Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**программное средство «ToDoDuck»**

БГУИР КП 1-40 01 01  3  ПЗ

Студент: гр. 751001 Бобко А. В.

Руководитель: асс. Шостак Е.В.

Минск 2018

СОДЕРЖАНИЕ

Введение

[1 Анализ предметной области 5](#_Toc514923443)

[1.1 Обзор аналогов 5](#_Toc514923444)

[1.2 Постановка задачи 8](#_Toc514923445)

[2 Разработка программного средства 10](#_Toc514923449)

[2.1 Структура программного средства 10](#_Toc514923450)

[2.2 Интерфейс программного средства 10](#_Toc514923451)

[2.3 Функциональная часть 15](#_Toc514923452)

[3 Тестирование программного средства 20](#_Toc514923454)

[4 Руководство пользователя 23](#_Toc514923455)

[Заключение 26](#_Toc514923456)

[Список использованных источников 27](#_Toc514923457)

Приложение А. Исходный код программы 28

**ВВЕДЕНИЕ**

Двадцать первый век — век огромного потока информации. В повседневной жизни человек постоянно сталкивается с различными данными. Каждый день люди пытаются держать в голове огромные списки дел, важные даты, мелкие бытовые вопросы, рецепты, инструкции. Также очень часто бывает необходимо запоминать большие объемы информации, различного рода формулы, речи для выступлений, определенные правила и законы. Кроме того, постоянно приходится сталкиваться с рекламой, которая встречается повсеместно: по радио, телевизору, в интернете, на улице – которая также остается у нас в голове и забивает нашу память.

Неудивительно, что у людей постоянно вылетает что-то из головы. Очень трудно уследить за всеми делами в огромном ритме жизни. Люди давно начали записывать то, что имеет наибольшую важность: каждый когда-нибудь составлял списки продуктов, которые необходимо купить, в школах мы регулярно вели дневники с домашним заданием, а количество различных ежедневников в магазинах только растет. Также известно, что многие гении составляли свои списки дел, ведь так можно проанализировать весь объем необходимых работ, что позволит правильно скорректировать свое время и успеть все сделать к сроку.

Так же наша память может стать хуже из-за многих факторов. Причиной тому необязательно может быть возраст. Обезвоживание, стресс, инфекционные заболевания — вот всего лишь несколько причин, помимо улетающих лет. Испортить память могут и определенные лекарства, и алкоголь, и плохое питание, и депрессия, и волнение, и проблемы с щитовидной железой. Но и годы тоже берут свое. Старея, мы замечаем, что наша память все чаще начинает нас подводить, так что составление списков дел — это скорее вопрос времени.

В интернете можно найти разные календари, которые помогут распределить свои дела по датам, но у большинства из них слишком перегруженный и сложный интерфейс, который будет понятен не для всех. Так же такие календари обладают огромным количеством функций, часть из которых практически никому не нужна.

Целью данного курсового проекта является анализ и разработка программного средства, позволяющего формировать свои заметки, размещать дела в календаре, которое будет понятно широкому кругу пользователей и будет выделятся своей простотой в использовании.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

В настоящее время можно найти большое количество различных органайзеров на просторах интернета. Они существуют практически на всех платформах, и каждый из них обладает огромным набором функций и инструментов. При создании органайзера необходимо учитывать тот фактор, что им может пользоваться огромное количество людей разных возрастов и профессий, поэтому необходимо делать его максимально удобным если не для всех, то для большинства пользователей. В ходе разработки были проанализированы и изучены некоторые аналоги, выявлены их положительные и отрицательные стороны. Далее проведены наиболее интересные примеры.

## Обзор аналогов

«Wunderlist» — это один из самых популярных органайзеров. Программное средство, принадлежащее компании Microsoft, является очень популярным в наши дни.

У данного приложения можно отметить очень красивый и понятный интерфейс. Один из самых больших достоинств этой программы является его кроссплатформенность: вы можете завести один аккаунт, а ваши дела будут синхронизироваться между вашими устройствами. Все записи хранятся на облачном хранилище, что позволит не потерять данные в случае экстренного завершения работы или поломки вашего устройства. Еще одним плюсом является оповещение о важных делах на почту, которое в случае чего можно отключить. Стоит отметить, что программное средство позволяет пользователям делиться своими задачами со своими друзьями.

К недостаткам можно отнести отсутствие наглядного календаря с делами, что делает немного затруднительным планирование вашей недели или месяца. Так же процесс синхронизации начинается вручную, из-за чего пользователь может постоянно забывать обновлять свой список дел.

Интерфейс данного приложения иллюстрирует рисунок 1.1:

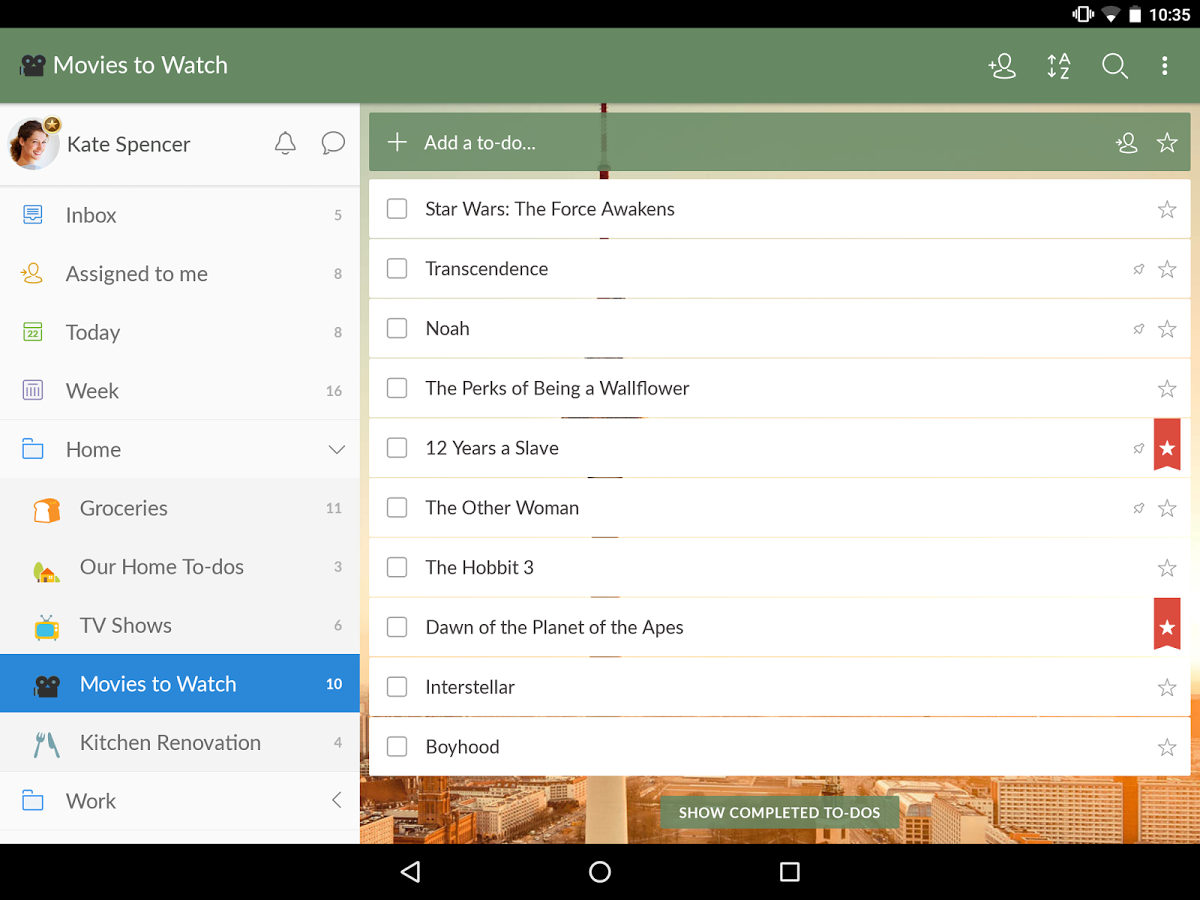


Рисунок 1.1 – Интерфейс программного средства «Wunderlist»

В качестве аналога можно привести «Google Календарь». Это мощная среда, для организации своих расписаний.

Количество возможностей данного сервиса действительно огромно. Вот только малая часть его достоинств:

- кроссплатформенность;

- огромный наглядный календарь с приятной для глаза цветовой гаммой;

- возможность ведения сразу нескольких календарей и переключение между ними. Так же можно добавлять доступные календари;  
 - сервис позволяет подробно описывать все аспекты какого-либо мероприятия;

- возможность настройки внешнего вида программы;

- поддержка оповещений на почту;

- наличие поиска по вашим задачам и интернету;

- благодаря диаграммам можно следить за течением времени днем;

- взаимодействие с «Google Задачами» и заметками из вашего смартфона.

Входе анализа были выявлены следующие недостатки:

- очень огромный функционал, часть из которого будет использоваться только узким кругом лиц;

- отсутствие скрытия выполненных дел, из-за чего можно запутаться на большом окне с обилием задач;

- работа на компьютере возможна только с доступом в интернет.

Однако стоит заметить, что огромное количество положительных сторон делают данный сервис одним из лучших в своем роде. Интерфейс данного приложения проиллюстрирован на рисунке 1.2:

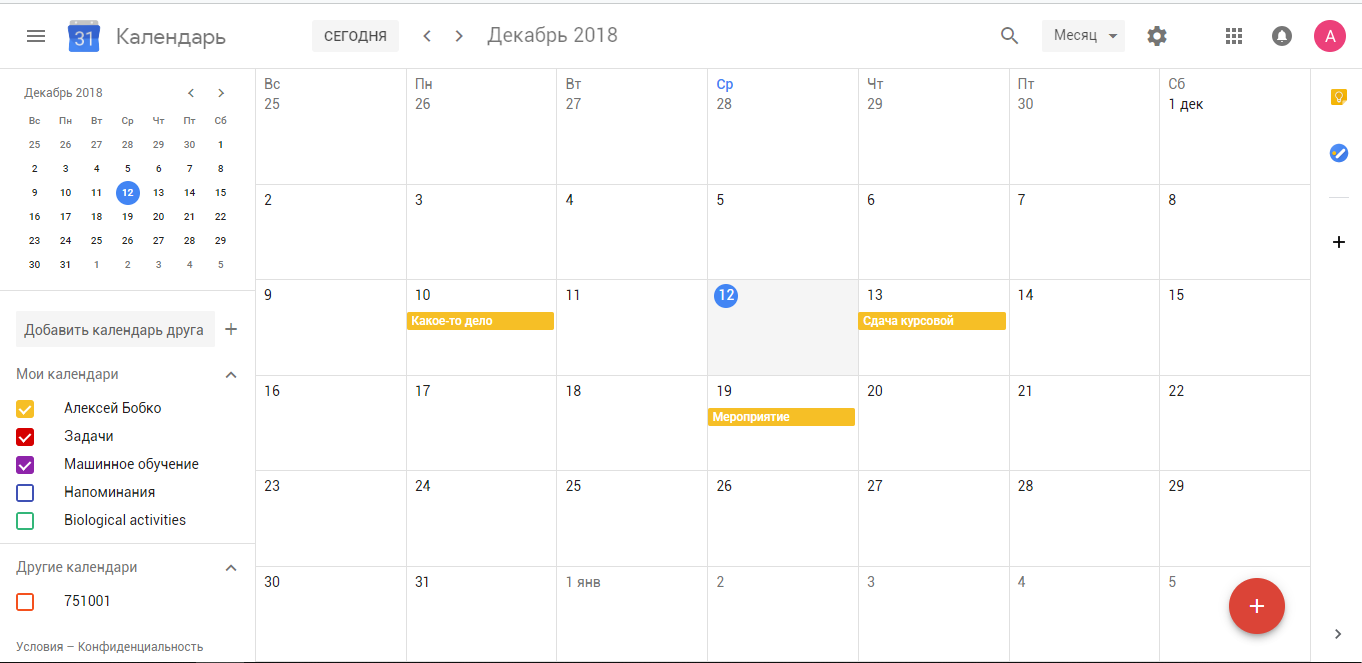


Рисунок 1.2 – Интерфейс сайта сервиса «Google Календарь»

«KissMyPlan» — это планировщик дел, предназначенный для смартфонов под операционной системой Android. В процессе анализа приложения был выявлен ряд его положительных и отрицательных сторон. Далее приводится часть из них.

Достоинства:

- уведомления о задачах на телефон;

- невыполненные задачи все равно уходят в архив, что помогает не отвлекаться на старые записи;

- возможность отложить дело;

- возможность установки неопределенной даты;

- доступно добавление фото и аудио материалов в свой календарь;

- возможность повторять одно и тоже дело регулярно.

Недостатки:

- отсутствие кроссплатформенности;

- отсутствие наглядного календаря;

- не очень приятный интерфейс;

- при наличии огромного количества задач все начинает теряться.

Интерфейс данного приложения представляет рисунок 1.3.

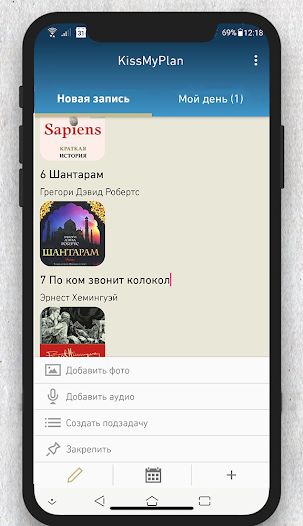


Рисунок 1.3 – Интерфейс приложения «KissMyPlan»

* 1. **Постановка задачи**

Целью данной курсовой работы является создание органайзера, который поможет пользователю планировать свое расписание, делать заметки и удобно взаимодействовать с программным продуктом.

В приложении планируется реализовать ряд функций, связанных с работой с календаря, алгоритмами обработки даты, сохранение списков дел.

Для корректной работы с интерфейсом планируется организовать:

– автоматическая блокировка ненужных кнопок;

– наглядность всех возможных функций программы;

– интуитивно понятный и интерфейс программного продукта.

Для разработки алгоритмической части приложения необходимо:

– спроектировать функции, реализующие работу с временем для генерации календаря;

– реализовать функции, которые дают возможность сохранять данные в файл и загружать их оттуда.

Для справочной части разработки приложении необходимо:

– подсказки пользователю во время эксплуатации приложения;

– наглядная короткая документация, объясняющая все возможности программного продукта.

В процессе реализации органайзера планируются разработать алгоритмы, реализующие все необходимые функции для работы с датой и временем. Реализовать удобный и практичный пользовательский интерфейс, соответствующий современным тенденциям в области создания программных продуктов.

В результате исследования аналогов программного продукта было принято решение особое внимание уделить разработке качественного интерфейса, реализации всех функций, которые необходимы пользователю при планировании дел, быстроте и качеству работы программного средства.

Для разработки программного средства использован язык программирования С#. Среда разработки — Microsoft Visual Studio 2017(Windows Forms). Использование данной среды разработки дает множество возможностей в создании современных оконных приложений Windows, отвечающих современным тенденциям модульности при разработке программного продукта. С помощью данной среды можно очень легко и быстро разработать качественное оконное приложение с приятным для пользователя интерфейсов.

2 Разработка программного средства

2.1 Структура программного средства

В данном приложении необходимо использовать 5 модулей:

− fMainMenu – главное окно программы, реализация основных функций, связанных с главным меню, вызовов других классов и модулей;

− fCalendar – окно программы, предназначенное для генерации и отображения календаря, а также взаимодействия работы с ним;

− fToDoDayList – модуль, в котором спроектировано и реализовано окно для взаимодействия пользователя с календарем и сохранением его данных;

− fDesks – окно программы, в котором осуществляется создание, управление и хранение досок для записей пользователя;

− fHelp – окно помощи программного средства, отображающее краткую справочную информацию по использованию приложения.

**2.2 Интерфейс программного средства**

Для разработки гибкого интерфейса программного продукта было принято решение использовать формы Windows Forms, так как они позволяют сделать приятный пользовательский интерфейс.

Главное окно состоит из нескольких крупных составляющих, таких как четыре кнопки мерю («Календарь», «Доски», «Справка», «Выход»), заголовок и подзаголовок. Внешний вид формы описывается частью кода:

// fMainMenu

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.Color.Maroon;

this.ClientSize = new System.Drawing.Size(700, 480);

this.Controls.Add(this.bExit);

this.Controls.Add(this.bDesk);

this.Controls.Add(this.bHelp);

this.Controls.Add(this.bCalendar);

this.Controls.Add(this.lSubTitle);

this.Controls.Add(this.lTitle);

this.DoubleBuffered = true;

this.ForeColor = System.Drawing.Color.Maroon;

this.FormBorderStyle =

System.Windows.Forms.FormBorderStyle.None;

this.MaximumSize = new System.Drawing.Size(700, 480);

this.MinimumSize = new System.Drawing.Size(700, 480);

this.Name = "fMainMenu";

this.StartPosition =

System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "ToDoDuck";

this.TransparencyKey = System.Drawing.Color.Transparent;

this.ResumeLayout(false);

this.PerformLayout();

Внешний вид данной формы представлен на рисунке 2.1:



Рисунок 2. ­­– Внешний вид главного меню приложения

Для единства стиля разработки интерфейса было принято решение разрабатывать кнопки, элементы визуализации, дочерние формы в одном стиле. По этой причине были подобраны сочетающиеся цвета, которые будут приятны пользователю. Также была сделана подсветка кнопок при нажатии и наведении. Программное средство сделано в современном минималистичном стиле.

По нажатию на кнопку «Календарь» открывается соответствующее окно с сгенерированным календарем. В календаре присутствует подсветка сегодняшней даты, а также подсветка чисел, в которых прописаны какие-то задачи. Внешний вид данного окна представлен на рисунке 2.2:



Рисунок 2.2 ­­– Внешний вид окна с календарем

По нажатию на кнопки с стрелками пользователь может переключаться между месяцами. Все кнопки календаря создаются динамически, что позволяет изображать месяц с любым расположением чисел. По кнопке «Назад» пользователь может вернуться в главное меню программы.

По нажатию на любое число откроется дочернее окно со списком задач, запланированных на это число. На рисунке 2.3 представлен внешний вид окна календаря вместе с дочерней формой:

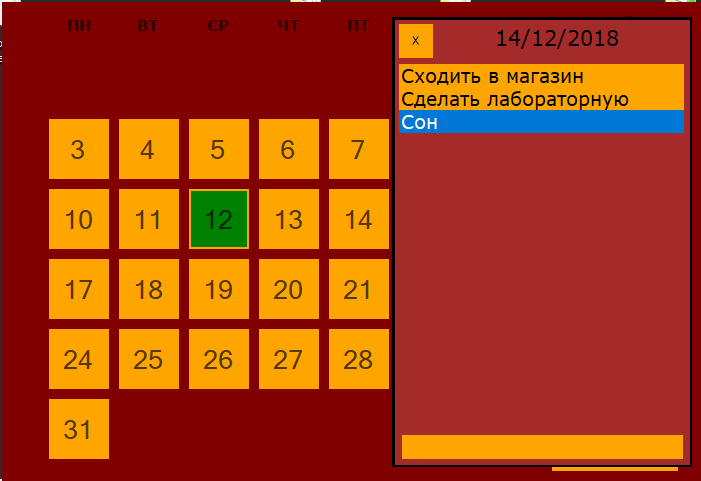


Рисунок 2.3 ­­– Окно календаря вместе с формой списка дел

При нажатии на кнопку «Доски» откроется соответствующее окно с досками. Данное окно позволяет создавать доски, удалять их, редактировать и сохранять для дальнейшего использования. Все сохранения и загрузки происходят автоматически. Благодаря кнопкам с изображением стрелок пользователь может переключаться между досками. При отсутствии каких-либо записей об этом сообщит соответствующая надпись. Вариант данной формы при первом запуске программы изображен на рисунке 2.4:

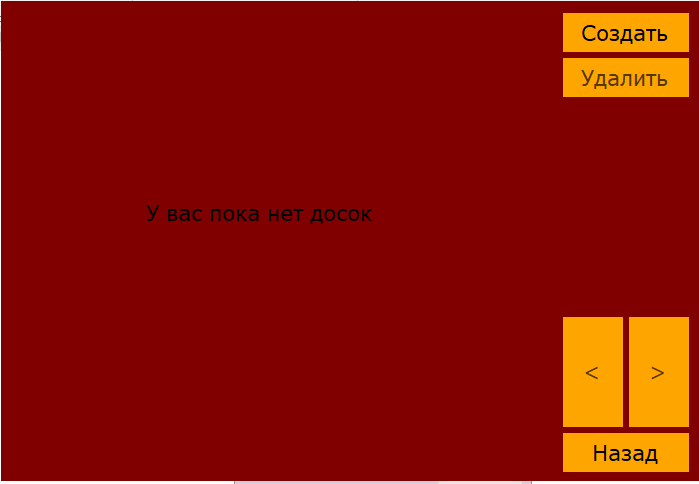


Рисунок 2.4 ­­– Внешний вид формы досок

Так же в главном меню присутствует кнопка «Справка». При нажатии на которую пользователь сможет получить подробную инструкцию по использованию программного средства, что позволит ему легко освоить приложение, а также задействовать весь функционал программы. Пример вызова формы с календарей:

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

{

Form fCalendar = new fCalendar();

//создаем новую форму календарь

fCalendar.Left = this.Left;

fCalendar.Top = this.Top;

//устанавливаем такие же координаты

//как у родительской формы

fCalendar.Show();

//показываем форму календарь

this.Hide();

//скрываем родительскую форму

}

Возврат обратно к главному меню происходит по похожему коду:

private void bBack\_Click(object sender, EventArgs e) {

Form fMainMenu = Application.OpenForms[0];

//присваиваем первую открытую форму

fMainMenu.StartPosition = FormStartPosition.Manual;

fMainMenu.Left = this.Left;

fMainMenu.Top = this.Top;

//устанавливаем стартовую позицию родительской формы

fCalendar.ActiveForm.Close();

//закрываем форму календарь

fMainMenu.Show();

//отображаем родительскую форму

}

Формы «Справка» и «Доски» вызываются и закрываются по аналогичному принципу.

2.3 Функциональная часть

Одной из основных функций является создание календаря на соответствующей форме. В момент отрисовки каждая кнопка создается динамически. Местоположение каждого числа также рассчитывается во время работы программы. Основной цикл по созданию одного месяца представлен следующим кодом:

while (daysCnt <= DaysInMonth) {

while ((numOfDayInWeek != (DAYS\_IN\_WEEK + 1))

&& (daysCnt <= DaysInMonth))

{

//DAYS\_IN\_WEEK - количество дней в неделе

int x = (numOfDayInWeek - 1) \* (SIZE\_OF\_CELL +

CELL\_DISTANCE) + SIZE\_OF\_CELL / 2;

//рассчет координаты х для кнопки

//SIZE\_OF\_CELL - длина стороны кнопки

//CELL\_DISTANCE - расстояние между кнопками

numbersOfMonth.Add(new DateButton());

//добавление новой кнопки в список

numbersOfMonth.Last().Text = daysCnt.ToString();

numbersOfMonth.Last().Font = new

Font(FONT\_OF\_NUM\_OF\_MONTH, SIZE\_OF\_NUM\_OF\_MONTH);

//FONT\_OF\_NUM\_OF\_MONTH - шрифт на кнопке

//SIZE\_OF\_NUM\_OF\_MONTH - размер шрифта на кнопке

numbersOfMonth.Last().Width =

numbersOfMonth.Last().Height = SIZE\_OF\_CELL;

numbersOfMonth.Last().BackColor = Color.Orange;

numbersOfMonth.Last().FlatAppearance.BorderSize = 2;

numbersOfMonth.Last().FlatAppearance.BorderColor =

Color.Orange;

numbersOfMonth.Last().FlatStyle = FlatStyle.Flat;

//установка свойств кнопки

numbersOfMonth.Last().Location = new Point(x, y);

//размещение кнопки на форме

pane.Controls.Add(numbersOfMonth.Last());

numOfDayInWeek++;

//увеличение дня недели

daysCnt++;

//следующее число

}

y += SIZE\_OF\_CELL + CELL\_DISTANCE;

//рассчет нового у

numOfDayInWeek = 1;

}

В данном цикле происходит создание кнопки-числа месяца. Первоначально во вложенном цикле программа создает одну неделю. Все кнопки являются аналогичными, поэтому было принято решение создать массив кнопок. Кнопки являются экземплярами класса DateButton, который в свою очередь является наследованным от встроенного класса Button. Основные величины были вынесены в константы, чтобы было проще изменять свойства элементов при разработке. Также в процессе разработки были подобраны наиболее оптимальные значения констант, которые помогли сделать приятный интерфейс.

Также одной из важных функций является создание формы с досками. При запуске данного окна осуществляется проверка наличия файлов с предыдущими записями, созданных во время предыдущих сеансов. Далее, при наличии таких файлов, происходит загрузка данных из текстового файла в программу. За все эти действия отвечает конструктор формы с досками. Реализация загрузки данных представляет собой:

int i = 0;

string deskFileLocation = NAME\_OF\_DIRECTOTY + "\\desk\_" +

i.ToString() + ".txt";

//путь к первому файлу

while (File.Exists(deskFileLocation)) {

//цикл выполняется до тех пор, пока есть нжные файлы

DeskCreate();

//метод, создающий доску

using (StreamReader desk = File.OpenText(deskFileLocation))

{

desks.Last().Text = desk.ReadToEnd();

//чтение из файла

}

desks.Last().Visible = false;

i++;

openDeskIndex++;

deskFileLocation = NAME\_OF\_DIRECTOTY + "\\desk\_" +

i.ToString() + ".txt";

//переход к следующему файлу

}

В процессе загрузки данных параллельно создаются доски под эти данные. Все это осуществляется в цикле, приведенном выше. Создание же доски осуществляется через следующую функцию:

private void DeskCreate() {

desks.Add(new TextBox());

//добавление новой доски в список досок

desks.Last().Font = new Font(FONT\_IN\_DESKS,

SIZE\_OF\_CHARS\_IN\_DESKS);

desks.Last().Multiline = true;

desks.Last().BackColor = Color.Orange;

desks.Last().BorderStyle = BorderStyle.FixedSingle;

desks.Last().Location = new Point(POSITION\_OF\_DESK\_X,

POSITION\_OF\_DESK\_Y);

desks.Last().Height = HEIGHT\_OF\_DESK;

desks.Last().Width = WIDTH\_OF\_DESK;

//задание параметров доски

this.Controls.Add(desks.Last());

//добавление доски на основную панель

}

Длина и высота доски хранятся соответственно в константах WIDTH\_OF\_DESK и HEIGHT\_OF\_DESK, а ее координаты в POSITION\_OF\_DESK\_X и POSITION\_OF\_DESK\_Y.

При нажатии на кнопку «Назад» все записи автоматически сохраняются в файл. Функция, осуществляющая сохранение представлена ниже:

private void SaveDesksInFile() {

if (Directory.Exists(NAME\_OF\_DIRECTOTY)) {

DeleteDirectory(NAME\_OF\_DIRECTOTY);

//удаляем старую папку с досками

}

if (desks.Count > 0) {

Directory.CreateDirectory(NAME\_OF\_DIRECTOTY);

//создаем новую папку для досок

for (int i = 0; i < desks.Count; i++) {

using (StreamWriter desk =

File.CreateText(NAME\_OF\_DIRECTOTY +

"\\desk\_" + i.ToString() + ".txt")) {

desk.Write(desks[i].Text);

//записываем данные в файл

}

}

}

}

Пример полученных файлов представлен на рисунке 2.5:

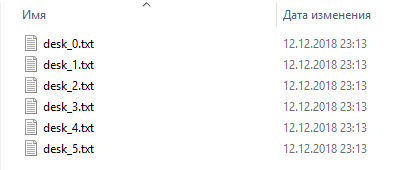


Рисунок 2.5 ­­– Полученная после сохранения папка

В форме fToDoDayList было разработано отправка текста в список путем нажатия клавиши Enter. Также были добавлены необходимые проверки. Код обработки представлен ниже:

private void tbEnter\_KeyUp(object sender, KeyEventArgs e) {

if ((e.KeyCode == Keys.Enter)||(e.KeyCode == Keys.Return)) {

// e == Enter

if (tbEnter.Text.Length > 0) {

//проверка, чтобы строка была непустой

string tmpS1 = tbEnter.Text;

tmpS1 = tmpS1.Trim(' ');

if (tmpS1.Length > 0) {

//проверка на то, чтобы строка не состояла //лишь из пробелов

string tmpS2 = tbEnter.Text;

tmpS2 = tmpS2.TrimStart(' ');

lbToDoList.Items.Add(tmpS2);

//удаление пробелов вначале + добавление

tbEnter.Clear();

if (lbToDoList.Items.Count != 1) {

lbToDoList.Height = lbToDoList.Height +

ADD\_HEIGHT;

lbToDoList.SetSelected(

lbToDoList.Items.Count – 1, true);

}

}

else

tbEnter.Clear();

//если строка не подходит - очищаем

}

}

}

Для удаления строки из списка необходимо дважды кликнуть по необходимой строке. Функция, реализующая удаление:

private void lbToDoList\_MouseDoubleClick(object sender,

MouseEventArgs e)

{

if (lbToDoList.Items.Count > 0) {

//проверка, что список не пуст lbToDoList.Items.RemoveAt(lbToDoList.SelectedIndex);

//удаление элемента

if (lbToDoList.Items.Count > 0) {

lbToDoList.Height = lbToDoList.Height –

SUB\_HEIGHT;

//SUB\_HEIGHT – дельта, на которую меняется ListBox lbToDoList.SetSelected(lbToDoList.Items.Count - 1,

true);

//выделяем последний элемент

//выравнивение ListBox

}

}

}

Константа SUB\_HEIGHT показывает, насколько увеличился или уменьшился наш объект lbToDoList по высоте. При достижении элементом максимальной длины, в lbToDoList появляется [Scrollbar, благодаря чему мы можем хранить любое количество элементов в списке.](https://www.google.by/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&ved=2ahUKEwiXyIy3j5vfAhWrxYUKHSGMAR4QFjAIegQIBRAB&url=http%3A%2F%2Fcontext.reverso.net%2F%25D0%25BF%25D0%25B5%25D1%2580%25D0%25B5%25D0%25B2%25D0%25BE%25D0%25B4%2F%25D0%25B0%25D0%25BD%25D0%25B3%25D0%25BB%25D0%25B8%25D0%25B9%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B9-%25D1%2580%25D1%2583%25D1%2581%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B9%2Fscrollbar&usg=AOvVaw2KPfS6o-B1QeaE8ov-PVUx)

3 тестирование программного средства

В ходе тестирования программы были выявлены определенные недостатки и ошибки в работе программного средства.

В окне досок, после удаления всех записей и выходе в главное меню при дальнейшем запуске программы оставались кое-какие данные. Вначале была предпринята попытка перезаписывать имеющиеся файлы и контролировать их содержимое, но появилась проблема того, что определенные файлы перестали использоваться. Проблема была решена путем удаления папки с файлами после чтения и создания новой папки для сохранения новых данных. Функция, исправляющая данную ошибку приведена ниже:

private void DeleteDirectory(string target\_dir) {

string[] files = Directory.GetFiles(target\_dir);

string[] dirs = Directory.GetDirectories(target\_dir);

foreach (string file in files) {

File.SetAttributes(file, FileAttributes.Normal);

File.Delete(file);

}

foreach (string dir in dirs) {

DeleteDirectory(dir);

}

Directory.Delete(target\_dir, false);

}

В окне календарей обнаружилась ошибка в том, что пользователь мог сколько угодно раз вызывать дочернее окно fToDoDayList, что вызывало путаницу и могло привести к экстренному завершению программы. Пришлось сделать блокировку родительского окна до тех пор, пока пользователь не закроет дочернюю форму.

Первоначально календарь должен был рисоваться вручную, а не путем прикрепления кнопок. Но появились проблемы по перерисовке календаря при сворачивании приложения. Тогда была предпринята попытка заменить всю отрисовку созданием кнопок. Это помогло сильно оптимизировать код, избежать появления других ошибок и сделать интерфейс более приятным для пользователя.

Пример с нарисованным календарем представлен на рисунке 3.1:

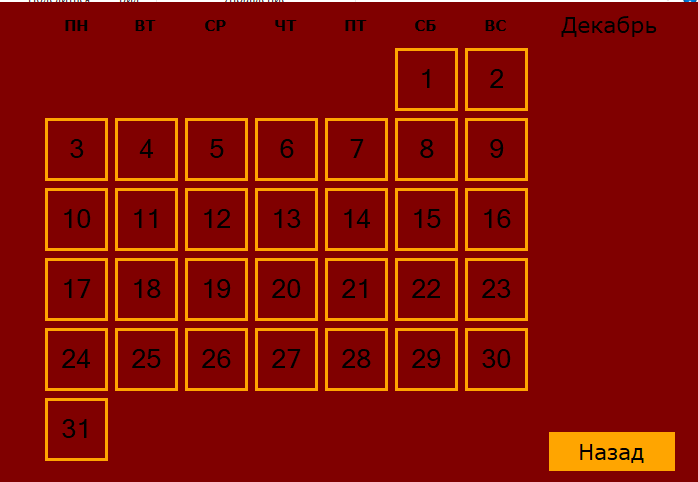


Рисунок 3.1 ­­– Окно fCalendar с нарисованным календарем

Также было замечено, что в окне fToDoDayList вмещаются только короткие строки, а остальное выходит за рамки списка, что являлось очень большим недостатком, так как просмотр задач становился очень неудобным. Для того, чтобы пользователь мог вводить текст большой длины были добавлены всплывающие подсказки, которые содержали текст целиком. На рисунке 3.2 можно увидеть пример такой подсказки:

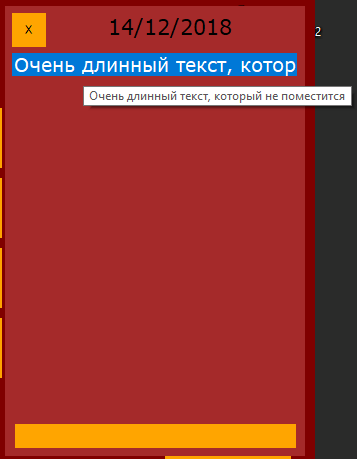


Рисунок 3.2 ­­– Окно fToDoDayList с всплывающей подсказкой

Остальные проблемы были связаны с недочётами на стадии проектирования и были устранены в процессе разработки приложения. Больше ошибок в ходе тестирования обнаружить не удалось.

4 Руководство пользователя

Для начала работы приложения необходимо запустить исполняемый файл «ToDoDuck.exe». Для корректной работы приложения необходимо иметь операционную систему Windows версии XP, 7, 8 или 10.

При запуске приложения мы наблюдаем главное меню с четырьмя кнопками: «Календарь», «Доски», «Справка», «Выход».

Для выхода из программы нажмите кнопку «Выход».

Для получения инструкции по использованию приложения необходимо нажать на кнопку «Справка». В появившемся окне будут картинки со всем руководством по использованию программы. Для переключения между картинками нужно использовать кнопки со стрелочками.

Чтобы приступить к работе с календарем, нажмите на кнопку «Календарь». После нажатия открывается окно, на котором изображен календарь на текущий месяц. Число с зеленой заливкой – сегодняшняя дата, кнопка с белой границей – дата, в которой есть какие-либо задачи. Для переключения между месяцами необходимо использовать две кнопки со стрелками. Нижняя граница календаря – текущий год, все, что было ранее удаляется.

Для того, чтобы посмотреть задачи на определенный день, а также добавить новые или удалить имеющиеся, необходимо кликнуть на кнопку с нужным числом. После нажатия появится небольшое окно с списком дел. Вверху располагается дата, которую вы выбрали. Под ней находится сам список задач. Поле для ввода находится внизу окна. Чтобы добавить дело в список, необходимо ввести его в нижнее поле, затем начать на клавиатуре клавишу Enter. Для того, чтобы удалить дело из списка, нужно просто дважды кликнуть на нужный элемент, и он удалится из списка. Чтобы вернуться назад к календарю, необходимо нажать на кнопку с крестиком в углу окна. На рисунке 4.1 подписаны поле ввода и список задач:

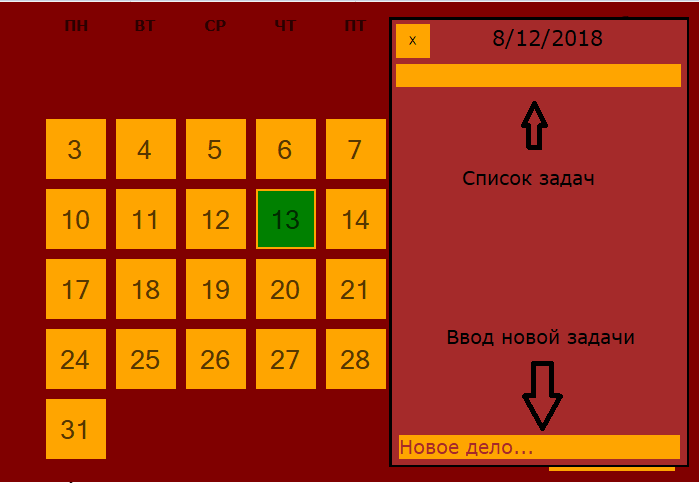


Рисунок 4.1 – Поле ввода и список задач в окне fToDoDayList

Чтобы вернуться в главное меню, необходимо нажать на кнопку «Назад» (такой же метод возврата из окон с досками и справки).

После нажатия на кнопку «Доски» откроется соответствующее окно. Если вы ранее создавали какие-либо доски в приложении, то они откроются в том же порядке, что и при прошлой работе. Если же у вас вообще нету досок, то вы увидите текст «У Вас пока нет досок». Чтобы создать новую доску, необходимо нажать на кнопку «Создать». После нажатия появится пустая доска, с которой вы уже можете начинать работу. Для того, чтобы начать что-то записывать на пустую доску, просто кликните на нее и начните вводить свой текст. Вы можете вести одновременно несколько досок. Для переключения между ними используйте кнопки с стрелками вправо и влево. Если вы хотите удалить доску, то для этого нажмите на кнопку «Удалить». Если удаляемая доска находится между другими досками, то они просто сдвинутся в нужную сторону. Как только вы закончите работу с досками, просто нажмите на кнопку «Назад». Все ваши изменения и записи сохранятся, и при следующей работе вы сможете спокойно продолжить с теми же записями. На следующем рисунке кратко представлен функционал формы с досками:



Рисунок 4.2 – Краткое описание окна с досками

При возникновении каких-либо вопросов всегда можно пересмотреть подробную встроенную справку.

ЗАКЛЮЧЕНИЕ

В наши дни человеку нужно держать очень много дел в голове. Так как в наши дни практически у каждого есть свой персональный компьютер или мобильный телефон, были разработаны специальные органайзеры, в которые можно управлять всеми своими задачами, создавать расписание на год вперед.

В рамках данного курсового проекта было разработано программное средство «ToDoDuck», которое поможет пользователю создавать свое расписание, планировать дела заранее, делать записи и заметки. Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* удобный и интуитивно понятный интерфейс;
* подробная справка по использованию приложения;
* генерация встроенного календаря и загрузка данных в него.

Для успешного выполнения всех поставленных целей потребовалось изучить объектно-ориентированные возможности языка C#, изучить основные принципы данной парадигмы, а также освоить создание собственных компонентов на основе уже существующих.

В дальнейшем можно сделать приложение кроссплатформенным, попробовать усовершенствовать интерфейс программы, а также расширить функционал программного средства. В последующем можно добавить визуальные эффекты. Еще возможно сделать окно настроек, чтобы пользователь смог менять характеристики приложения, такие как размер окна и цветовую тему программы.

И главное помнить, что мечты — это планы, реализовавшиеся в срок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Зиборов, В. Visual C# 2012 на примерах / В. Зиборов. - М.: БХВ-Петербург, 2015.

[2] Вагнер, Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 c.

[3] MSDN – сеть разработчиков Microsoft [электронный ресурс]. Электронные данные. Режим доступа: https://msdn.microsoft.com – Дата доступа: 02.10.2018.

[4] Шилдт, Г. Полный справочник по С#: учеб. Пособие. – СПб, 2007.

[5] Microsoft Docs [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.microsoft.com/en-us/dotnet/csharp/>.

[6] Введение в Windows Forms [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/windowsforms/1.1.php.

[7] Хейлсберг, А. Язык программирования C#. – М.: Питер, **2016**.

приложение а  
исходный код программы

Главная форма: (код)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ToDoDuck

{

public partial class fMainMenu : Form

{

public fMainMenu()

{

InitializeComponent();

}

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

{

Form fCalendar = new fCalendar();

//создаем новую форму календарь

fCalendar.Left = this.Left;

fCalendar.Top = this.Top;

//устанавливаем такие же координаты

//как у родительской формы

fCalendar.Show();

//показываем форму календарь

this.Hide();

//скрываем родительскую форму

}

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

{

Form fDesks = new fDesks();

fDesks.Left = this.Left;

fDesks.Top = this.Top;

fDesks.Show();

this.Hide();

}

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

{

Form fHelp = new fHelp();

fHelp.Left = this.Left;

fHelp.Top = this.Top;

fHelp.Show();

this.Hide();

}

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

{

this.Close();

}

}

}

Окно fCalendar (код):

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.IO;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ToDoDuck

{

public partial class fCalendar : Form

{

private class DateButton : Button

{

public List<string> toDo = new List<string>();

}

DateTime displayedDate = new DateTime();

Boolean childWinIsOpen = false;

const int SIZE\_OF\_CELL = 60;

const int CELL\_DISTANCE = 10;

const int DAYS\_IN\_WEEK = 7;

const int START\_Y\_FOR\_MONTH\_TABLE = 20;

int lastClicked = 0;

int[] posOfNameOfDayOfWeek = new int[DAYS\_IN\_WEEK];

List<DateButton> numbersOfMonth = new List<DateButton>();

public class TtoDo

{

public DateTime date;

public List<string> tasks;

}

private List<TtoDo> allTasks = new List<TtoDo>();

const string TASKS\_FILE\_LOCATION = "tasks.txt";

public fCalendar()

{

InitializeComponent();

}

private string GetMonthName(int month)

{

string[] months = {

"Январь", "Февраль",

"Март", "Апрель", "Май",

"Июнь", "Июль", "Август",

"Сентябрь", "Октябрь", "Ноябрь",

"Декабрь"

};

return months[month - 1];

}

private int DayOfWeekInRussianStyle

(DayOfWeek dayOfWeekInEnglishStyle)

{

const int FIRST\_INDEX = 0;

int numOfDayInWeek;

switch (dayOfWeekInEnglishStyle) {

case DayOfWeek.Monday:

numOfDayInWeek = FIRST\_INDEX + 1;

break;

case DayOfWeek.Tuesday:

numOfDayInWeek = FIRST\_INDEX + 2;

break;

case DayOfWeek.Wednesday:

numOfDayInWeek = FIRST\_INDEX + 3;

break;

case DayOfWeek.Thursday:

numOfDayInWeek = FIRST\_INDEX + 4;

break;

case DayOfWeek.Friday:

numOfDayInWeek = FIRST\_INDEX + 5;

break;

case DayOfWeek.Saturday:

numOfDayInWeek = FIRST\_INDEX + 6;

break;

default:

numOfDayInWeek = FIRST\_INDEX + 7;

break;

}

return numOfDayInWeek;

}

const string FONT\_OF\_NUM\_OF\_MONTH = "Veranda";

const int SIZE\_OF\_NUM\_OF\_MONTH = 20;

private void CreateNewButton(int numOfDayInWeek,

int daysCnt, int y) {

int x = (numOfDayInWeek - 1) \* (SIZE\_OF\_CELL + CELL\_DISTANCE) +

SIZE\_OF\_CELL / 2;

numbersOfMonth.Add(new DateButton());

numbersOfMonth.Last().Text = daysCnt.ToString();

numbersOfMonth.Last().Font = new Font(FONT\_OF\_NUM\_OF\_MONTH,

SIZE\_OF\_NUM\_OF\_MONTH);

numbersOfMonth.Last().Width = numbersOfMonth.Last().Height

SIZE\_OF\_CELL;

numbersOfMonth.Last().BackColor = Color.Orange;

numbersOfMonth.Last().FlatAppearance.BorderSize = 2;

numbersOfMonth.Last().FlatAppearance.BorderColor = Color.Orange;

numbersOfMonth.Last().FlatAppearance.MouseOverBackColor

Color.DarkOrange;

numbersOfMonth.Last().FlatStyle = FlatStyle.Flat;

numbersOfMonth.Last().Location = new Point(x, y);

pane.Controls.Add(numbersOfMonth.Last());

}

delegate void SomeFunc();

private void CreateCalendar(DateTime dateForDisplay, SomeFunc

InitializeNumbers)

{

lMonth.Text = GetMonthName(dateForDisplay.Month);

lYear.Text = dateForDisplay.Year.ToString();

int daysCnt = 1;

DateTime firstDayOfMonth = new DateTime(dateForDisplay.Year,

dateForDisplay.Month, 1, 0, 0, 0);

int numOfDayInWeek =

DayOfWeekInRussianStyle(firstDayOfMonth.DayOfWeek);

int y = START\_Y\_FOR\_MONTH\_TABLE;

int DaysInMonth = DateTime.DaysInMonth(dateForDisplay.Year,

dateForDisplay.Month);

while (daysCnt <= DaysInMonth) {

while ((numOfDayInWeek != (DAYS\_IN\_WEEK + 1)) && (daysCnt <=

DaysInMonth)) {

CreateNewButton(numOfDayInWeek, daysCnt, y);

numOfDayInWeek++;

daysCnt++;

}

y += SIZE\_OF\_CELL + CELL\_DISTANCE;

numOfDayInWeek = 1;

}

InitializeNumbers();

if ( (displayedDate.Month == DateTime.Now.Month) &&

(displayedDate.Year == DateTime.Now.Year) ) {

numbersOfMonth[displayedDate.Day - 1].BackColor = Color.Green;

}

for (int i = 0; i < numbersOfMonth.Count; i++) {

numbersOfMonth[i].Click += numbersOfMonth\_Click;

}

}

private void EmptyFunc() {}

private void InitializeToDoListOfDaysOfMonth() {

DateTime thisMonth = new DateTime(displayedDate.Year,

displayedDate.Month,

DateTime.DaysInMonth(displayedDate.Year,

displayedDate.Month));

for (int i = 0; i < allTasks.Count; i++) {

if (allTasks[i].date > thisMonth) {

break;

}

if ((allTasks[i].date.Month == displayedDate.Month) &&

(allTasks[i].date.Year == displayedDate.Year)) {

numbersOfMonth[allTasks[i].date.Day - 1].toDo =

allTasks[i].tasks;

numbersOfMonth[allTasks[i].date.Day –

1].FlatAppearance.BorderColor = Color.AliceBlue;

}

}

}

private void fCalendar\_Shown(object sender, EventArgs e)

{

if (File.Exists(TASKS\_FILE\_LOCATION)) {

using (StreamReader taskFile =

File.OpenText(TASKS\_FILE\_LOCATION)) {

while (!taskFile.EndOfStream) {

allTasks.Add(new TtoDo());

allTasks.Last().date =

DateTime.Parse(taskFile.ReadLine());

allTasks.Last().tasks = new List<string>();

string tmp = taskFile.ReadLine();

while (tmp != "") {

allTasks.Last().tasks.Add(tmp);

if (taskFile.EndOfStream) {

tmp = "";

}

else {

tmp = taskFile.ReadLine();

}

}

}

}

}

displayedDate = DateTime.Now;

if (allTasks.Count == 0) {

CreateCalendar(displayedDate, EmptyFunc);

}

else {

CreateCalendar(displayedDate,

InitializeToDoListOfDaysOfMonth);

}

LabelOfNameOfDaysOfWeekLocation();

if (displayedDate.AddMonths(-1).Year != DateTime.Now.Year) {

bPrevMonth.Enabled = false;

}

else {

bPrevMonth.Enabled = true;

}

}

const int OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK = 5;

private void LabelOfNameOfDaysOfWeekLocation() {

for (int i = 0; i < DAYS\_IN\_WEEK; i++) {

posOfNameOfDayOfWeek[i] = i \* (SIZE\_OF\_CELL + CELL\_DISTANCE)

+ 3 \* SIZE\_OF\_CELL / 4 + pane.Location.X ;

}

lMon.Location = new Point(posOfNameOfDayOfWeek[0],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

lTue.Location = new Point(posOfNameOfDayOfWeek[1],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

lWen.Location = new Point(posOfNameOfDayOfWeek[2],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

lThu.Location = new Point(posOfNameOfDayOfWeek[3],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

lFri.Location = new Point(posOfNameOfDayOfWeek[4],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

lSat.Location = new Point(posOfNameOfDayOfWeek[5],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

lSun.Location = new Point(posOfNameOfDayOfWeek[6],

lMonth.Location.Y + OFFSET\_Y\_FOR\_NAME\_OF\_DAYS\_OF\_WEEK);

}

private void bBack\_Click(object sender, EventArgs e) {

using (StreamWriter tmpFile =

File.CreateText(TASKS\_FILE\_LOCATION)) {

for (int i = 0; i < allTasks.Count; i++) {

tmpFile.WriteLine(allTasks[i].date.ToString());

for (int j = 0; j < allTasks[i].tasks.Count; j++) {

tmpFile.WriteLine(allTasks[i].tasks[j]);

}

tmpFile.WriteLine();

}

}

Form fMainMenu = Application.OpenForms[0];

fMainMenu.StartPosition = FormStartPosition.Manual;

fMainMenu.Left = this.Left;

fMainMenu.Top = this.Top;

//устанавливаем стартовую позицию родительской формы

fCalendar.ActiveForm.Close();

//закрываем форму календарь

fMainMenu.Show();

//отображаем родительскую форму

}

const int OFFSET\_OF\_WINDOW\_WITH\_TO\_DO\_LIST\_X = 10;

const int OFFSET\_OF\_WINDOW\_WITH\_TO\_DO\_LIST\_Y = 15;

private void numbersOfMonth\_Click(object sender, EventArgs e) {

if (!childWinIsOpen) {

this.Enabled = false;

lastClicked = int.Parse((sender as Button).Text);

fToDoDayList fToDoDayList = new fToDoDayList(

new DateTime(displayedDate.Year, displayedDate.Month,

int.Parse((sender as Button).Text)),

int.Parse((sender as Button).Text),

(sender as DateButton).toDo

);

fToDoDayList.Show();

fToDoDayList.Left = this.Left + (this.Width –

fToDoDayList.Width) - OFFSET\_OF\_WINDOW\_WITH\_TO\_DO\_LIST\_X;

fToDoDayList.Top = this.Top +

OFFSET\_OF\_WINDOW\_WITH\_TO\_DO\_LIST\_Y;

}

}

private void bNextMonth\_Click(object sender, EventArgs e) {

pane.Controls.Clear();

numbersOfMonth.Clear();

displayedDate = displayedDate.AddMonths(1);

if (allTasks.Count == 0) {

CreateCalendar(displayedDate, EmptyFunc);

}

else {

CreateCalendar(displayedDate,

InitializeToDoListOfDaysOfMonth);

}

if (displayedDate.AddMonths(-1).Year < DateTime.Now.Year) {

bPrevMonth.Enabled = false;

}

else {

bPrevMonth.Enabled = true;

}

}

private void bPrevMonth\_Click(object sender, EventArgs e) {

pane.Controls.Clear();

numbersOfMonth.Clear();

displayedDate = displayedDate.AddMonths(-1);

if (allTasks.Count == 0) {

CreateCalendar(displayedDate, EmptyFunc);

}

else {

CreateCalendar(displayedDate,

InitializeToDoListOfDaysOfMonth);

}

if (displayedDate.AddMonths(-1).Year < DateTime.Now.Year) {

bPrevMonth.Enabled = false;

}

else {

bPrevMonth.Enabled = true;

}

}

private void AddToListOfTasks(DateButton checkedNum) {

allTasks.Add(new TtoDo());

allTasks.Last().date = new DateTime(displayedDate.Year,

displayedDate.Month, int.Parse(checkedNum.Text));

allTasks.Last().tasks = checkedNum.toDo;

}

private void SortListOfTastks() {

for (int i = 0; i < allTasks.Count; i++)

for (int j = 0; j < allTasks.Count - 1; j++) {

if (allTasks[j].date > allTasks[j + 1].date) {

TtoDo tmp = allTasks[j];

allTasks[j] = allTasks[j + 1];

allTasks[j + 1] = tmp;

}

}

}

private void DeleteFromListOfTasks(DateTime findDate) {

for (int i = 0; i < allTasks.Count; i++) {

if (allTasks[i].date == findDate) {

allTasks.RemoveAt(i);

break;

}

}

}

private void FindAndCorrectInListOfTasks(DateButton checkedNum) {

DateTime nesDate = new DateTime(displayedDate.Year,

displayedDate.Month, int.Parse(checkedNum.Text));

for (int i = 0; i < allTasks.Count; i++) {

if (allTasks[i].date == nesDate) {

allTasks[i].tasks = checkedNum.toDo;

}

}

}

private void fCalendar\_EnabledChanged(object sender, EventArgs e) {

if (this.Enabled) {

using (StreamReader tmpFile = File.OpenText("tmpFile.txt")) {

numbersOfMonth[lastClicked - 1].toDo.Clear();

while (!tmpFile.EndOfStream) {

numbersOfMonth[lastClicked –

1].toDo.Add(tmpFile.ReadLine());

}

if (numbersOfMonth[lastClicked - 1].toDo.Count == 0) {

if (numbersOfMonth[lastClicked –

1].FlatAppearance.BorderColor == Color.AliceBlue) {

DeleteFromListOfTasks(new

DateTime(displayedDate.Year, displayedDate.Month, int.Parse(numbersOfMonth[lastClicked - 1].Text)));

}

numbersOfMonth[lastClicked –

1].FlatAppearance.BorderColor = Color.Orange;

}

else {

if (numbersOfMonth[lastClicked –

1].FlatAppearance.BorderColor == Color.Orange) {

AddToListOfTasks(numbersOfMonth[lastClicked –

1]);

}

else {

FindAndCorrectInListOfTasks(numbersOfMonth[lastClicked

- 1]);

}

numbersOfMonth[lastClicked –

1].FlatAppearance.BorderColor = Color.AliceBlue;

}

SortListOfTastks();

//чтение из файла

}

}

}

}

}

Окно fToDoDayList:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.IO;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ToDoDuck

{

public partial class fToDoDayList : Form

{

const int ADD\_HEIGHT = 25;

int currentNum;

public fToDoDayList(DateTime titleDate, int num, List<string> toDo) {

InitializeComponent();

currentNum = num;

lTitle.Text = titleDate.Day.ToString() + '/' +

titleDate.Month.ToString() + '/' +

titleDate.Year.ToString();

tbEnter.Text = "Новое дело...";//подсказка

tbEnter.ForeColor = Color.Brown;

if (toDo != null) {

for (int i = 0; i < toDo.Count; i++) {

lbToDoList.Items.Add(toDo[i]);

if (lbToDoList.Items.Count != 1) {

lbToDoList.Height = lbToDoList.Height + ADD\_HEIGHT;

lbToDoList.SetSelected(lbToDoList.Items.Count - 1,

true);

}

}

}

}

private void bClose\_Click(object sender, EventArgs e) {

Form fCalendar = Application.OpenForms[1];

List<string> tasks = new List<string>();

int i = 0;

foreach (object item in lbToDoList.Items) {

tasks.Add(item.ToString());

}

using (StreamWriter tmpFile = File.CreateText("tmpFile.txt")) {

if (tasks.Count > 0) {

for (i = 0; i < tasks.Count; i++) {

tmpFile.WriteLine(tasks[i]);

}

}

}

fCalendar.Enabled = true;

this.Close();

}

private void fToDoDayList\_Shown(object sender, EventArgs e) {

Graphics pane = panel.CreateGraphics();

Pen pen = new Pen(Color.Black, 5);

pane.DrawRectangle(pen, new Rectangle(0, 0, this.Width,

this.Height));

}

private void tbEnter\_KeyUp(object sender, KeyEventArgs e) {

if ((e.KeyCode == Keys.Enter) || (e.KeyCode == Keys.Return)) {

if (tbEnter.Text.Length > 0) {

//проверка, чтобы строка была непустой

string tmpS1 = tbEnter.Text;

tmpS1 = tmpS1.Trim(' ');

if (tmpS1.Length > 0) {

string tmpS2 = tbEnter.Text;

tmpS2 = tmpS2.TrimStart(' ');

lbToDoList.Items.Add(tmpS2);

//удаление пробелов вначале и добавление строки

tbEnter.Clear();

if (lbToDoList.Items.Count != 1) {

lbToDoList.Height = lbToDoList.Height +

ADD\_HEIGHT;

lbToDoList.SetSelected(lbToDoList.Items.Count –

1, true);

}

}

else

tbEnter.Clear();

}

}

}

private void lbToDoList\_MouseClick(object sender, MouseEventArgs e) {

if (lbToDoList.Items.Count > 0) {

toolTipAllStringInToDoList.SetToolTip(lbToDoList,

lbToDoList.SelectedItem.ToString());

}

}

private void tbEnter\_Enter(object sender, EventArgs e) {

tbEnter.Text = null;

tbEnter.ForeColor = Color.Black;

}

const int SUB\_HEIGHT = 15;

private void lbToDoList\_MouseDoubleClick(object sender,

MouseEventArgs e) {

if (lbToDoList.Items.Count > 0) {

lbToDoList.Items.RemoveAt(lbToDoList.SelectedIndex);

//удаление элемента

if (lbToDoList.Items.Count > 0) {

lbToDoList.Height = lbToDoList.Height - SUB\_HEIGHT;

lbToDoList.SetSelected(lbToDoList.Items.Count - 1, true);

//выравнивение ListBox

}

}

}

}

}

Окно fDesks:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.IO;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ToDoDuck

{

public partial class fDesks : Form

{

int openDeskIndex = -1;

const string NAME\_OF\_DIRECTOTY = "Desks Data";

List<TextBox> desks = new List<TextBox>();

const string FONT\_IN\_DESKS = "Veranda";

const int SIZE\_OF\_CHARS\_IN\_DESKS = 15;

const int POSITION\_OF\_DESK\_X = 12;

const int POSITION\_OF\_DESK\_Y = 12;

const int HEIGHT\_OF\_DESK = 460;

const int WIDTH\_OF\_DESK = 540;

private void DeskCreate() {

desks.Add(new TextBox());

//добавление новой доски в список досок

desks.Last().Font = new Font(FONT\_IN\_DESKS,

SIZE\_OF\_CHARS\_IN\_DESKS);

desks.Last().Multiline = true;

desks.Last().BackColor = Color.Orange;

desks.Last().BorderStyle = BorderStyle.FixedSingle;

desks.Last().Location = new Point(POSITION\_OF\_DESK\_X,

POSITION\_OF\_DESK\_Y);

desks.Last().Height = HEIGHT\_OF\_DESK;

desks.Last().Width = WIDTH\_OF\_DESK;

//задание параметров доски

this.Controls.Add(desks.Last());

//добавление доски на основную панель

}

public fDesks() {

InitializeComponent();

bNextDesk.Enabled = bPrevDesk.Enabled = false;

if (!Directory.Exists(NAME\_OF\_DIRECTOTY)) {

openDeskIndex = -1;

}

else {

int i = 0;

string deskFileLocation = NAME\_OF\_DIRECTOTY + "\\desk\_" +

i.ToString() + ".txt";

//путь к первому файлу

while (File.Exists(deskFileLocation)) {

//цикл выполняется до тех пор, пока есть нжные файлы

DeskCreate();

//метод, создающий доску

using (StreamReader desk =

File.OpenText(deskFileLocation)) {

desks.Last().Text = desk.ReadToEnd();

//чтение из файла

}

desks.Last().Visible = false;

i++;

openDeskIndex++;

deskFileLocation = NAME\_OF\_DIRECTOTY + "\\desk\_" +

i.ToString() + ".txt";

//переход к следующему файлу

}

DeleteDirectory(NAME\_OF\_DIRECTOTY);

if (desks.Count > 0) {

openDeskIndex = 0;

desks[0].Visible = true;

if (desks.Count > 1) {

bNextDesk.Enabled = true;

}

}

}

if (openDeskIndex < 0) {

bDeleteDesk.Enabled = false;

lYouHaventDesks.Visible = true;

}

else {

lYouHaventDesks.Visible = false;

bDeleteDesk.Enabled = true;

}

}

private void DeleteDirectory(string target\_dir) {

string[] files = Directory.GetFiles(target\_dir);

string[] dirs = Directory.GetDirectories(target\_dir);

foreach (string file in files) {

File.SetAttributes(file, FileAttributes.Normal);

File.Delete(file);

}

foreach (string dir in dirs) {

DeleteDirectory(dir);

}

Directory.Delete(target\_dir, false);

}

private void SaveDesksInFile() {

if (Directory.Exists(NAME\_OF\_DIRECTOTY)) {

DeleteDirectory(NAME\_OF\_DIRECTOTY);

//удаляем старую папку с досками

}

if (desks.Count > 0) {

Directory.CreateDirectory(NAME\_OF\_DIRECTOTY);

//создаем новую папку для досок

for (int i = 0; i < desks.Count; i++) {

using (StreamWriter desk =

File.CreateText(NAME\_OF\_DIRECTOTY + "\\desk\_" +

i.ToString() + ".txt")) {

desk.Write(desks[i].Text);

//записываем данные в файл

}

}

}

}

private void bBack\_Click(object sender, EventArgs e) {

if (desks.Count > 0) {

SaveDesksInFile();

}

Form fMainMenu = Application.OpenForms[0];

fMainMenu.StartPosition = FormStartPosition.Manual;

fMainMenu.Left = this.Left;

fMainMenu.Top = this.Top;

fDesks.ActiveForm.Close();

fMainMenu.Show();

}

private void bCreateDesk\_Click(object sender, EventArgs e) {

lYouHaventDesks.Visible = false;

if (desks.Count > 0) {

for (int i = 0; i < desks.Count; i++) {

desks[i].Visible = false;

}

}

DeskCreate();

openDeskIndex = desks.Count - 1;

bNextDesk.Enabled = false;

if (openDeskIndex != 0) {

bPrevDesk.Enabled = true;

}

bDeleteDesk.Enabled = true;

}

private void bDeleteDesk\_Click(object sender, EventArgs e) {

desks[openDeskIndex].Visible = false;

if (desks.Count - 1 > openDeskIndex) {

desks[openDeskIndex + 1].Visible = true;

desks.Remove(desks[openDeskIndex]);

}

else if (openDeskIndex > 0) {

desks[openDeskIndex - 1].Visible = true;

desks.Remove(desks[openDeskIndex]);

openDeskIndex--;

}

else {

desks.Remove(desks[openDeskIndex]);

lYouHaventDesks.Visible = true;

openDeskIndex = -1;

}

if (openDeskIndex == -1) {

bDeleteDesk.Enabled = false;

bNextDesk.Enabled = bPrevDesk.Enabled = false;

}

else if (desks.Count == 1) {

bNextDesk.Enabled = bPrevDesk.Enabled = false;

}

else if (openDeskIndex == desks.Count - 1) {

bNextDesk.Enabled = false;

}

else if (openDeskIndex == 0) {

bPrevDesk.Enabled = false;

}

}

private void bNextDesk\_Click(object sender, EventArgs e) {

bPrevDesk.Enabled = true;

desks[openDeskIndex].Visible = false;

openDeskIndex++;

desks[openDeskIndex].Visible = true;

if (openDeskIndex == desks.Count - 1) {

bNextDesk.Enabled = false;

}

}

private void bPrevDesk\_Click(object sender, EventArgs e) {

bNextDesk.Enabled = true;

desks[openDeskIndex].Visible = false;

openDeskIndex--;

desks[openDeskIndex].Visible = true;

if (openDeskIndex == 0) {

bPrevDesk.Enabled = false;

}

}

}

}

Окно fHelp:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ToDoDuck

{

public partial class fHelp : Form

{

int displayedPic = 0;

List<Bitmap> pic = new List<Bitmap>();

public fHelp() {

InitializeComponent();

for (int i = 0; i < 5; i++) {

pic.Add(new Bitmap("Help\\" + i.ToString() + ".png"));

}

pbPicture.Image = (Image) pic[0];

}

private void bBack\_Click(object sender, EventArgs e) {

Form fMainMenu = Application.OpenForms[0];

fMainMenu.StartPosition = FormStartPosition.Manual;

fMainMenu.Left = this.Left;

fMainMenu.Top = this.Top;

fDesks.ActiveForm.Close();

fMainMenu.Show();

}

private void bNextPic\_Click(object sender, EventArgs e) {

if (displayedPic != pic.Count - 1) {

pbPicture.Image = (Image)pic[++displayedPic];

}

bPrevPic.Enabled = true;

if (displayedPic == pic.Count - 1) {

bNextPic.Enabled = false;

}

}

private void bPrevPic\_Click(object sender, EventArgs e) {

if (displayedPic != 0) {

pbPicture.Image = (Image)pic[--displayedPic];

}

bNextPic.Enabled = true;

if (displayedPic == 0) {

bPrevPic.Enabled = false;

}

}

}

}