry.get())

    if current_month == 12:
        year_entry.delete(0, tk.END)
        year_entry.insert(tk.END, str(current_year + 1))
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, "1")
    else:
        month_entry.delete(0, tk.END)
        month_entry.insert(tk.END, str(current_month + 1))

    generate_calendar()

# Создаем главное окно
root = tk.Tk()
root.title("Генератор календаря")
root.geometry('1400x600+0+0')
```

```
root.configure(bg="#99CCCC")

# Создаем элементы интерфейса
year_label = tk.Label(root, text="Год:", font=("Arial", 12),
bg="#99CCCC")
year_label.pack()

year_entry = tk.Entry(root, font=("Arial", 12), width=45)
year_entry.pack()

month_label = tk.Label(root, text="Месяц:", font=("Arial", 12),
bg="#99CCCC")
month_label.pack()

month_entry = tk.Entry(root, font=("Arial", 12), width=45)
month_entry.pack()

controls_frame = tk.Frame(root)
controls_frame.pack(pady=10)

width_button = 30
height_button = 2
generate_button = tk.Button(controls_frame, text="Сгенерировать
календарь", font=("Arial", 12), bg='blue',
command=generate_calendar, width=width_button,
height=height_button)
generate_button.pack(side=tk.LEFT)

prev_button = tk.Button(controls_frame, text="Предыдущий месяц",
font=("Arial", 12), bg='green', command=previous_month,
width=width_button, height=height_button)
prev_button.pack(side=tk.LEFT)

next_button = tk.Button(controls_frame, text="Следующий месяц",
font=("Arial", 12), bg='yellow', command=next_month,
width=width_button, height=height_button)
next_button.pack(side=tk.LEFT)

calendar_frame = tk.Frame(root, bg="white", bd=2,
relief="solid")
calendar_frame.pack(pady=50, padx=50)

# Задаем равномерное распределение ячеек в рамке
rows = 8
columns = 7
for row in range(rows):
    calendar_frame.grid_rowconfigure(row, weight=1)
for col in range(columns):
    calendar_frame.grid_columnconfigure(col, weight=1)

# Запускаем главный цикл обработки событий
root.mainloop()
```

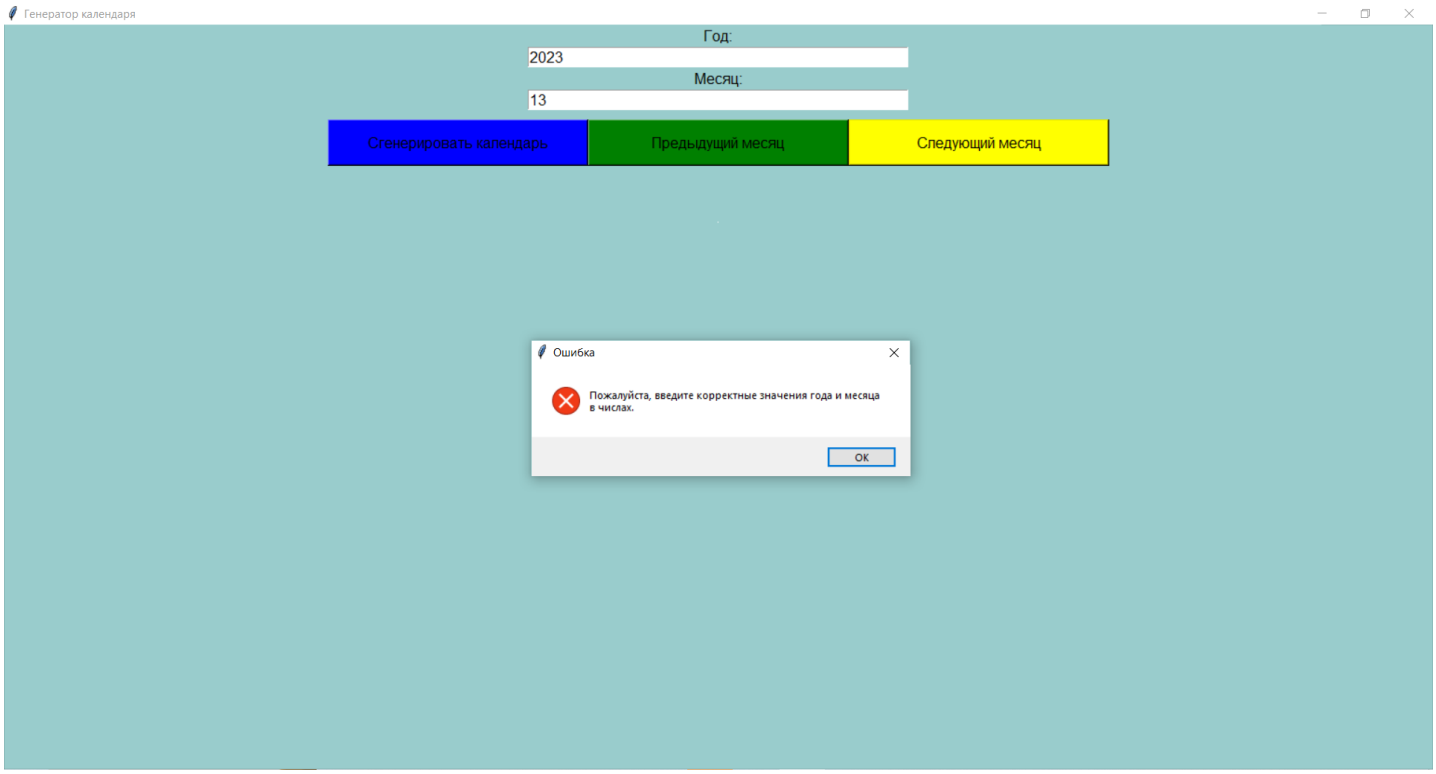
<u>Фотографии программы:</u>



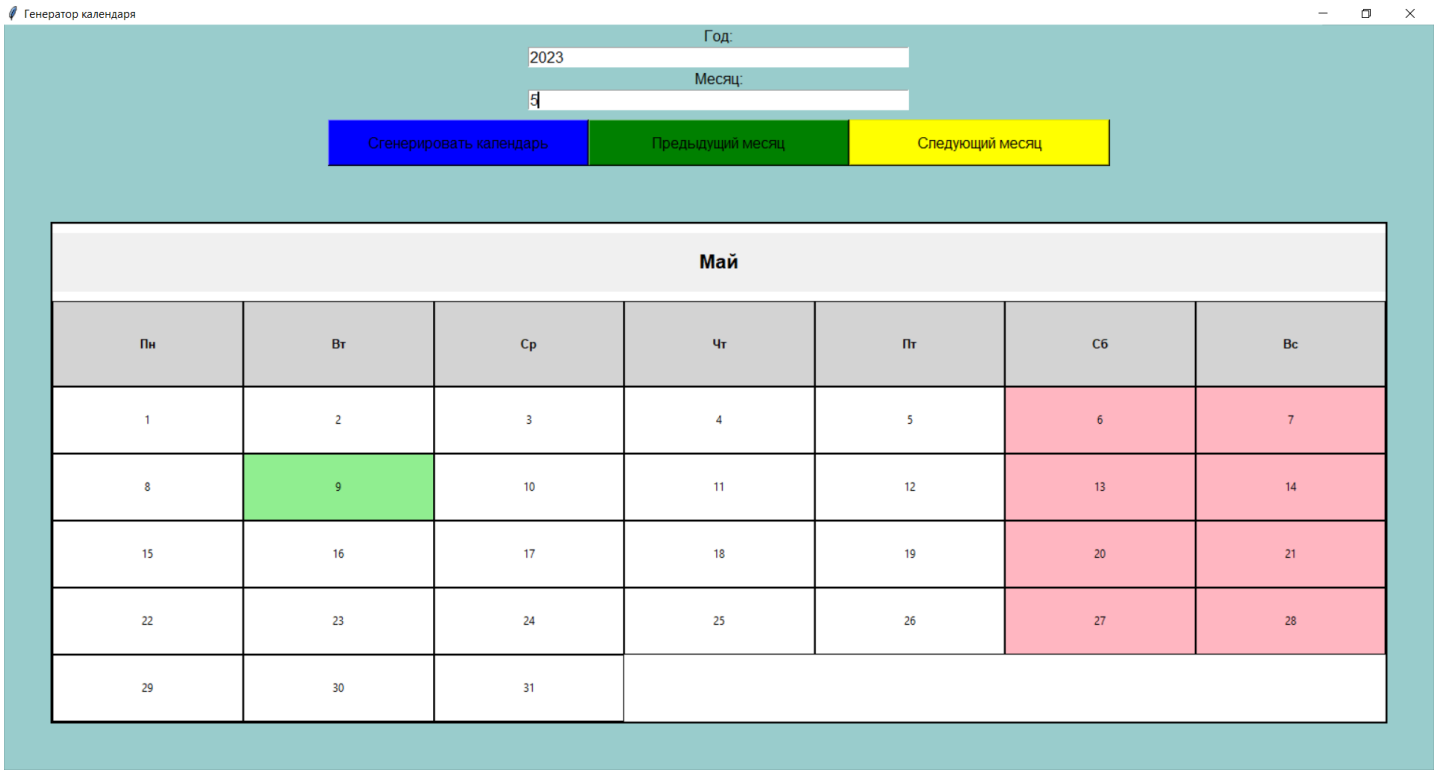
Главное окно программы



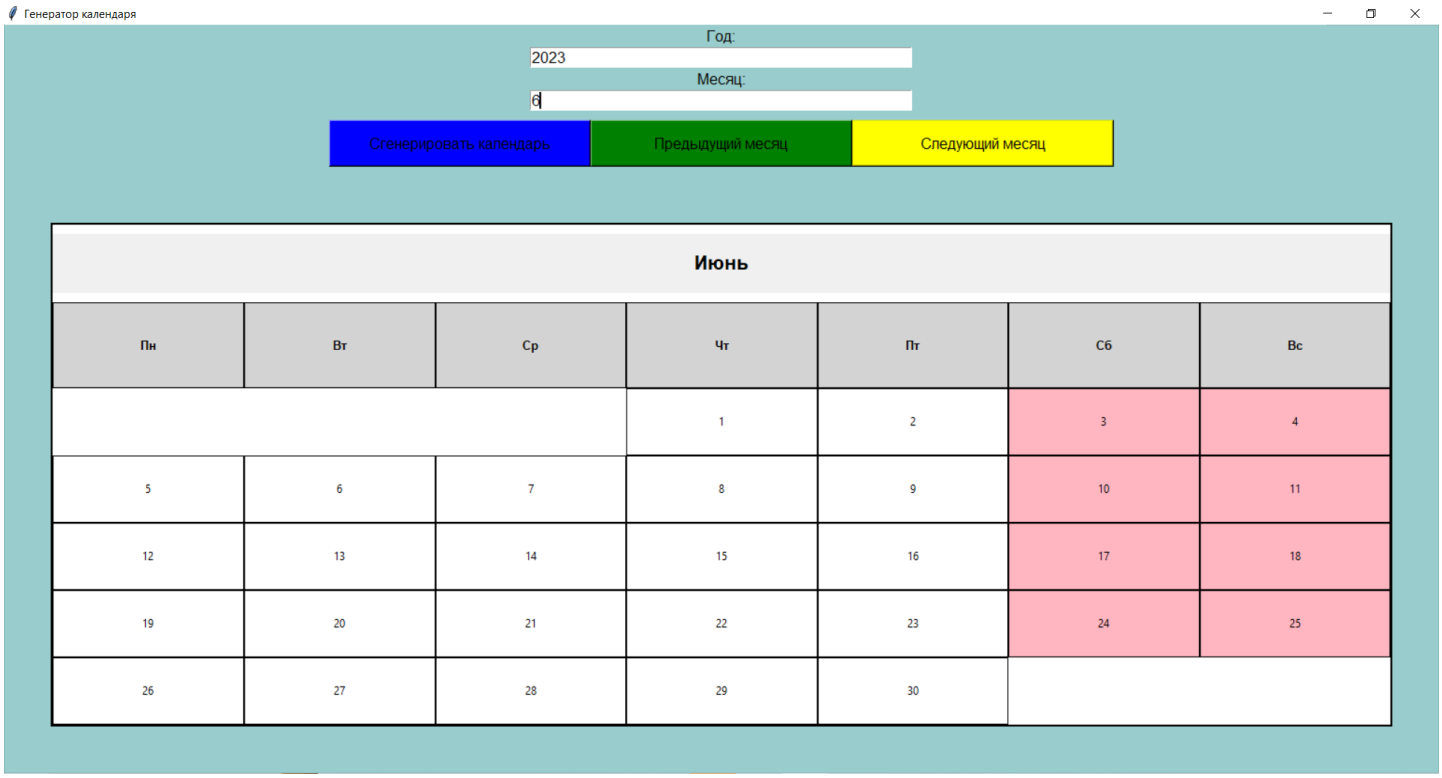
Ввод чисел: год и месяц



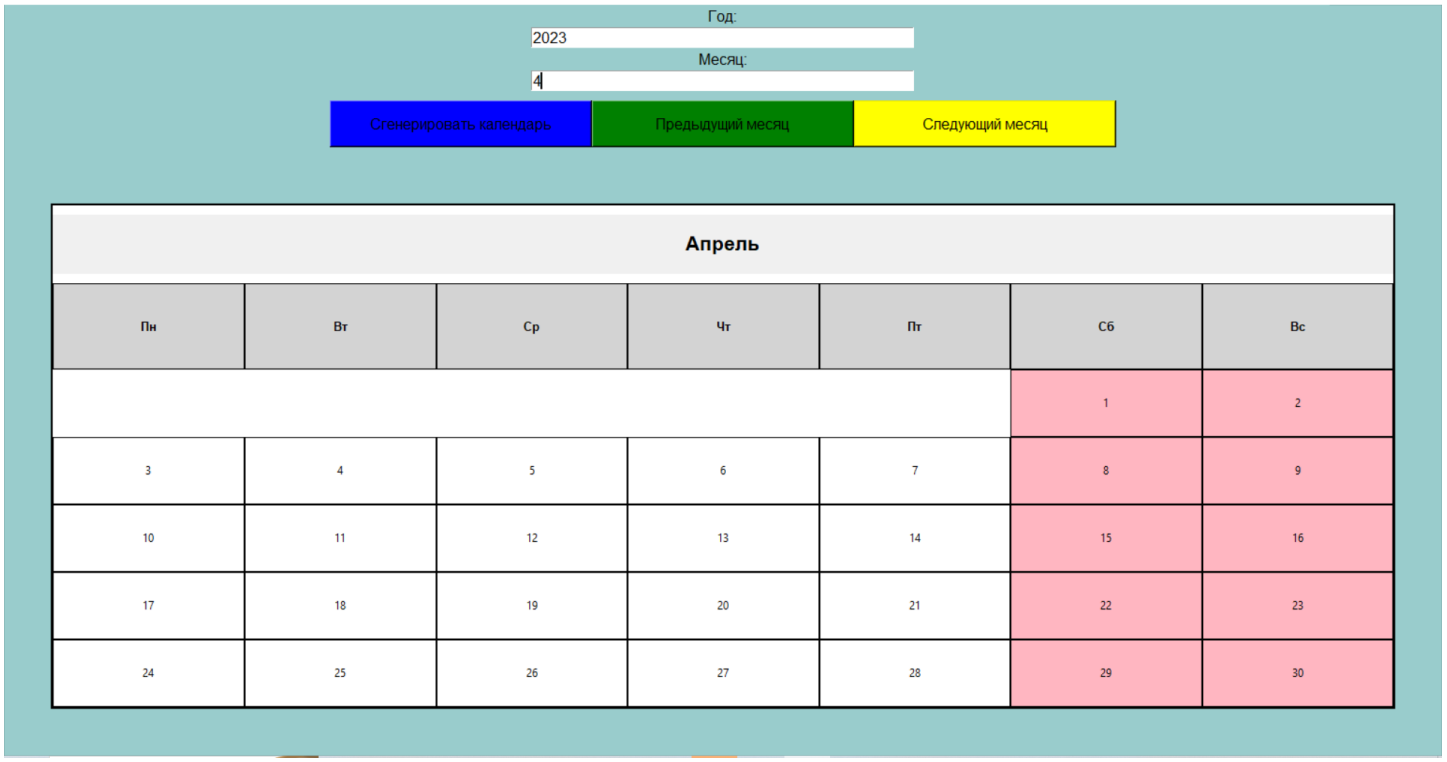
Если ввести некорректные значения



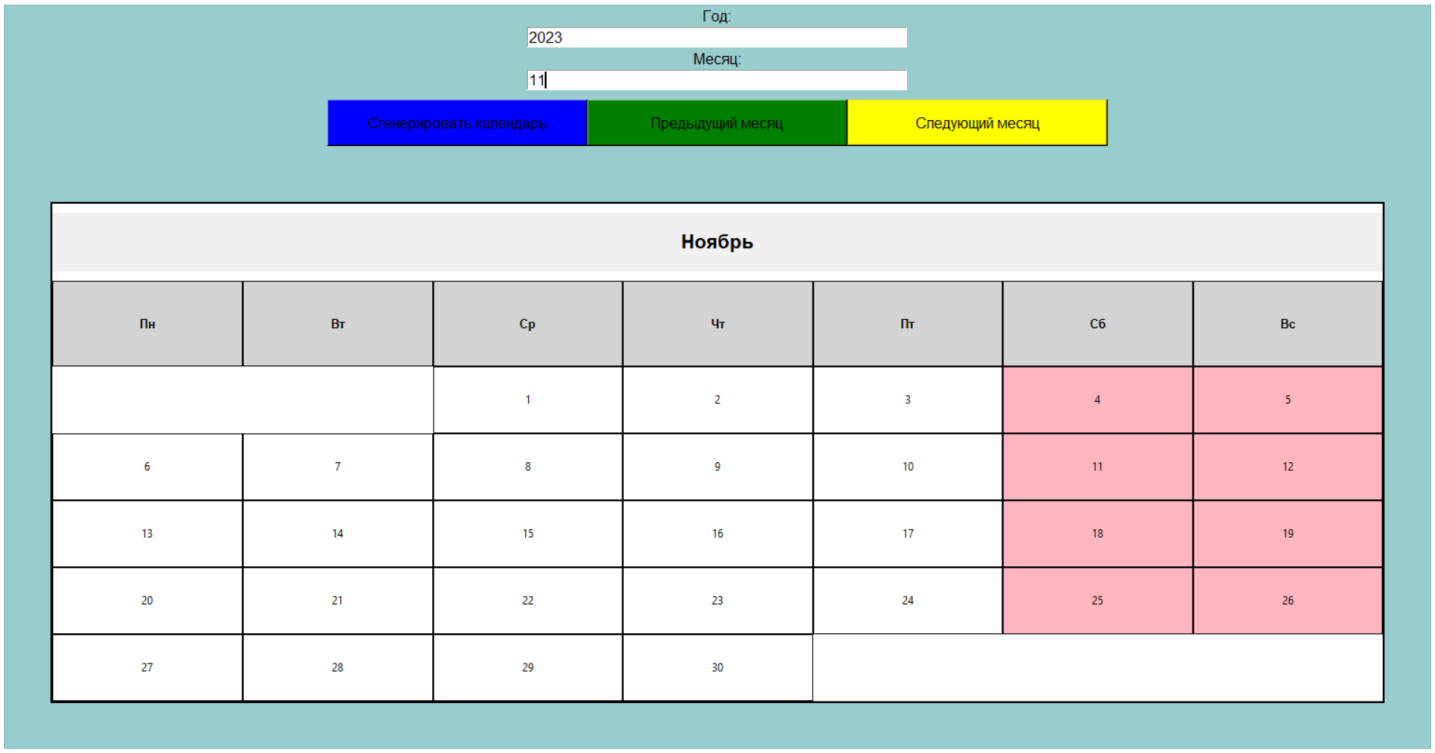
Отображение календаря: выходные дни отмечены розовым цветом, а праздничные зеленым цветом



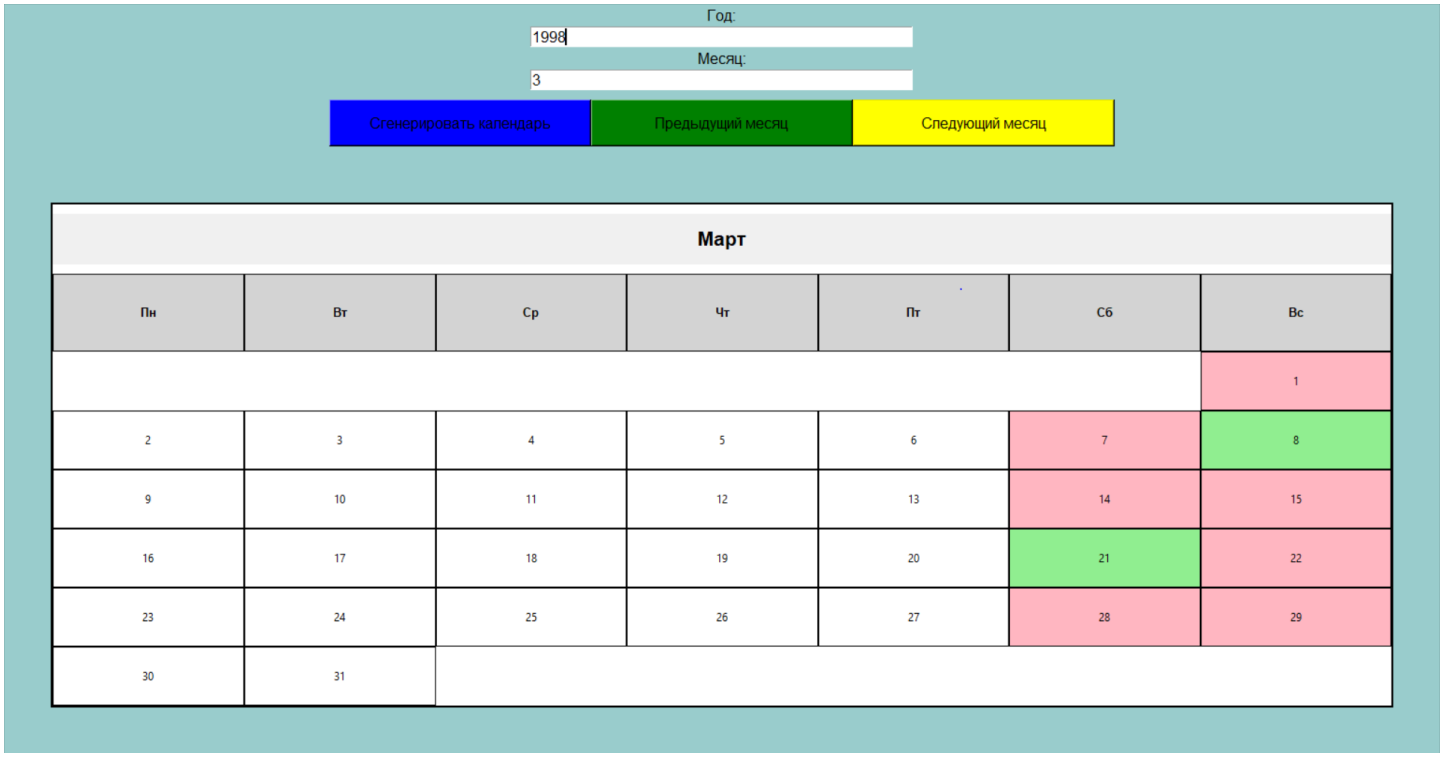
Переключение на “Следующий месяц”



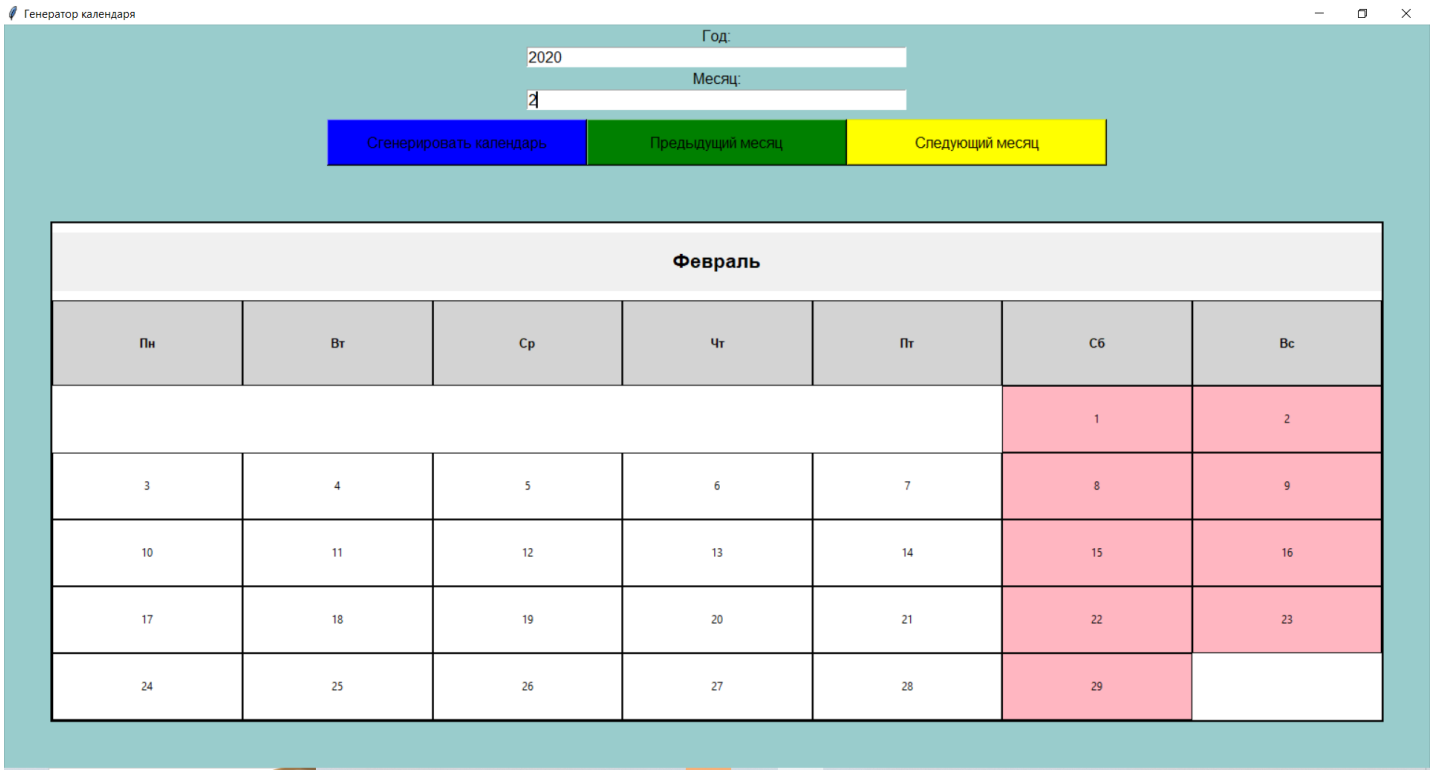
Переключение на “Предыдущий месяц”



Если мы хотим выбрать календарь на Ноябрь 2022, то вводим число месяца 11 (“Ноябрь”) и нажимаем на “Сгенерировать календарь”



Если мы хотим выбрать календарь на Март 1998, то вводим число месяца 3 (“Март”) и нажимаем на “Сгенерировать календарь”



Отображения календаря с високосным годом