Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«Учёт семейного бюджета»

БГУИР КР 1-40 01 01 225 ПЗ

Студент: гр. 951002 Протасеня В.Н.

Руководитель:

асс. Фадеева Е.Е.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту   Протасене Валентине Николаевне

1. Тема работы   Учет семейного бюджета

2. Срок сдачи студентом законченной работы––01.06.2020 г.–––

3. Исходные данные к работе язык программирования делфи, динамический массив, файлы записей

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;

2. Анализ требований к программному средству и разработка функциональных требований;

3. Проектирование программного средства;

4. Создание (конструирование) программного средства;

5. Тестирование, проверка работоспособности и анализ полученных результатов;

6. Руководство по установке и использованию;

Список используемой литературы

Заключение

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. "Учета семейного бюджета", А1, схема программы, чертеж.

6. Консультант по курсовой работе

Фадеева Е.Е.

7. Дата выдачи задания 10.02.2020

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 01.03.2020 – 15 % готовности работы;

разделы 2, 3 к 15.03.2020 – 30 % готовности работы;

разделы 4, 5 к 15.04.2020 – 60 % готовности работы;

раздел 6 к 15.05.2020 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 20.05.2020 – 100 % готовности работы.

Защита курсовой работы с 01.06.2020 по 09.06.2020 г.–––––––––––––––––––––

РУКОВОДИТЕЛЬ–––––– Е.Е.Фадеева

(подпись)

Задание принял к исполнению 110.02.2020

(дата и подпись студента)

# **СОДЕРЖАНИЕ**

[СОДЕРЖАНИЕ 4](#_Toc42062711)

[1 АНАЛИЗ ПРОТОТИПОВ, ЛИЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМУ ПРОГРАММНОМУ СРЕДСТВУ 7](#_Toc42062712)

[1.1 Анализ существующих прототипов 7](#_Toc42062713)

[1.2 Постановка задачи 10](#_Toc42062714)

[2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАМНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 11](#_Toc42062715)

[2.1 Описание функциональности ПС 11](#_Toc42062716)

[2.2 Спецификация функциональных требований 12](#_Toc42062717)

[3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 14](#_Toc42062718)

[3.1 Проектирование динамических структур данных 14](#_Toc42062719)

[3.2 Разработка алгоритма добавления пользователя 14](#_Toc42062720)

[3.3 Разработка алгоритма удаления пользователя 15](#_Toc42062721)

[3.4 Разработка алгоритма добавление счетов пользователя 16](#_Toc42062722)

[3.5 Разработка алгоритма добавления расходов, доходов 17](#_Toc42062723)

[3.6 Разработка алгоритма фильтрации доходов, расходов по дате 18](#_Toc42062724)

[3.7 Разработка алгоритма удаления расходов, доходов 19](#_Toc42062725)

[3.8 Разработка алгоритма отрисовки круговой диаграммы расходов, доходов 20](#_Toc42062726)

[4 КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 21](#_Toc42062727)

[4.1 Взаимодействие между формами 21](#_Toc42062728)

[4.2 Структура модулей программы 25](#_Toc42062729)

[4.3 Описание модуля UnitMain 25](#_Toc42062730)

[4.4 Описание модуля UnitAddMoney 30](#_Toc42062731)

[4.5 Описание модуля UnitAddUserInfo 31](#_Toc42062732)

[4.6 Описание модуля UnitData 32](#_Toc42062733)

[5 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 35](#_Toc42062734)

[6 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ 42](#_Toc42062735)

[ЗАКЛЮЧЕНИЕ 48](#_Toc42062736)

[СПИСОК ЛИТЕРАТУРЫ 49](#_Toc42062737)

[ПРИЛОЖЕНИЕ 1 50](#_Toc42062738)

[ПРИЛОЖЕНИЕ 2 51](#_Toc42062739)

**ВВЕДЕНИЕ**

**Семейный бюджет** —это учет всех доходов и расходов семьи, который позволит проконтролировать, спрогнозировать траты и прибыль для достижения финансовых целей на любой временной период.

Бюджет состоит из статей доходов и расходов. Статьи дохода — это все источники доходов: зарплата, проценты с депозита, подработки и т.д. Статьи расходов — это все траты по категориям: еда, аренда квартиры, авто, транспорт, покупки, путешествия и т.д.

Основная задача при составлении бюджета семьи:

* контроль над финансовым положением;
* уменьшение незапланированных и ненужных расходов;
* осознанное планирование будущих покупок;
* формирование внутренней финансовой дисциплины;
* более эффективное и сознательно распределение средств.

Целью данной курсовой работы является разработка программного средства, которое поможет вести полный учет доходов и расходов, планировать затраты, составлять подробные отчеты о состоянии бюджета и даже рисовать диаграммы.

Данная пояснительная записка содержит следующие основные разделы:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;
2. Анализ требований к программному средству и разработка функциональных требований;
3. Проектирование программного средства;
4. Создание (конструирование) программного средства;
5. Тестирование, проверка работоспособности и анализ полученных результатов;
6. Руководство по установке и использованию.
7. **АНАЛИЗ ПРОТОТИПОВ, ЛИЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМУ ПРОГРАММНОМУ СРЕДСТВУ**
   1. **Анализ** **существующих прототипов**

На данный момент существует множество разных форматов приложений для учета бюджета. Поэтому в качестве прототипов будут использованы самые известные и широко используемые программы.

**«Дребеденьги» —** простой инструмент для учёта финансов, который поможет контролировать расходы и вести учет доходов.

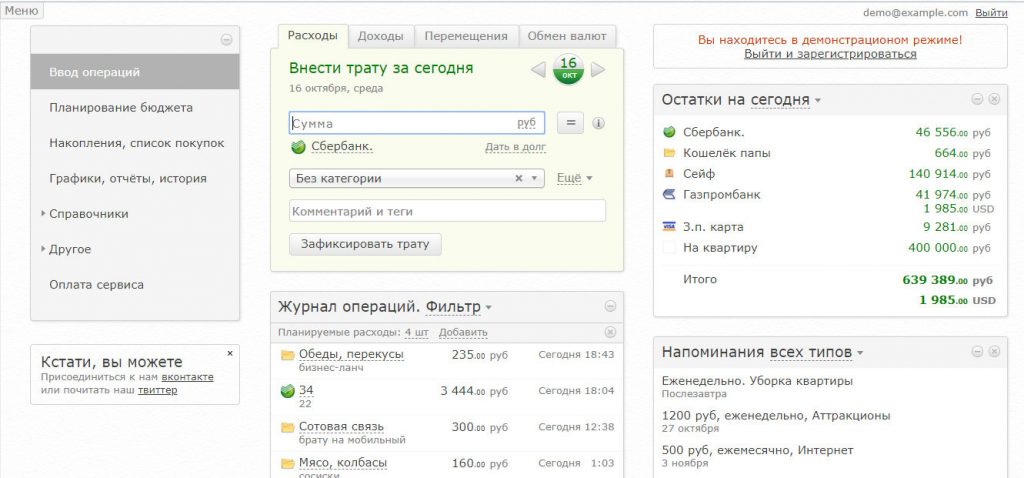


Рисунок 1.2 – Интерфейс «Дребеденьги»

Плюсы:

* неограниченное число пользователей и счетов;
* механизм учета заимствований и кредитов;
* возможность установки в качестве базовой валюты любой мировой финансовой единицы;
* резервное копирование данных;
* механизм напоминаний о важных платежах;
* возможность ограничить доступ к программе через пароль.

Минусы:

* низкая производительность;
* работа приложения осуществляется на платной основе.

**«Easy Finance»** — облачная платформа по учету финансов, интегрируемая в любой интернет-банк и сайт в интернете, позволяющая видеть конечному пользователю всю его финансовую информацию из разных источников в одном месте.

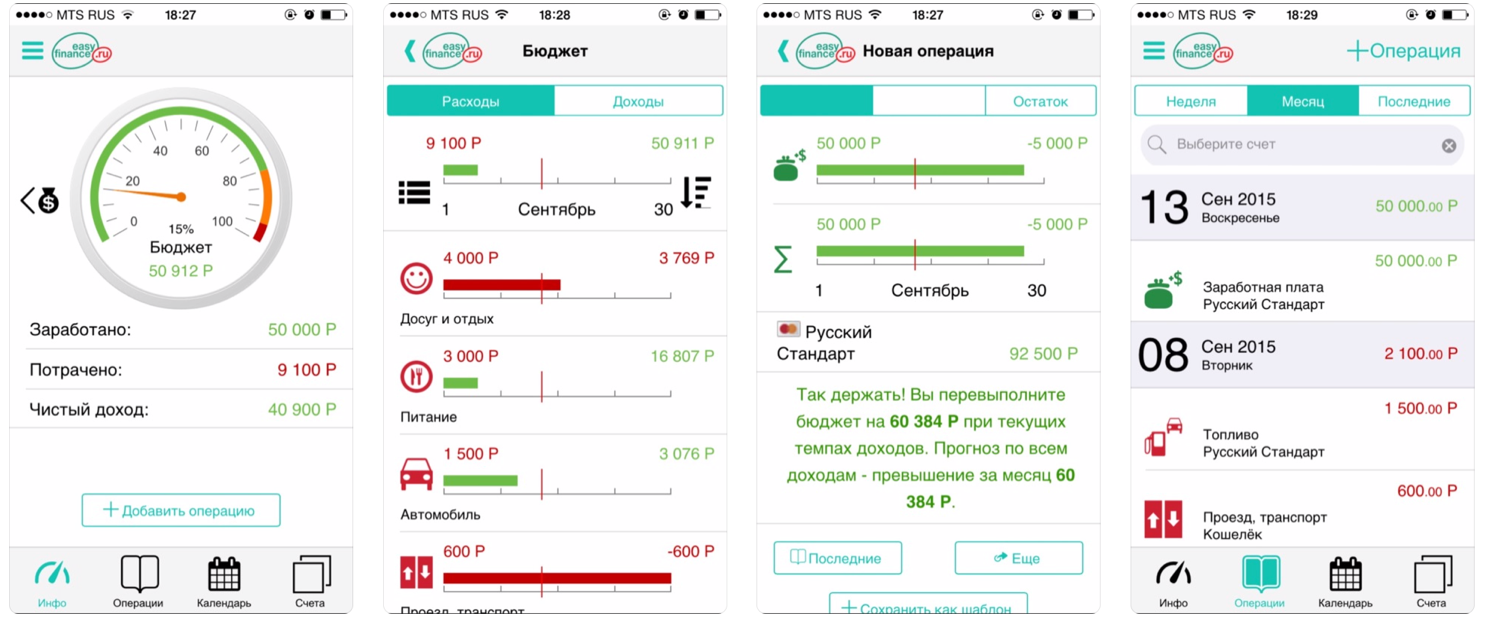


Рисунок 1.3 – Интерфейс «Easy Finance»

Плюсы:

* многофункциональность;
* не собирает личные данные;
* поддержка валют;
* постановка целей;
* синхронизация операций из банков и сайтов;
* распознавание SMS от банков;
* повторы операций;
* напоминания о платежах.

Минусы:

* бесплатное использование программы возможно только онлайн;
* высокая стоимость платной версии с полным функционалом;
* отсутствие личного кабинета и синхронизации данных.

**«Monefy»** — Простое и удобное приложение для ведения личного бюджета.



Рисунок 1.4 – Интерфейс «Monefy»

Плюсы:

* интуитивно понятный интерфейс;
* программа не требует регистрации, ввода личных данных, синхронизации с другими сервисами, сразу готова к работе;
* основные категории расходов предлагаются в понятных картинках;
* возможность добавлять новые категории;
* мультиплатформенность;
* темная тема.

Минусы:

* не хватка инструментов анализа;
* требуется доступ в интернет.

**Вывод**

Однозначного ответа на вопрос какое из существующих приложений самое лучшее с точки зрения интерфейса, быстроты работы и функциональных возможностей нет. Установлено, что при использовании программ на мощных компьютерах разница в быстродействии практически незаметна. Каждое из протестированных приложений имеет свои неоспоримые плюсы и минусы, по-своему удобно и найдет своих почитателей.

## Постановка задачи

Для максимально удобного взаимодействия пользователя с проектируемым программным средством должен быть создан информативный и интуитивно понятный пользовательский интерфейс. Также программное средство должно иметь достаточный функционал для обеспечения удобной работы.

В проектируемом программном средстве должны быть реализованы следующие функции:

* добавление неограниченного числа пользователей;
* удаление пользователей;
* отображение пользователей;
* добавление счетов пользователей;
* поддержка валютных счетов;
* добавление расходов, доходов по категориям;
* удаление расходов, доходов;
* добавление планов расходов, доходов на месяц;
* просмотр расходов, доходов;
* просмотр планов расходов, доходов;
* отображение разницы между планируемыми и фактическими расходами, доходами;
* сортировка таблиц расходов, доходов;
* отчет о расходах, доходах за различный промежуток времени.

В качестве языка программирования выбран язык Delphi и среда разработки – Borland Delphi 10, так как они были хорошо усвоены в курсе предмета “Основы Алгоритмизации и Программирования”.

# 

# **АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАМНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

# **2.1 Описание функциональности ПС**

Программное средство позволяет добавлять, удалять, просматривать бюджет пользователей и планы по его распределению. Так же предоставлять статистику о расходах, доходах в графическом и текстовом виде.

**Функция добавления пользователей** будет реализована с помощью кнопки c надписью «Добавить пользователя» и формой ввода ФИО пользователя. Отображение списка всех пользователей будет осуществляться с помощью компонента «ListBox».

Так же данная форма будет использоваться и для реализации **функции добавления счетов пользователя**, в которой будут содержаться следующие пункты:

* **cчета:** наличные, кредитная карта, счета в банке и свой вариант, который пользователь может ввести в появляющемся поле.
* **валюта:** белорусский рубль, российский рубль и доллар;
* **курс валют:** курс российского рубля и доллара по отношению к белорусскому рублю;
* **кнопка «Добавить».**

Для реализации функции **добавление расходов, доходов по категориям** будет создана новая форма, которая будет открываться с помощью кнопки с надписью: «Добавить». На форме будут располагаться следующие пункты:

* **пользователь;**
* **категории расходов или источник доходов**. Это зависит от выбранного пользователем пункта меню: расходы или доходы;
* **счета:** отображение всех счетов для выбранного пользователя при условии их наличия.
* **валюта**: отображение валюты для выбранного счета;
* **дата:** возможность пользователю выбрать, дату записи расхода или дохода. По умолчанию будет отображаться текущая дата;
* **примечания;**
* **поле ввода суммы**;
* **кнопки «Добавить» и «Отмена».**

**Категории расходов будут содержать следующие пункты:** продукты, алкоголь, квартира, мобильная связь и интернет, ЖКХ, образование, кредит и долг, проезд, красота, здоровье, путешествия, автомобиль, другое.

**Источники доходов:** заработная плата, стипендия, пенсия, сдача в аренду недвижимости, взял в долг, доход от других источников.

Данная форма также будет открываться и при **добавлении планов расходов, доходов на месяц**. В этом случае на ней будут располагаться только названия категорий расходов (источников доходов), поле ввода.

После нажатия на кнопку «Добавить» введенные пользователем данные заносятся в таблицу в соответствии с выбранными пунктами. Если выбрана кнопка «Отмена», то данные не заносятся, и форма добавления закрывается.

**Функция удаления пользователей, доходов и расходов** будет реализована с помощью кнопки с надписью: «Удалить».

Так же пользователям будет предоставлена возможность сортировать расходы, доходы по различным пунктам таблицы и фильтрация по выбранной дате.

## Спецификация функциональных требований

Среди функциональных требований есть «Отчет о расходах, доходах за различный промежуток времени». Спецификация данной функции может иметь следующий вид:

* Доходы за день;
* Доходы за неделю;
* Расходы за текущий месяц;
* Расходы за прошлый месяц;
* Доходы за день;
* Доходы за неделю;
* Доходы за текущий месяц;
* Доходы за прошлый месяц;
* Доход минус расход.

Отображение отчета будет происходить при помощи таблицы с 3 колоннами:

1. цвет категории;
2. название категории;
3. сумма дохода или расхода за выбранный промежуток времени.



Рисунок 2.1 – пример таблицы отчета

Так же отчет будет изображаться в графическом виде (круговая диаграмма) на компоненте «PaintBox» c помощью обращения к канве - свойству Canvas компонента.

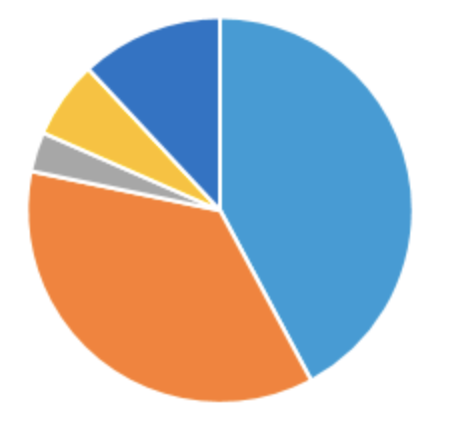


Рисунок 2.2 – пример круговой диаграммы

# **3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## 3.1 Проектирование динамических структур данных

При работе программного средства необходимо предоставлять возможность добавления, удаления пользователей и их данных. В качестве основной динамической структуры данных было принято решение использовать динамический массив пользователей.

ArrayOfUsers: array of TUser

Каждый элемент массива представляет собой запись типа TUser, в которой имеются поля, содержащие необходимые данные пользователя:

TUser = record

Name, Surname, Lastname: string [40];

MoneySpend: TMoneySpend;

MoneyEarn: TMoneyEarn;

Bill: array [0..9] of TBill;

Plan: Array[1..12,2000..2050] of TPlan;

end;

Подробное описание данной динамической структуры представлено в пункте 4.6.

Наполнение массива данными пользователей происходит при чтении файла соответствующего типа.

## 3.2 Разработка алгоритма добавления пользователя

Разработанное программное средство позволяет вводить неограниченное число пользователей. Для их добавления необходимо кликнуть на кнопку «Добавить пользователя», после чего появится новая форма «FrmAddUserInfo» для ввода фамилии, имени, отчества. После добавления форма закрывается, а введенные ФИО заносятся в компонент «ListBox», где отображаются все пользователи, и сохраняются в типизированный файл User’s Info.

При этом необходимо учитывать, что пользователь может некорректно ввести данные, отменить ввод или попытаться добавить уже существующего. Следовательно, надо производить проверку вводимой информации о пользователе. Схема алгоритма представлена на рисунке 3.1.

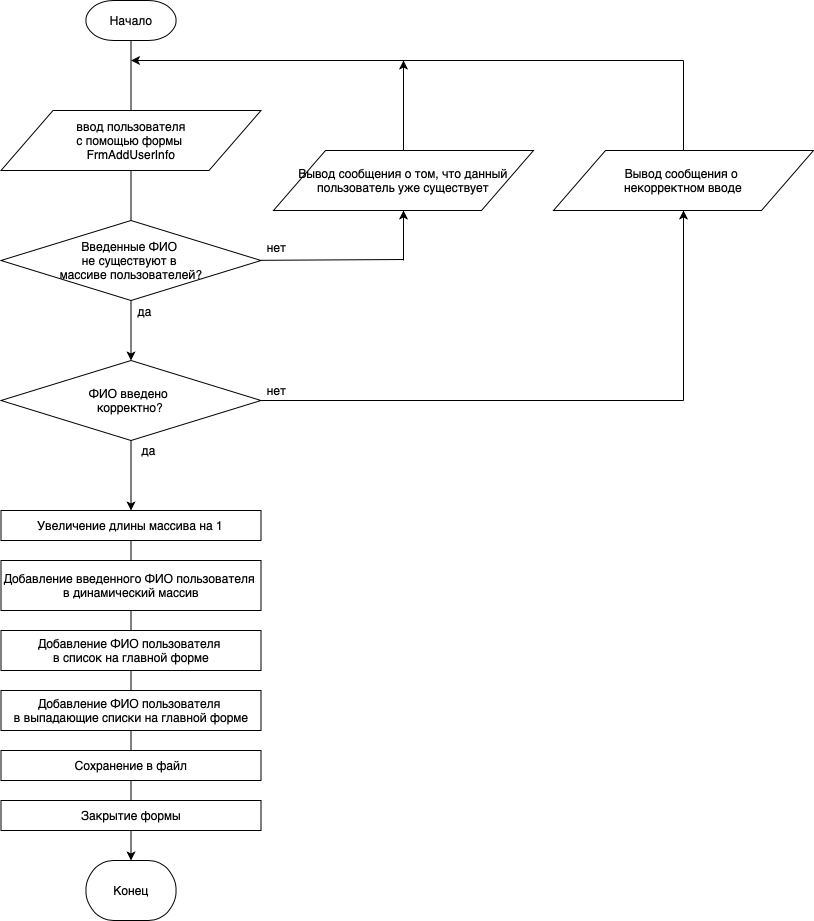


Рисунок 3.1 – схема алгоритма добавления пользователя

**3.3 Разработка алгоритма удаления пользователя**

Если возникнет необходимость удалить пользователя, а, следовательно, и все его данные из программы, то для этого надо выбрать конкретного пользователя из списка всех пользователей и нажать на кнопку с надписью «Удалить пользователя».

Если же в программе отсутствуют пользователи, то при попытке удаления кнопка будет неактивна. Схема реализации функции удаления пользователя представлен на рисунке 3.2.

я

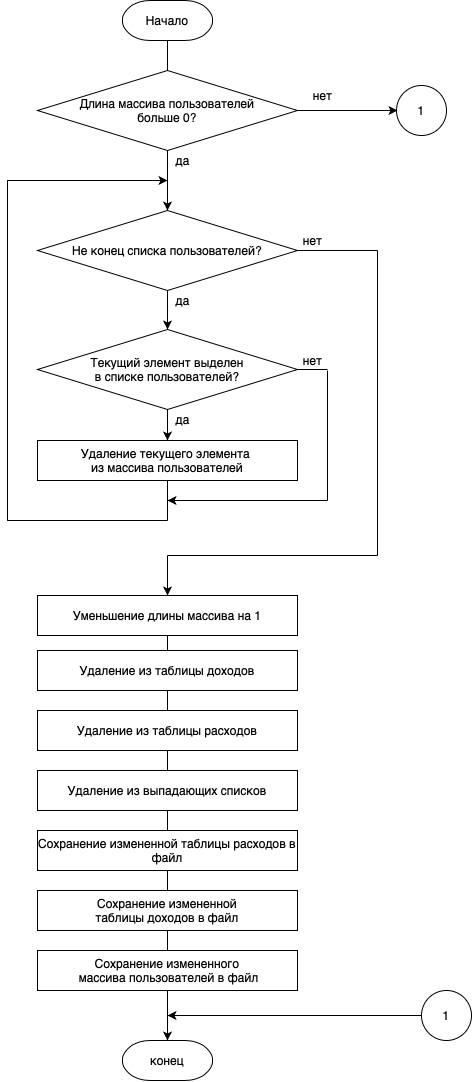
****

Рисунок 3.2 –схема алгоритма удаления пользователя

**3.4 Разработка алгоритма добавление счетов пользователя**

Еще одним важным пунктом для полноценного функционирования программы является добавление счетов. При выборе нужного пользователя и нажатии на кнопку «Добавить счет» произойдёт вызов той же формы FrmAddUserInfo, что и при добавлении ФИО. Но при этом форма будет содержать следующие пункты: валюта, курс валюты, счета и поле «Edit» для ввода своего вариант названия счета.

Так же при проектировании программного средства я предусмотрела попытку добавления счетов при отсутствии пользователей̆. В данном случае открытие формы добавления счета происходить не будет, а пользователь будет оповещен сообщением об отсутствии пользователей̆. Схема алгоритма представлена на рисунке 3.4.

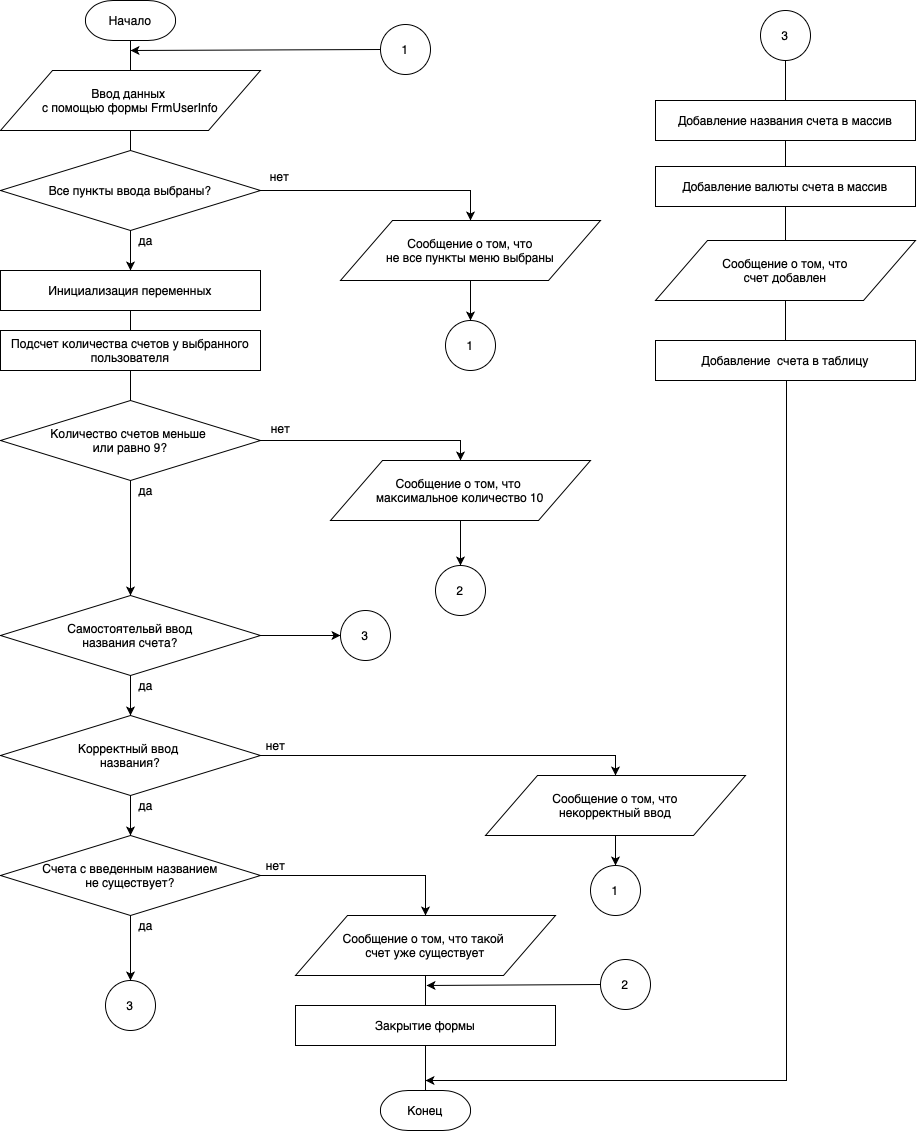


Рисунок 3.3 – схема алгоритма добавления счета пользователя

**3.5 Разработка алгоритма добавления расходов, доходов**

Для добавления платежей затрат или прибыли необходимо кликнуть на кнопку «Добавить», после чего произойдет вызов формы FrmAddMoney. Данные из заполненной формы заносятся в соответствующую таблицу.

При разработке так же учитывались возможные ошибки при заполнении формы. Схема алгоритма добавления расходов, доходов изображена на рисунке 3.4.



Рисунок 3.4 –схема алгоритма добавления расходов, доходов

**3.6 Разработка алгоритма фильтрации доходов, расходов по дате**

Пользователь может просмотреть доходы или расходы за определенный день. Для это необходимо в компоненте «DateTimePicker» выбрать дату и поставить флажок в компоненте «CheckBox»с надписью «Фильтровать».

При этом в таблице расходов или же доходов отобразятся траты или прибыль всех пользователей за выбранную дату. Схема алгоритма представлена на рисунке 3.5.

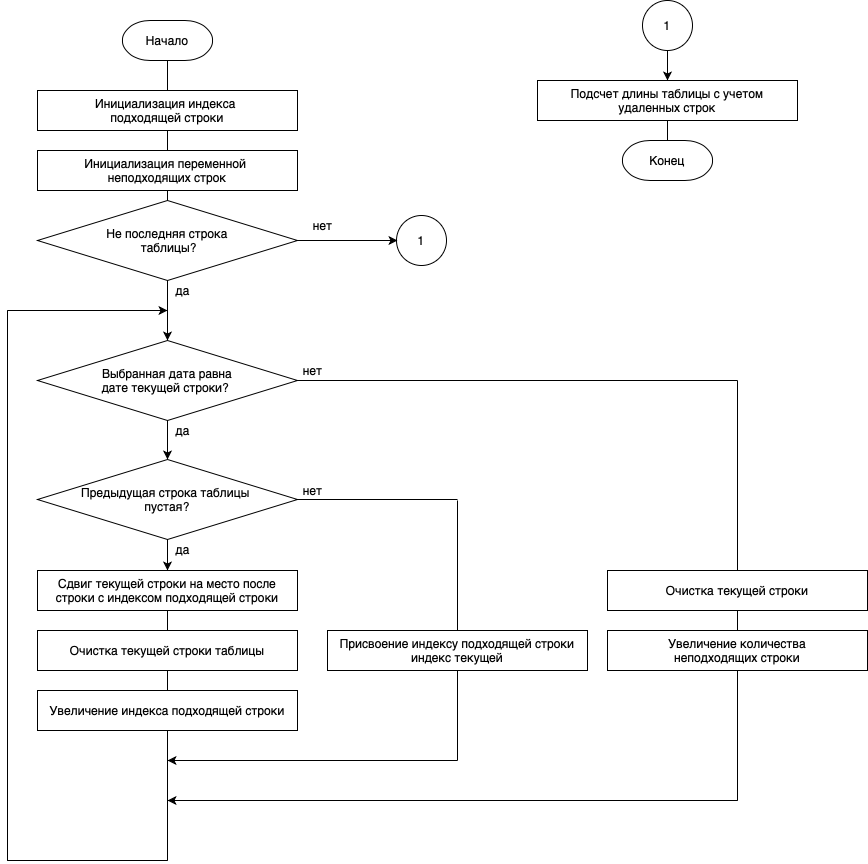
****

Рисунок 3.5 –схема алгоритма фильтрации расходов, доходов по дате

**3.7 Разработка алгоритма удаления расходов, доходов**

При ошибочном вводе платежа или желании удалить его из программы, мною предусмотрена реализация функции удаления расходов, доходов. При нажатии на кнопку с надписью «Удалить» выделенная строка будет удалена как из таблицы, так и из динамического массива пользователей.

Выбор, к какой таблице обращаться (доходов или расходов) и что удалять, осуществляется при помощи пользовательского меню и нажатии на кнопку «Доходы» или «Расходы». В случае отсутствия данных в таблицах кнопка будет неактивна. Схема реализации данной функции представлена на рисунке 3.6.

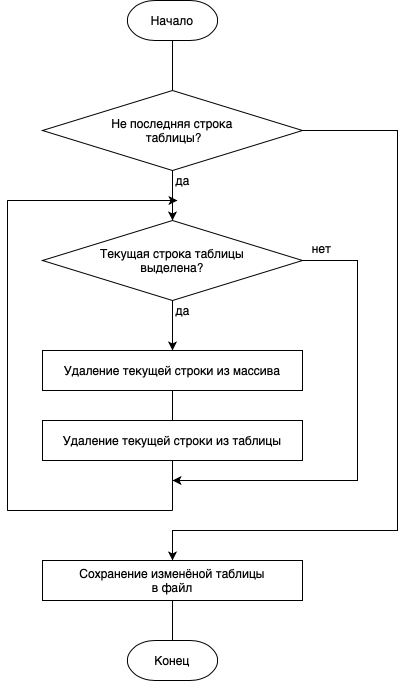


Рисунок 3.6 –схема алгоритма удаления расходов, доходов

## 3.8 Разработка алгоритма отрисовки круговой диаграммы расходов, доходов

Для более наглядного изображения отчета о бюджете пользователей была разработана функция отрисовки круговой диаграммы исходя из процентного соотношения суммы категории за выбранный промежуток времени к общей сумме. Схема алгоритма представлена на рисунке 3.7.

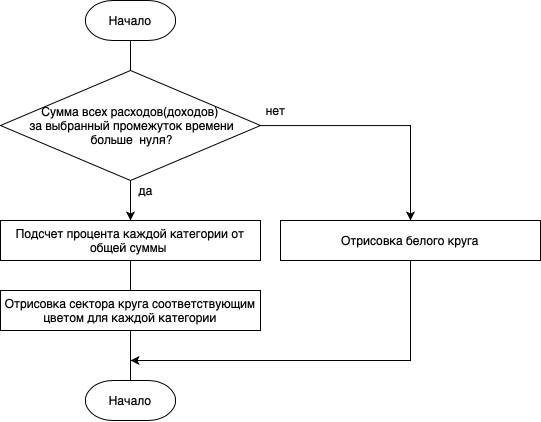
****

Рисунок 3.7 –схема алгоритма отрисовки диаграммы

# **КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## 4.1 Взаимодействие между формами

В программном средстве используется 4 форм:

* главная форм FrmMain (Все основные действия происходят на ней);
* форма добавления пользователя FrmAddUserInfo;
* форма ввод данных о расходах, доходах и планах FrmAddMoney;
* форма калькулятора FrmCalculator.

Схема взаимодействия форм представлена на рисунке 4.1.

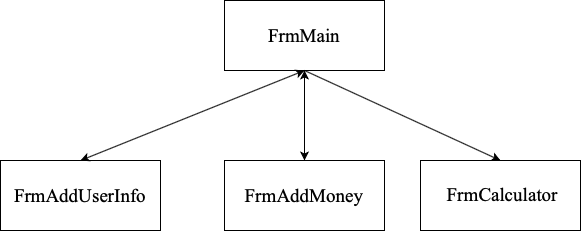


Рисунок 4.1 – схема взаимодействия форм

Форма FrmMain представляет собой главную форму. Основное взаимодействие пользователя осуществляется через нее благодаря пользовательскому меню. Панель основного меню (pnlMainMenu) располагается в левой части формы и состоит из 6 кнопок. Так же в верхней части формы присутствует еще одна панель верхнего меню (pnlTopMenu), на которой расположены дополнительные компоненты для взаимодействия с интерфейсом. Скриншот данной формы отображен на рисунке 4.2.

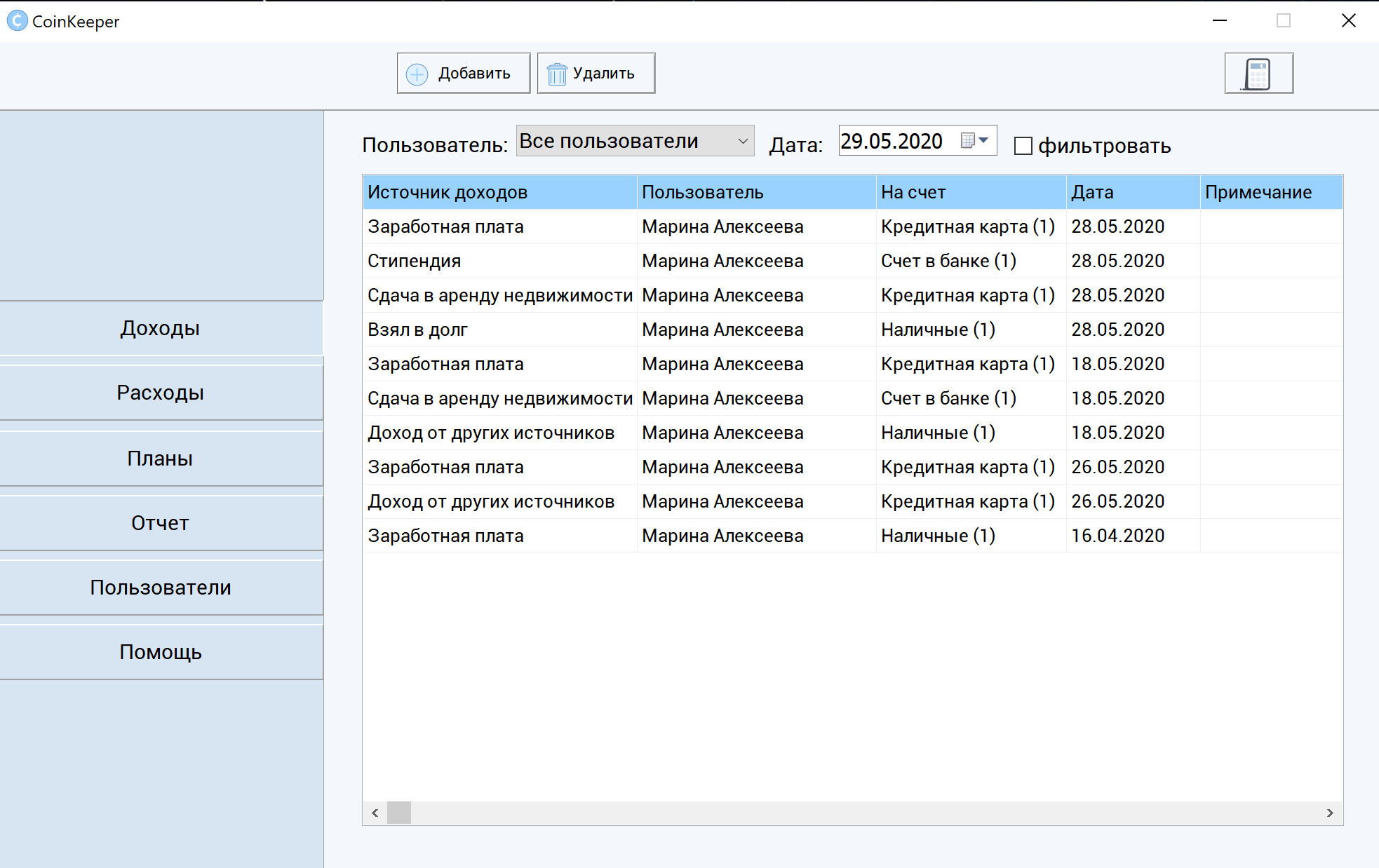


Рисунок 4.2 – скриншот формы FrmMain при запуске приложения

Открытие формы добавления пользователя FrmAddUserInfo происходит, при нажатии на кнопку «Добавить пользователя» или «Добавить счет» на главной форме. Наполненность формы компонентами зависит от выбранной пользователем кнопки. Скриншот данной формы представлен на рисунках 4.3-4.4.

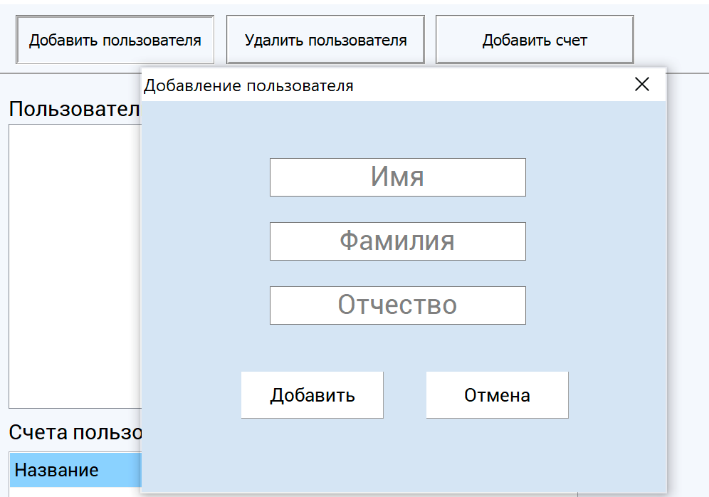


Рисунок 4.3 – скриншот формы FrmAddUserInfo при добавлении пользователя

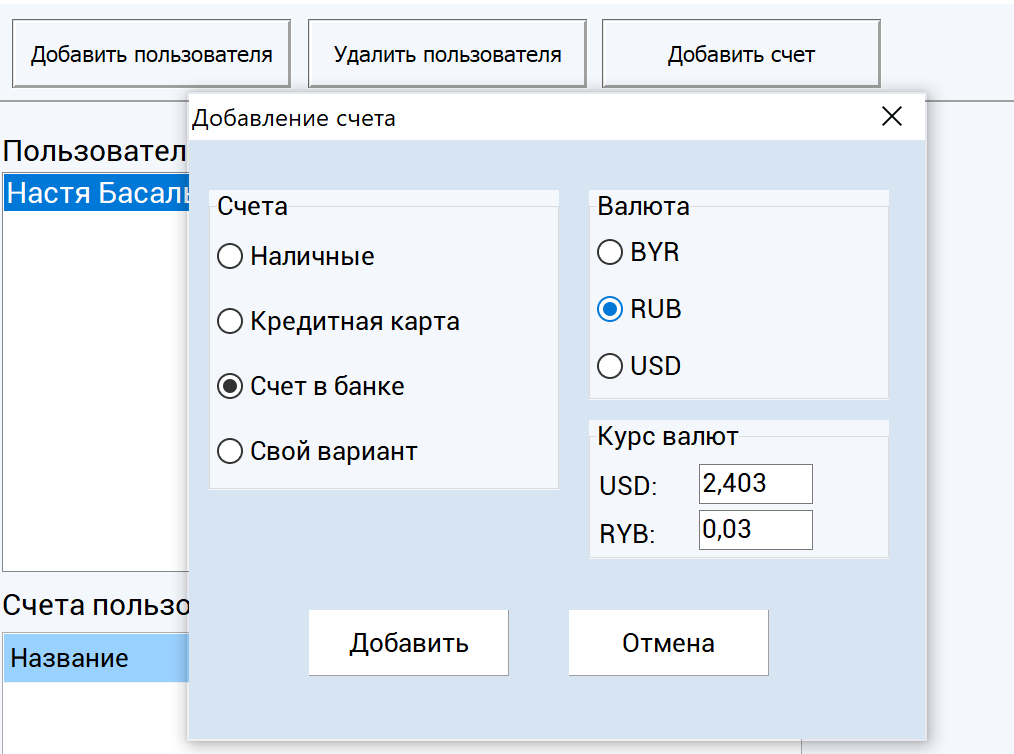


Рисунок 4.4 – скриншот формы FrmAddUserInfo при добавлении счета пользователя

Форма FrmAddMoney используется для ввода данных о расходах, доходах и планах на бюджет пользователей. Открытие формы происходит при нажатии на кнопки «Добавить», «Добавить план расхода», «Добавить план дохода». Наполненность формы компонентами и предоставляемые формой функции зависят от выбранной пользователем кнопки бокового и соответствующей верхнего меню. Схема открытия FrmAddMoney из главной формы и ее функции представлена на рисунке 4.5. Скриншот данной формы представлен на рисунках 4.6-4.8.

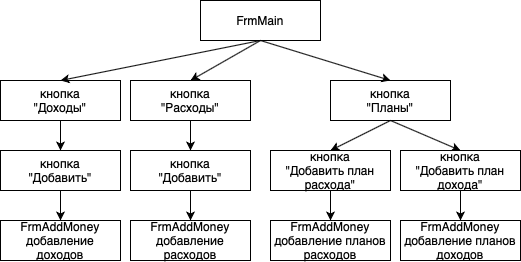


Рисунок 4.5 – схема открытия FrmAddMoney из главной формы FrmMain

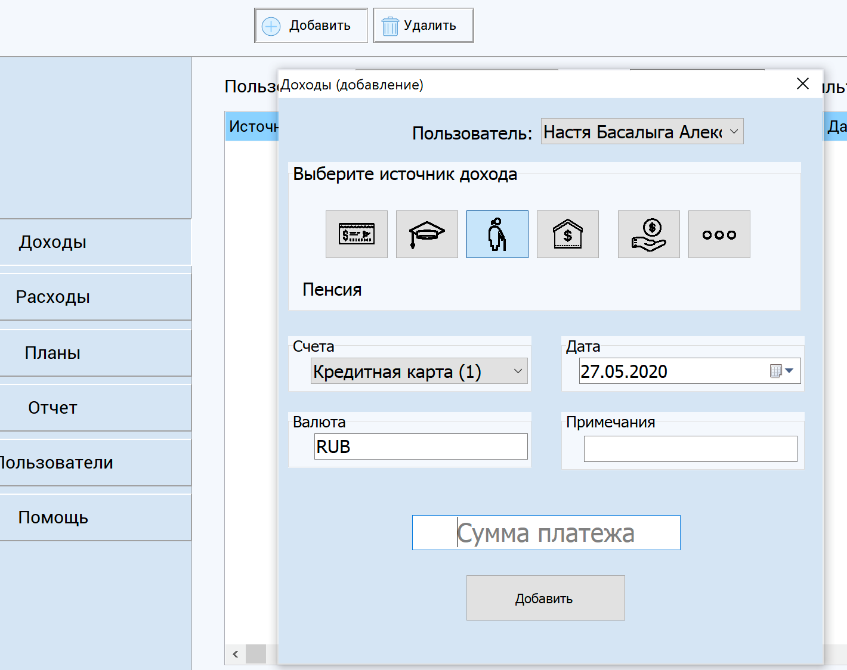


Рисунок 4.6 – скриншот формы FrmAddMoney при добавлении доходов

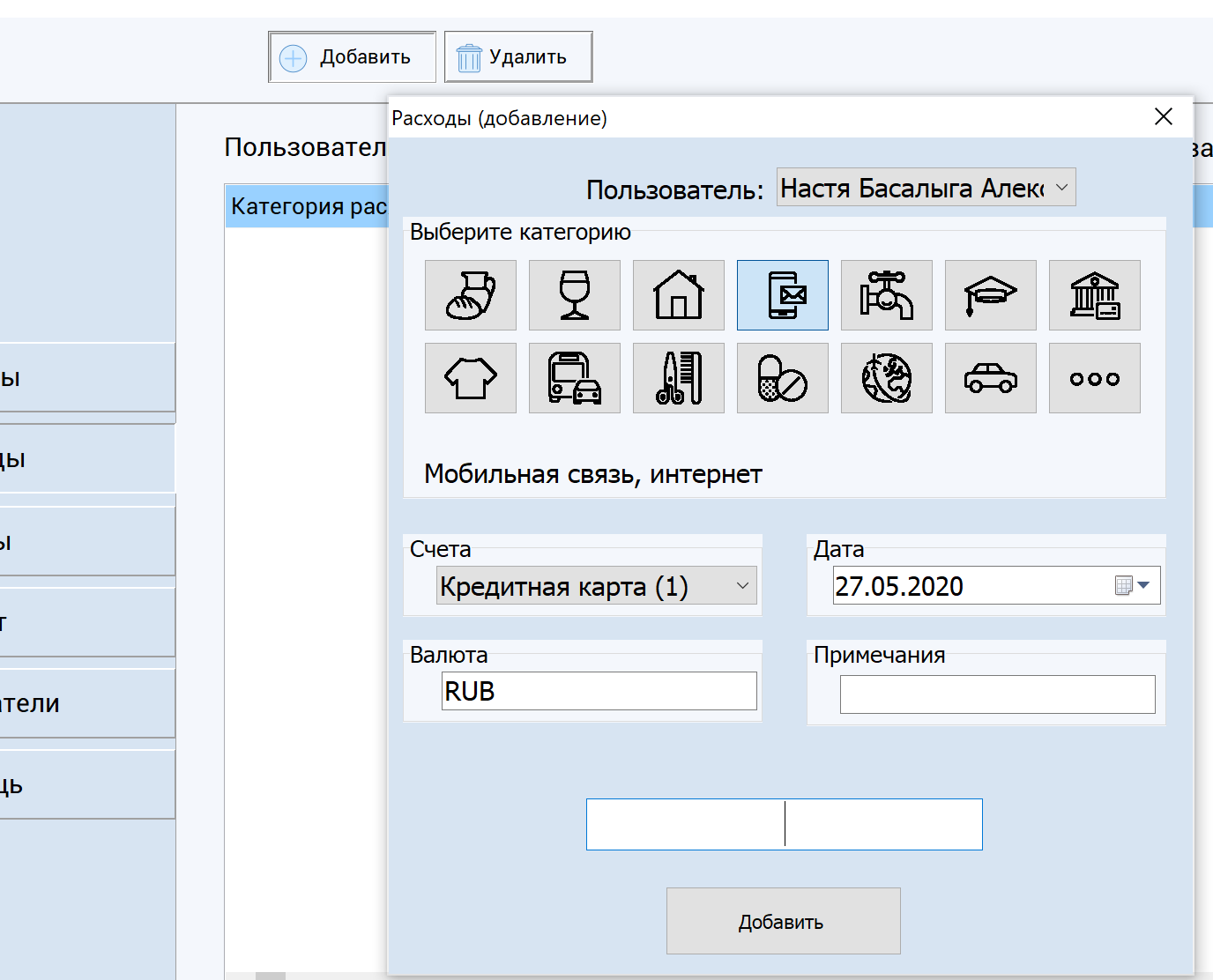


Рисунок 4.7 – скриншот формы FrmAddMoney при добавлении расходов

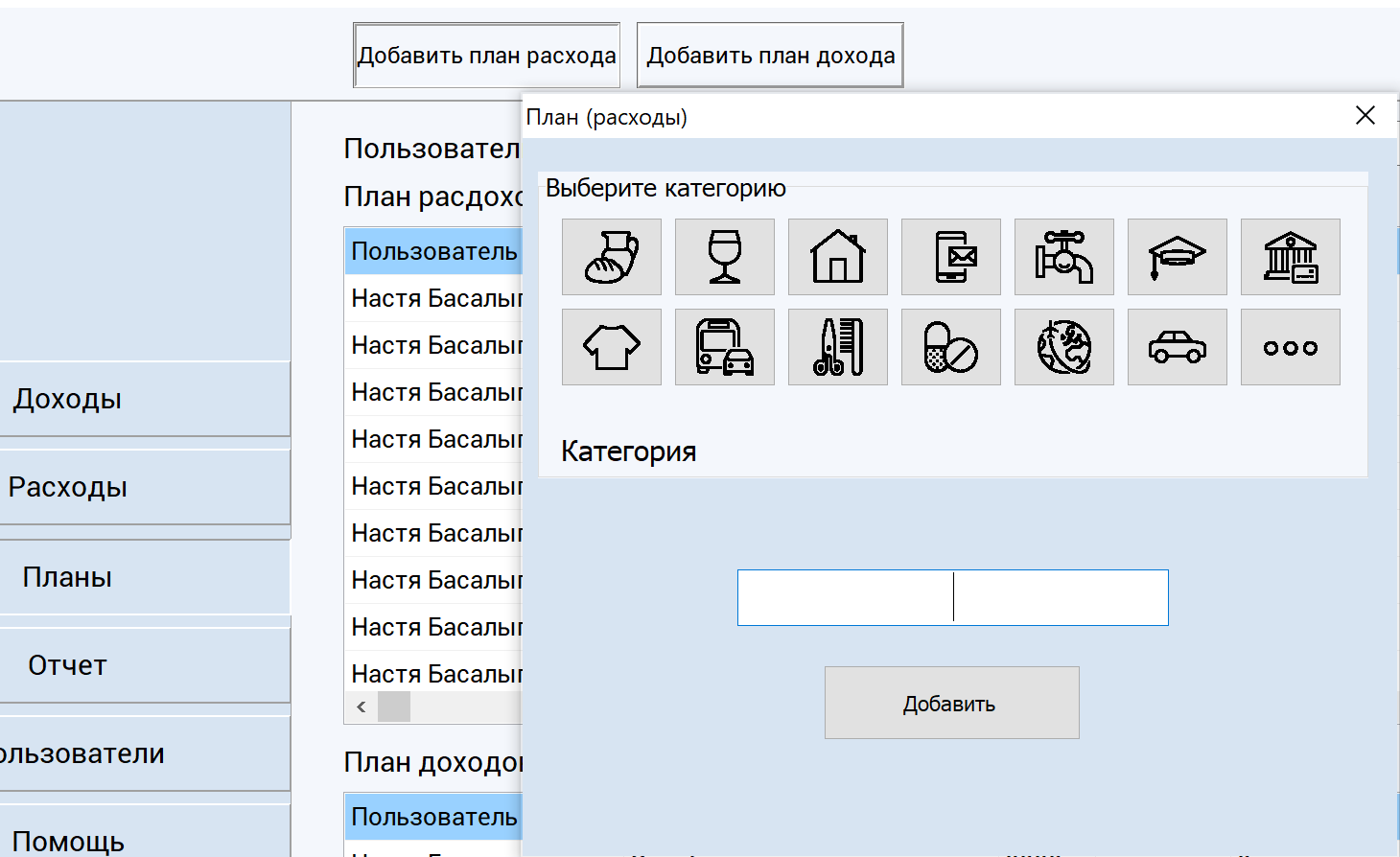


Рисунок 4.8 – скриншот формы FrmAddMoney при добавлении планов доходов

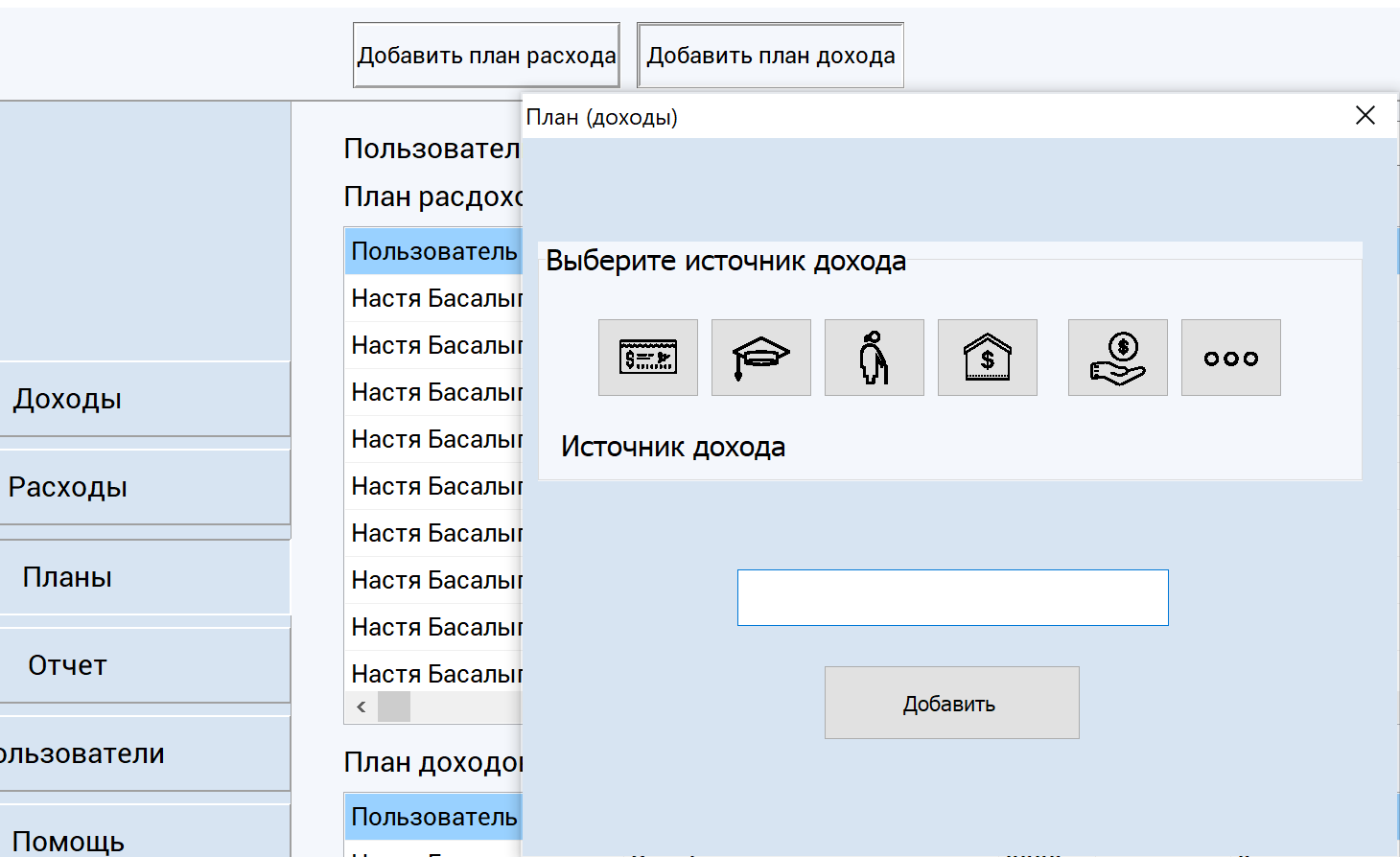


Рисунок 4.9 – скриншот формы FrmAddMoney при добавлении планов расходов

Форма калькулятора FrmCalculator открывается при нажатии на кнопку с картинкой калькулятора на главной форме. Предназначена для разнообразных подсчетов.

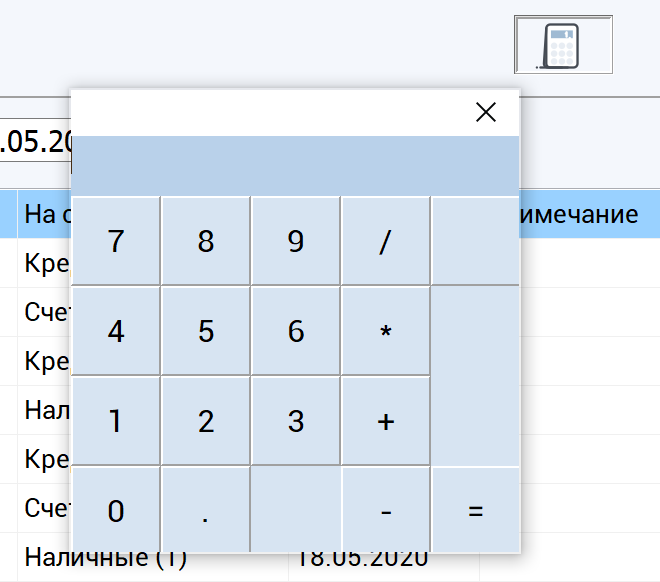


Рисунок 4.10 – скриншот формы FrmCalculator при добавлении планов расходов

Схема алгоритма работы всей программы представлена в приложении 1. Текст программы представлен в приложении 2.

## 4.2 Структура модулей программы

В процессе разработки программного средства мною было выделено 5 модулей.

Модули форм:

* UnitMain – модуль главной формы FrmMain;
* UnitAddMoney – модуль формы FrmAddMoney;
* UnitAddUserInfo– модуль формы FrmAddUserInfo;
* UnitCalculator – модуль формы FrmCalculator.

Модуль с основными структурами данных, константами и переменными:

* UnitData.

## 4.3 Описание модуля UnitMain

Модуль UnitMain является главным модулем программного средства. Описание основных подпрограмм, описанных в модуле UnitMain приведено в таблице 4.1.

Таблица 4.1. – основные подпрограммы модуля UnitMain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| FormCreate | Создание интерфейса программного средства | procedure TFrmMain.FormCreate(Sender: Tobject); | Sender | Объект, который сгенерировал событие |
| LoadUsers | Загрузка данных пользователей из файла в массив пользователей | procedure LoadUsers; | — | — |
| LoadUsersBill | Процедура загрузки счетов пользователей в таблицу | procedure LoadUsersBills(StringGrid: TstringGrid); | StringGrid | Таблица счетов |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SaveChanges | Сохранение  массива пользователей в файл | procedure SaveChanges; | — | — |
| MoneySpendVisible | Изменение видимости компонентов пункта меню раходы | procedure TfrmMain. MoneySpendVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |
| UsersVisible | Изменение видимости компонентов пункта меню пользователи | procedure TfrmMain. UsersVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |
| MoneyEarnVisible | Изменение видимости компонентов пункта меню доходы | procedure TfrmMain. MoneyEarnVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |
| HelpVisible | Изменение видимости компонентов пункта меню помощь | procedure TfrmMain. HelpVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |
| AboutVisible | Изменение видимости компонентов пункта помощь | procedure TfrmMain. AboutVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |
| ReportVisible | Изменение видимости компонентов пункта отчет | procedure TfrmMain. ReportVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PlansVisible | Изменение видимости компонентов пункта планы | procedure TfrmMain. PlansVisible(Value: Boolean); | Value | Переключатель  видимости компонентов |
| FilterGridForDate | Фильтрация таблиц по дате | procedure TfrmMain. SortGridForDate(StringGrid: TStringGrid); | StringGrid | Таблица, которую надо фильтровать |
| IsCellSelected | Проверка выделенной ячейки таблицы | function TfrmMain. IsCellSelected(StringGrid: TStringGrid; X, Y: Longint): Boolean; | StringGrid | Таблица |
| X | Координата ячейки |
| Y | Координата ячейки |
| DelRow | Удаление строки таблицы | procedure DelRow(StringGrid: TStringGrid); | StringGrid | Таблица, в которой надо удалить строку |
| SortGrid | Сортировка таблицы по выбранному  столбцу | procedure TfrmMain.SortGrid(SrtingGrid: TStringGrid; Column: Integer; | StringGrid | Таблицы |
| Column | Столбец |
| SortGridForUser | Сортировка таблицы по пользователям | procedure TfrmMain.SortGridForUser(StringGrid: TStringGrid); | StringGrid | Таблица |
| DeletUserFromTable | Удаление пользователя из таблицы | procedure TfrmMain.DeletUserFromTable(StringGrid: TStringGrid; | StringGrid | Таблица |
| SearchMoneyForToday | Поиск расходов ( доходов) за день | function TfrmMain.SearchMoneyForToday(StringGrid: TStringGrid;  UserIndex: Integer; Category: string): Integer; | StringGrid | Таблица |
| UserIndex | Индекс искомого пользователя |
| Category | Категория для поиска |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SearchMoneyForWeek | Поиск расходов ( доходов) за неделю | function TfrmMain.SearchMoneyForWeek(StringGrid: TStringGrid;  UserIndex: Integer; Category: string): Integer; | StringGrid | Таблица |
| UserIndex | Индекс искомого пользователя |
| Category | Категория для поиска |
| SearchMoneyForMonth | Поиск расходов ( доходов) за месяц | function TfrmMain.SearchMoneyForMonth(var StringGrid: TStringGrid;  UserIndex: Integer; Category: string; MyDate: TDateTime): Integer; | StringGrid | Таблица |
| UserIndex | Индекс искомого пользователя |
| Category | Категория для поиска |
| MyDate | Текущая дата |
| SumAllMoneyEarn | Подсчет суммы доходов за все время | function SumAllMoneyEarn(UserNumb: Integer): Integer; | UserNumb | Индекс пользоваетеля |
| SumAllMoneySpend | Подсчет суммы расходов за все время | function SumAllMoneySpend(UserNumb: Integer): Integer; | UserNumb | Индекс пользоваетеля |
| CountProcent | Подсчет процента суммы категории от общей суммы | function CountProcent(var AllMoney: Integer; Category: Integer): real; | AllMoney | Общая сумма |
| Category | Название категории |
| sector | Отрисовка одного сектора | procedure TfrmMain.sector(clr: TColor; angle1, angle2: real); | clr | Цвет |
| angle1 | Начальное положение |
| angle2 | Конечное положение |
| DeleteMoneyEarnFromArray | Удаление доходов из массива | procedure TFrmMain.DeleteMoneyEarnFromArray( i: integer); | i | Индекс текущей строки |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DeleteMoneySpendFromArray | Удаление расходов из массива | procedure TFrmMain.DeleteMoneySpendFromArray(i: integer); | i | Индекс текущей строки |
| DrawAllMoneySpendForAllTime | Отрисовка диаграммы расходов | procedure TfrmMain.DrawAllMoneySpendForAllTime(var AllMoney: Integer;  UserIndex: Integer; kind: Integer); | AllMoney | Сумма расходов |
| UserIndex | Индекс пользователя |
| kind: | Переменная  для выбора  варианта подсчета угла сектора |
| DrawAllMoneyEarnForAllTime | Отрисовка диаграммы доходов | procedure TfrmMain.DrawAllMoneyEarnForAllTime(var AllMoney: Integer;  UserIndex: Integer; kind: Integer); | AllMoney | Сумма расходов |
| UserIndex | Индекс пользователя |
| kind: | Переменная  для выбора  варианта подсчета угла сектора |
| SumAllForOneUsersBill | Подсчет общей суммы для одного счета | function TfrmMain.SumAllForOneUsersBill(StringGrid: TStringGrid;  BillName: string; UserName: string): Integer; | StringGrid | Таблица |
| BillName | Название счета |
| UserName | Имя пользователя |
| LoadFactPlan | Загрузка фактических расходов( доходов) в таблицы планов | procedure TfrmMain.LoadFactPlan(UserIndex: Integer); | UserIndex | Индекс пользователя |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CountPlanMinusFact | Подсчет разницы между фактическими и планируемыми расходами(доходами) | procedure TfrmMain.CountPlanMinusFact(StringGrid: TStringGrid); | StringGrid | Таблица планов |
| LoadStringGrid | Загрузка данных о бюджете из файла в таблицу | procedure LoadStringGrid(StringGrid: TStringGrid; sgNumber: Integer;  const FileName: TFileName); | StringGrid | Таблица |
| sgNumber | Номер таблицы |
| FileName | Название файла |
| SaveStringGrid | Сохранение данных из таблицы в файл | procedure SaveStringGrid(StringGrid: TStringGrid; const FileName: TFileName); | StringGrid | Таблица |
| FileName | Названия файла |

## 4.4 Описание модуля UnitAddMoney

Таблица 4.2. – основные подпрограммы модуля UnitAddMoney

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| SetFocusOn | Установка фокуса в поле ввода | procedure TfrmAddMoney.SetFocusOn; | — | — |
| NameOfCategorySet | Присвоение надписи категории расходов названия выбранной категории расходов | procedure TfrmAddMoney.NameOfCategorySet; | — | — |

Продолжение таблицы 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NameOfSourceSet | Присвоение надписи источника доходов названия выбранного источника  доходов | procedure TfrmAddMoney.NameOfSourceSet; | — | — |
| AddInfoToSpendTable | Добавление введенного расхода в таблицу расходов | procedure TfrmAddMoney.AddInfoToSpendTable( var BtnNumber: byte) ; | BtnNumber | Номер выбранной кнопки категории расходов |
| AddInfoTоEarnTable | Добавление введенного дохода в таблицу доходов | procedure TfrmAddMoney.AddInfoTоEarnTable( var BtnNumber: byte) ; | BtnNumber | Номер выбранной кнопки источника доходов |
| AddMoneySpendEarn | Ввод расходов (доходов) пользователем | procedure TfrmAddMoney.AddMoneySpendEarn; | — | — |
| AddPlanMoney | Ввод планируемых расходов (доходов) пользователем | procedure TfrmAddMoney.AddPlanMoney; | — | — |

## Описание модуля UnitAddUserInfo

Таблица 4.3. – основные подпрограммы модуля UnitAddUserInfo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| SetEditStyle | Задание цвета текста, надписи в поля ввода ФИО | procedure TfrmAddUserInfo.SetEditStyle; | — | — |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IsBillExist | Проверка наличия счета с таким именем | function TfrmAddUserInfo.IsBillExist ( BillName: string; j: integer): boolean; | BillName | Название счета |
| j | Индекс счета |
| AddUser | Добавление  пользователя | procedure TfrmAddUserInfo.AddUser; | — | — |
| AddBill | Добавление счета пользователя | procedure TfrmAddUserInfo.AddBill; | — | — |
| IsUserExit | Проверка наличия введенного пользователя в массиве пользователей | function TfrmAddUserInfo.IsUserExit(Name,Surname,Lastname: string): boolean; | Name | Имя |
| Surname | Фамилия |
| Lastname | Отчество |

## Описание модуля UnitData

Модуль UnitData содержит объявления структур данных (статических и динамических), констант и переменных. Ниже приведены основные структуры данных.

Таблица 4.4. – основные структуры данных

|  |  |  |
| --- | --- | --- |
| **Имя идентификатора структуры** | **Назначение структуры** | **Тип структуры** |
| sgCategorySpendPlan | Хранит названия категорий расходов | Массив строковых констант  sgCategorySpendPlan: array [1 .. 14] of string = ('Продукты', 'Алкоголь','Квартира', 'Мобильная связь, интернет', 'ЖКХ', 'Образование','Кредит, долг', 'Одежда', 'Проезд', 'Красота', 'Здоровье', 'Путешествия','Автомобиль', 'Другое'); |

Продолжение таблицы 4.4

|  |  |  |
| --- | --- | --- |
| sgCategoryEarnPlan | Храние названия категорий доходов | Массив строковых констант  sgCategoryEarnPlan: array [1 .. 6] of string = ('Заработная плата','Стипендия', 'Пенсия', 'Сдача в аренду недвижимости', 'Взял в долг','Доход от других источников'); |
| TMoneySpend | Хранение суммы расходов по категориям | Запись, где каждое поле — категория расхода |
| Products: integer; — Продукты |
| Alcohol: integer; — Алкоголь |
| Clothes: integer; — Одежда |
| Flat: integer; — Квартира |
| Health: integer; — Здоровье |
| Internet: integer; — Интернет |
| Beauty: integer; — Красота |
| DriveAway: integer; — Проезд |
| Education: integer; — Образование |
| Journey: integer; — Путешествия |
| JKH: integer; — ЖКХ |
| Car: integer; — Автомобиль |
| Credit: integer; — Кредит |
| Another: integer; —Доход от других источников |
| TMoneyEarn | Хранение суммы доходов по категориям | Запись, где каждое поле — источник дохода |
| ZP: integer; — Заработная плата |
| Grant: integer; — Стипендия |
| Rent: integer; — Сдача в аренду недвижимости |
| Debt: integer; — Взял в долг |
| Compensation: integer; — Пенсия |
| AnotherMoney: integer; —Доход от других источников |
| TBill | Хранение счетов пользователя | Запись |
| BillName: string [20]; — Название счета |
| Valuta: string [20] — Название валюты |

Продолжение таблицы 4.4

|  |  |  |
| --- | --- | --- |
| TPlan | Хранения планируемых расходов (доходов) по месяцам | Запись |
| Spend: TMoneySpend; —Планы расходов |
| Earn: TMoneyEarn; —Планы доходов |
| TUser | Данные пользователей | Запись |
| Name: string [40]; — Имя |
| Surname: string [40]; —Фамилия |
| Lastname: string [40]; —Отчество |
| MoneySpend: TMoneySpend; — Расходы |
| MoneyEarn: TMoneyEarn; — Доходы |
| Bill: array [0..9] of TBill — Статический массив счетов |
| Plan: Array[1..12,2000..2050] of TPlan; —Статический двумерный массив планов, где строки — 12 месяцев, столбцы— диапазон лет от 2000 до 2050 |
| ArrayOfUsers | Хранение данных пользователей | Динамический массив записей  ArrayOfUsers: array of TUser |
| UserInfoFile | Хранение данных пользователей и их загрузка из файла при запуске программы | Типизированный файл  UserInfoFile: file of TUser; |

# 

# **ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

Тестирование играет важную роль в разработке программного обеспечения и является неотъемлемой частью жизненного цикла его разработки, так как:

* повышает надежность, качество и производительность программного средства;
* помогает проверить, правильно ли работает программное средство, убедиться, что программа выполняет то, для чего оно предназначена;
* дает возможность сравнить реальное и ожидаемое поведение программы.

Данное тестирование проводилось на персональном компьютере с установленной операционной системой Windows 10.

Таблица 5.1 – Результаты функционального тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер**  **теста** | **Тестируемая функциональность** | **Последовательность действий** | **Ожидаемый результат** | **Полученный результат** |
| 0 | Добавление пользователя | 1.Кликнуть на кнопку «Пользователи»  2.Кликнуть на кнопку «Добавить пользователя»  3.В появившейся форме ввести корректное ФИО 4. Нажать «Добавить» | Введенное ФИО будет добавлено в список пользователей | Тест пройден |
| 1 |  | 1.Кликнуть на кнопку «Пользователи»  2.Кликнуть на появившуюся кнопку «Добавить пользователя»  3.В появившейся форме оставить одно или несколько полей ввода не заполненными  4.Нажать «Добавить» | Появится сообщение о том, что введено неверное значение | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Добавление счета пользователя | 1.Кликнуть на кнопку «Пользователи»  2.Выбрать пользователя из списка всех пользователей  3.Кликнуть на кнопку «Добавить счет»  4.В появившейся форме ввести данные счета  5.Нажать кнопку «Добавить» | Появится сообщение о том, что счет добавлен и введенные данные счета добавятся в таблицу счетов | Тест пройден |
| 3 | 1.Кликнуть на кнопку «Пользователи»  2.Выбрать пользователя из списка всех пользователей  3.Кликнуть на кнопку «Добавить счет»  4.В появившейся форме оставить незаполненным один или все пункты  5.Нажать кнопку «Добавить» | Появится сообщение о том, что не все пункты выбраны | Тест пройден |
| 4 | 1.Кликнуть на кнопку «Пользователи»  2.Кликнуть на кнопку «Добавить счет» при отсутствии пользователей | Появится сообщение о том, что список пользователей пуст | Тест пройден |
| 5 | 1.Кликнуть на кнопку «Пользователи»  2.Выбрать пользователя из списка всех пользователей, у которого в наличии уже 10 счетов  3.Кликнуть на кнопку «Добавить счет | Появится сообщение о том, что максимальное количество счетов 10 | Тест пройден |
| 6 | Удаление пользователя | 1.Кликнуть на кнопку «Пользователи»  2.Выбрать пользователя из списка всех пользователей  3.Кликнуть на кнопку «Удалить пользователя | Пользователь будет удален из списка всех пользователей | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | Добавление доходов | 1.Кликнуть на кнопку «Доходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого отсутствуют счета  4.Ввести данные дохода  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что необходимо добавить счет | Тест пройден |
| 8 | 1.Кликнуть на кнопку «Доходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого есть счета  4.Ввести данные дохода  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что платеж добавлен. Введенные данные дохода добавятся в таблицу доходов | Тест пройден |
| 9 | 1.Кликнуть на кнопку «Доходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого есть счета  4.Выбрать источник дохода, счет, но не ввести сумму  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что сумма платежа не введена | Тест пройден |
| 10 | 1.Кликнуть на кнопку «Доходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого есть счета  4.Ввести сумму дохода, но не выбрать категорию  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что категория не выбрана | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | Добавление расходов | 1.Кликнуть на кнопку «Расходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого отсутствуют счета  4.Ввести данные расхода  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что необходимо добавить счет | Тест пройден |
| 12 | 1.Кликнуть на кнопку «Расходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого есть счета  4.Ввести данные расхода  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что платеж добавлен. Введенные данные расхода добавятся в таблицу расходов | Тест пройден |
| 13 | Кликнуть на кнопку «Расходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого есть счета  4. Выбрать категорию, но не ввести сумму расхода  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что сумма платежа не введена | Тест пройден |
| 14 | 1.Кликнуть на кнопку «Расходы»  2.Кликнуть на кнопку «Добавить»  3.В появившейся форме выбрать пользователя, у которого есть счета  4.Ввести сумму расхода, но не выбрать категорию  5.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что категория не выбрана | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 | Добавление. расходов (доходов) при отсутствии пользователей | 1.Кликнуть на кнопку «Доходы»  2.Кликнуть на кнопку «Добавить» при отсутствии пользователей | Появится сообщение о том, список пользователей пуст | Тест пройден |
| 16 | Удаление расходов(доходов) | 1.Кликнуть на кнопку «Расходы» («Доходы»)  2.Кликнуть в таблице на строку с расходом(доходом), который хотим удалить  3.Кликнуть на кнопку «Удалить» | Строка удалиться из таблицы | Тест пройден |
| 17 | Отображение всех расходов (доходов) из таблицы расходов(доходов) для выбранного пользователя | 1.Кликнуть на кнопку «Расходы» («Доходы»)  2.Выбрать пользователя, данные расходов(доходов) которого хотим просмотреть | В таблице расходов(доходов) останутся данные выбранного пользователя | Тест пройден |
| 18 | Фильтрация таблицы расходов (доходов) для всех пользователей по выбранной дате | 1.Кликнуть на кнопку «Расходы» («Доходы»)  2.Выбрать дату, за которую хотим посмотреть расходы(доходы)  3.Поставить флажок в поле «Фильтровать» | В таблице расходов(доходов) останутся данные всех пользователей за выбранную дату | Тест пройден |
| 19 | Добавление планов расходов | 1.Кликнуть на кнопку «Планы»  2.Выбрать пользователя  3.Выбрать месяц и год  4.Кликнуть на кнопку «Добавить план расхода»  5.В появившуюся форму  вести данные  6.Кликнуть на кнопку «Добавить» | Введенные данные добавятся в таблицу планов расходов для выбранного пользователя | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | Добавление планов доходов | 1.Кликнуть на кнопку «Планы»  2.Выбрать пользователя  3.Выбрать месяц и год  4.Кликнуть на кнопку «Добавить план дохода»  5.В появившуюся форму  вести данные  6.Кликнуть на кнопку «Добавить» | Введенные данные добавятся в таблицу планов доходов для выбранного пользователя | Тест пройден |
| 21 | Некорректное  добавление планов расходов(доходов) | 1.Кликнуть на кнопку «Планы»  2.Выбрать пользователя  3.Выбрать месяц и год  4.Кликнуть на кнопку «Добавить план расхода» («Добавить план дохода»)  5.В появившейся форме выбрать категорию, но не ввести сумму плана  6.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что сумма плана не введена | Тест пройден |
| 1.Кликнуть на кнопку «Планы»  2.Выбрать пользователя  3.Выбрать месяц и год  4.Кликнуть на кнопку «Добавить план расхода» («Добавить план дохода»)  5.В появившейся форме ввести сумму плана, но не выбрать категорию  6.Кликнуть на кнопку «Добавить» | Появится сообщение о том, что категория не выбрана | Тест пройден |
| 22 | Отображение всех счетов пользователя и их остатка | 1.Кликнуть на «Пользователи»  2.Выбрать пользователя в списке всех пользователей | В таблице счетов отобразятся счета пользователя (если они есть) и остаток | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 | Отображение отчета доходов за выбранный промежуток времени для выбранного пользователя | 1.Кликнуть на кнопку «Отчет»  2.Выбрать пользователя  3.Выбрать промежуток времени | В таблицу отчета будут загружены доходы по категориям для выбранного пользователя и выбранного промежутка времени и нарисована соответствующая круговая диаграмма.  При этом суммы со всех счетов конвертируются и отображаются в белорусском рубле. | Тест пройден |
| 24 | Отображение отчета расходов за выбранный промежуток времени для выбранного пользователя | 1.Кликнуть на кнопку «Отчет»  2.Выбрать пользователя  3.Выбрать промежуток времени | В таблицу отчета будут загружены расходы по категориям для выбранного пользователя и выбранного промежутка времени и нарисована соответствующая круговая диаграмма | Тест пройден |
| 25 | Отображение отчета разницы между доходами и расходами за выбранный промежуток времени для выбранного пользователя | 1.Кликнуть на кнопку «Отчет»  2.Выбрать пользователя  3.Выбрать пункт «Доход минус расход» | В таблицу отчета будут загружены сумма всех расходов и сумма всех доходов за все время и нарисована соответствующая круговая диаграмма | Тест пройден |
| 26 | Отображение вкладки помощь | 1.Кликнуть на кнопку «Помощь» | Появятся кнопки раздела помощь | Тест пройден |

# **РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ**

Для того, чтобы запустить программу, нужно открыть файл CoinKeeper.exe. После открытия файла появится окно, показанное на рисунке 6.1.

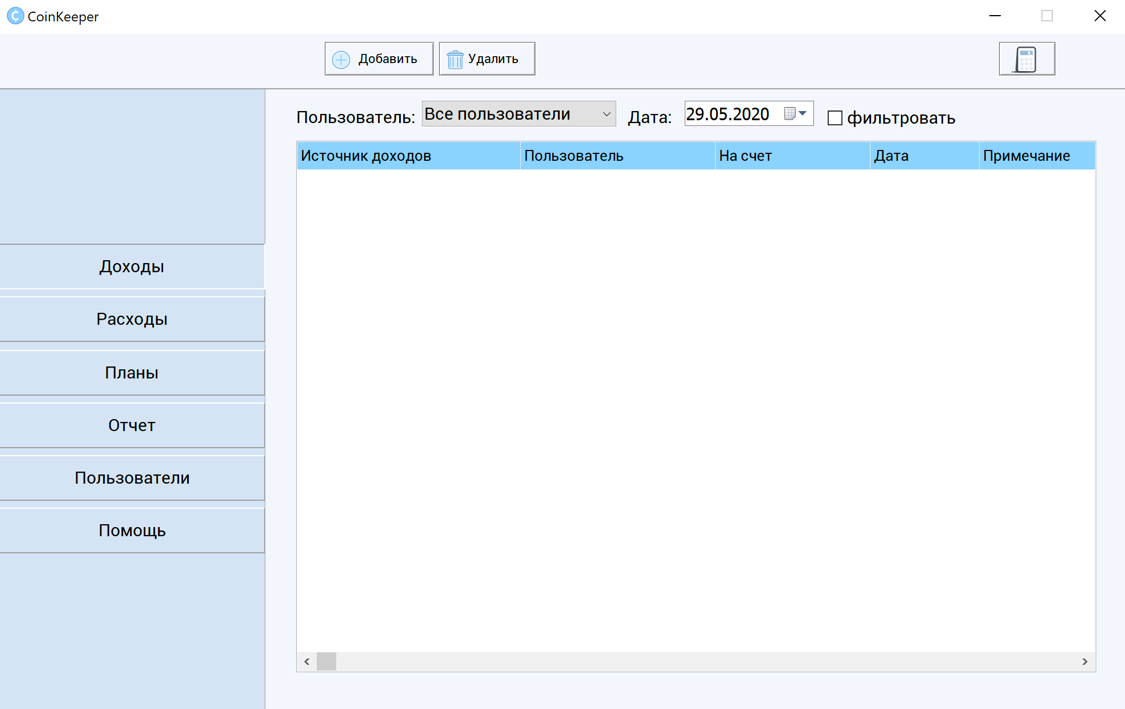


Рисунок 6.1 – окно программы CoinKeeper при открытии

Основное взаимодействие с программой осуществляется через интуитивно понятное пользовательское меню, которое состоит из 6 разделов: доходы, расходы, планы, отчет, пользователи, помощь.

Для добавления пользователя необходимо кликнуть на кнопку бокового меню «Пользователи». В данном разделе находится список пользователей и их счета.

Программой может пользоваться неограниченное количество пользователей, которые могут иметь до 10 счетов в разных валютах (белорусский рубль, российский рубль, доллар). Название счета можно выбрать из предложенных или ввести самостоятельно. Данный раздел показан на рисунках 6.2.-6.4

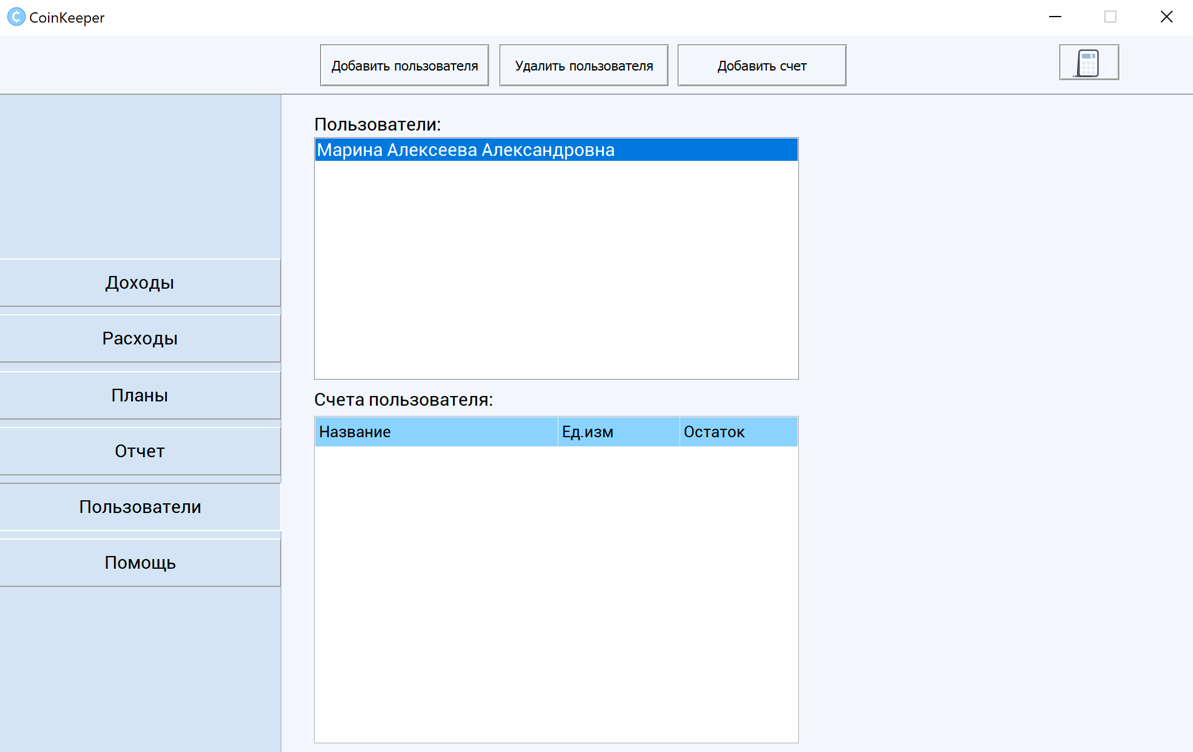


Рисунок 6.2 – раздел «Пользователи»

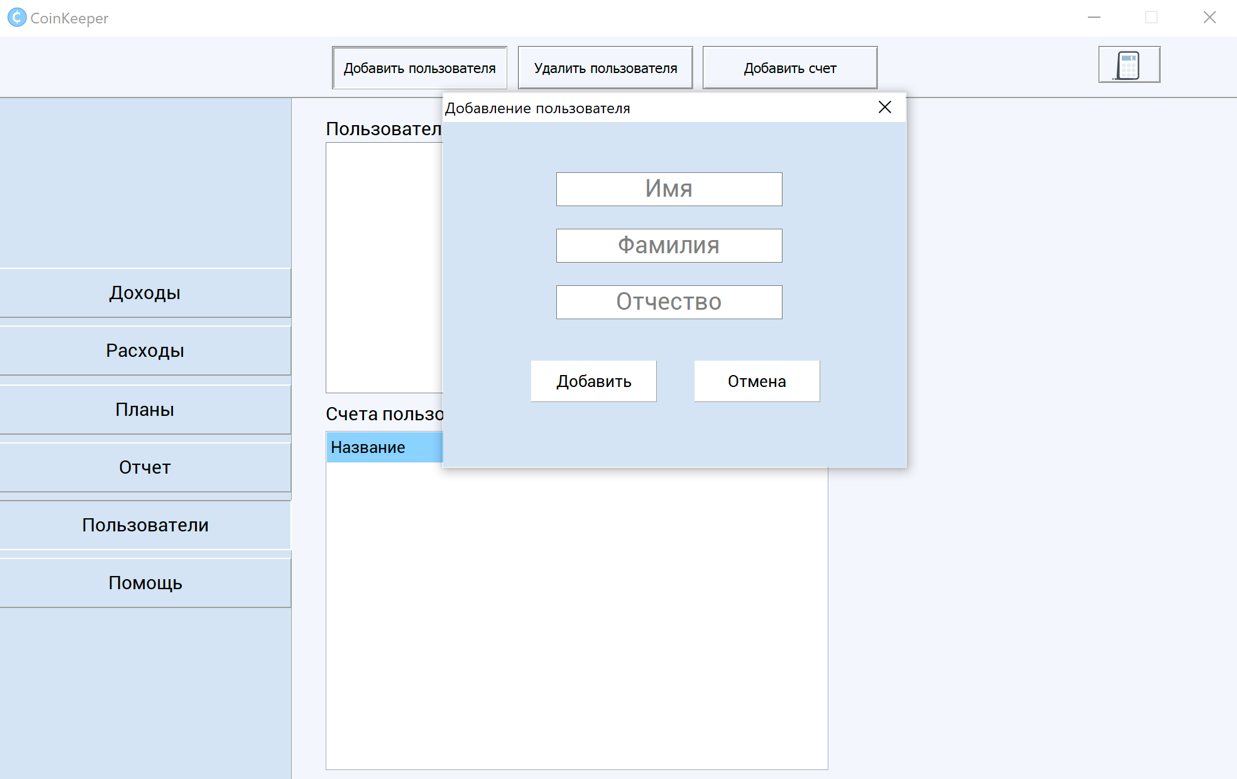


Рисунок 6.3 – добавление нового пользователя

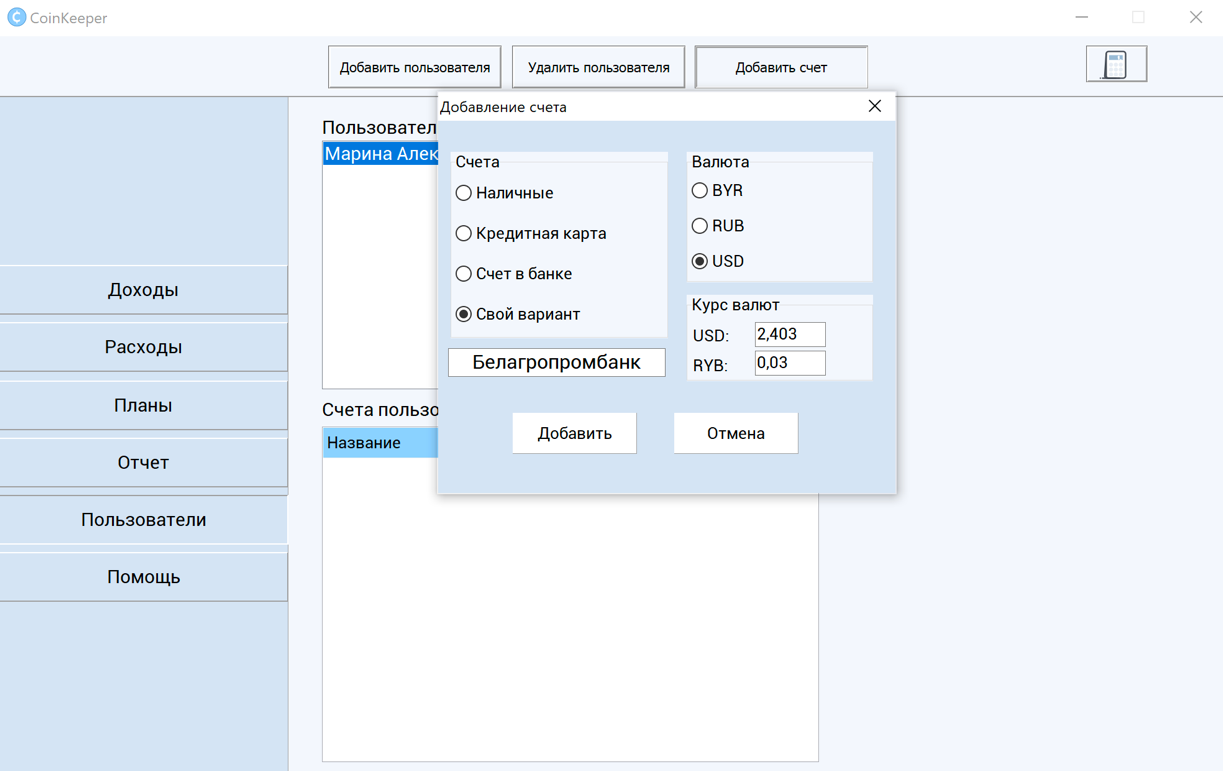


Рисунок 6.4 – добавление счета пользователя

После добавления необходимого количества пользователей и их счетов можно приступать к расчёту и планированию бюджета. Пополнение списка прибыли осуществляется в разделе «Доходы», который так же содержит таблицу заработка всех пользователей. Данный раздел представлен на рисунке 6.5.

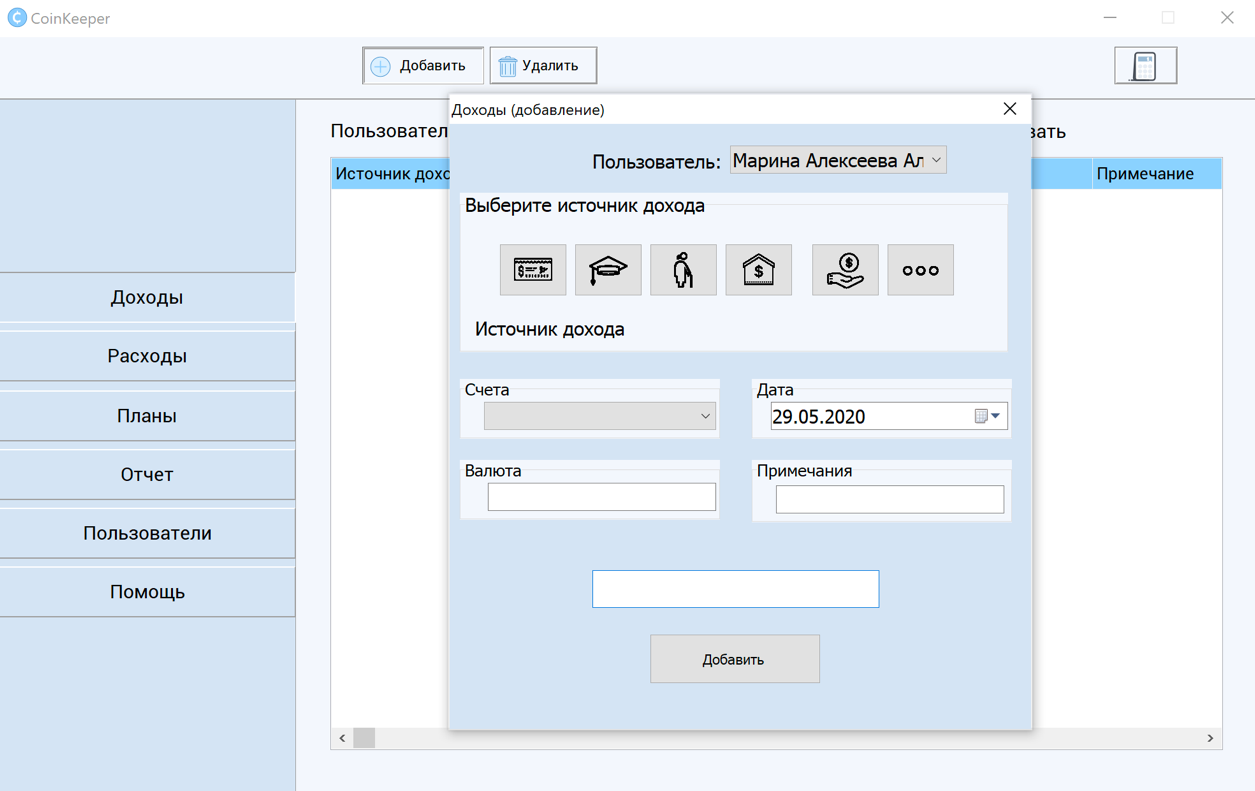


Рисунок 6.5 – добавление доходов пользователей

Внесение же затрат пользователя происходит в разделе «Расходы» после нажатия на кнопку «Добавить».

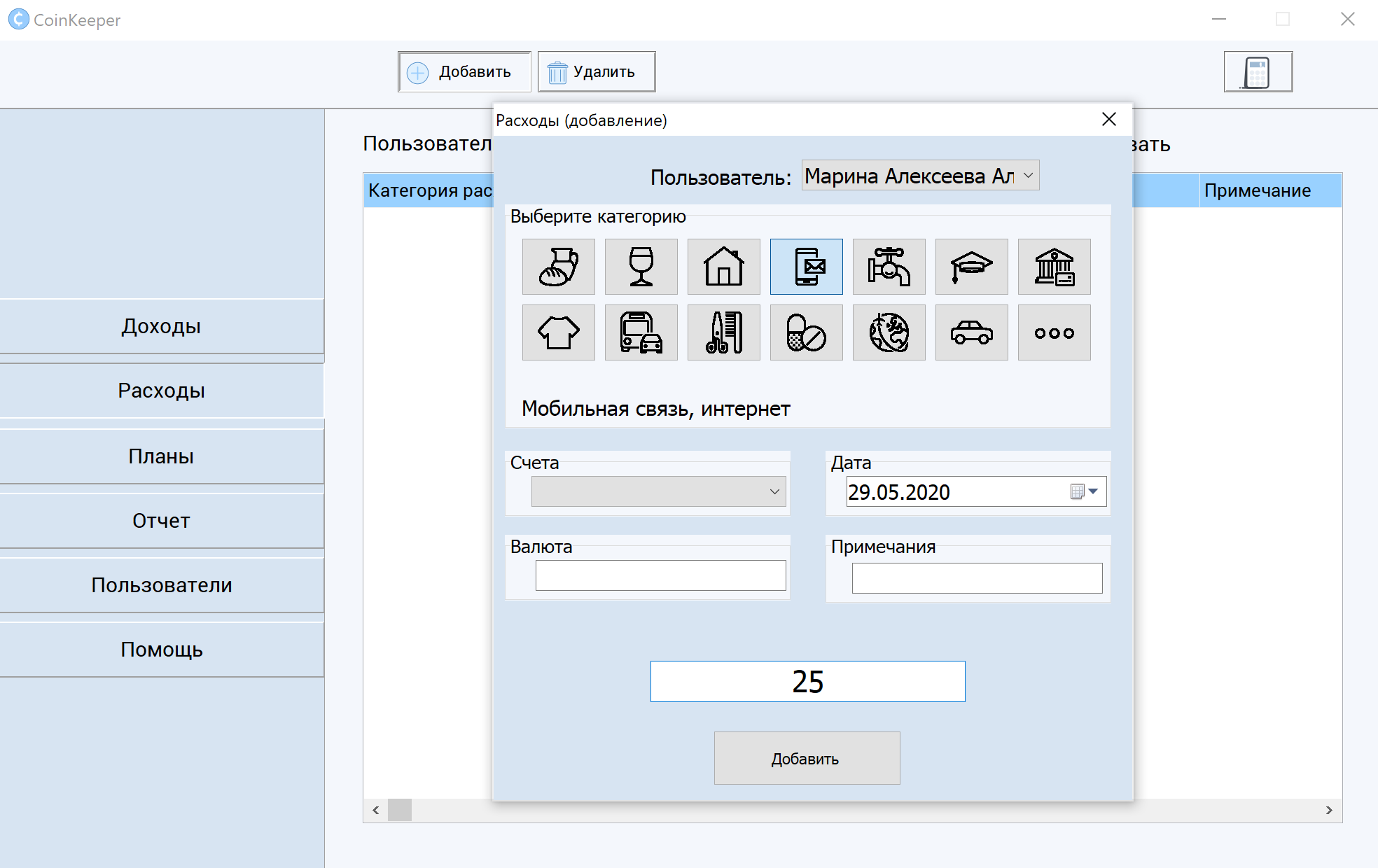


Рисунок 6.6 – добавление расходов пользователей

Планирование бюджета осуществляется в разделе «Планы». Для добавления планов расходов или же доходов на месяц необходимо кликнуть на соответствующую кнопку и ввести сумму в белорусских рублях. Так же для лучшего отслеживания бюджета в таблице планов расходов (доходов) отображается разница между фактическими и планируемыми расходами(доходами). Данный раздел изображен на рисунках 6.7-6.8.

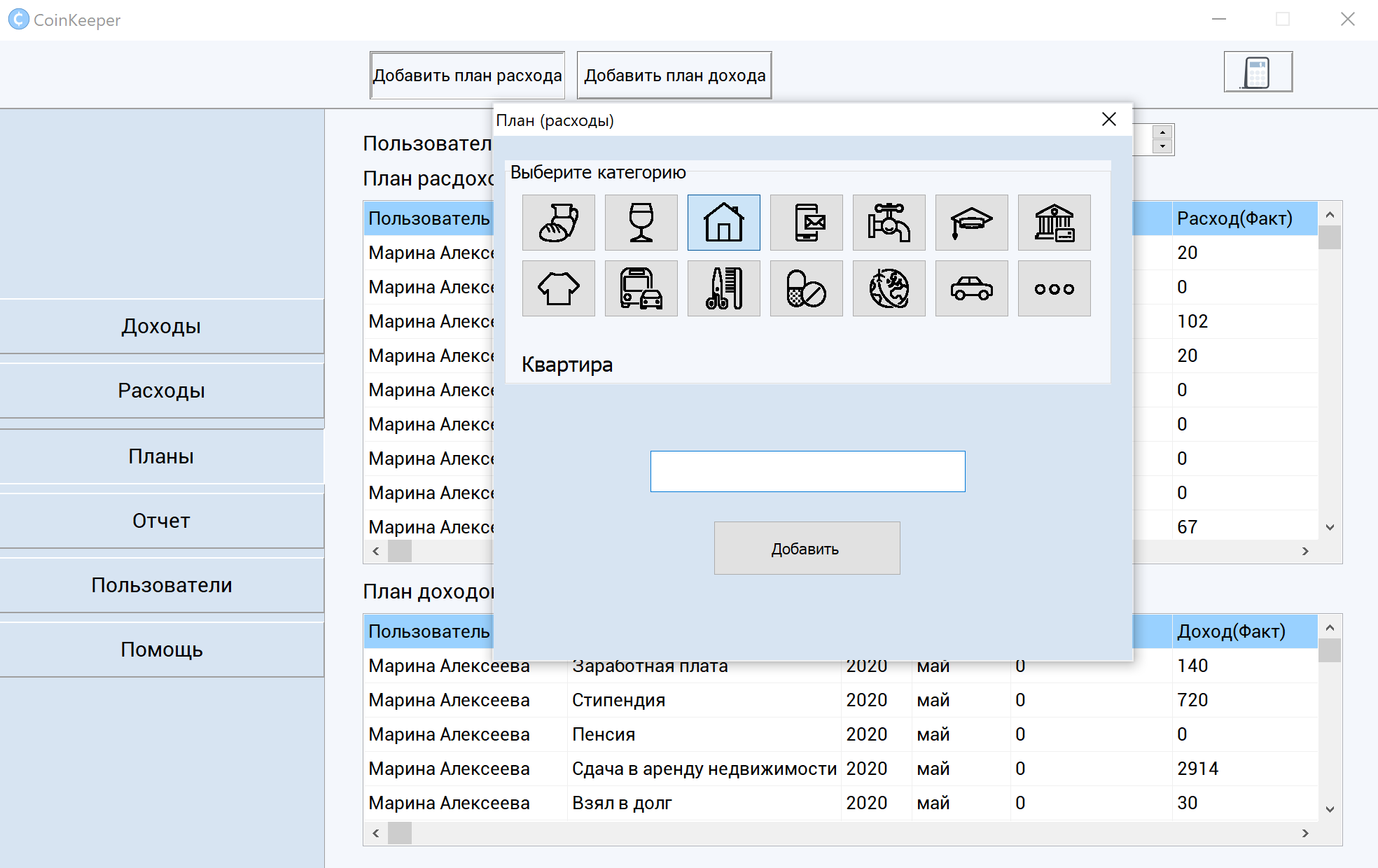


Рисунок 6.7 – добавление планов расходов на месяц

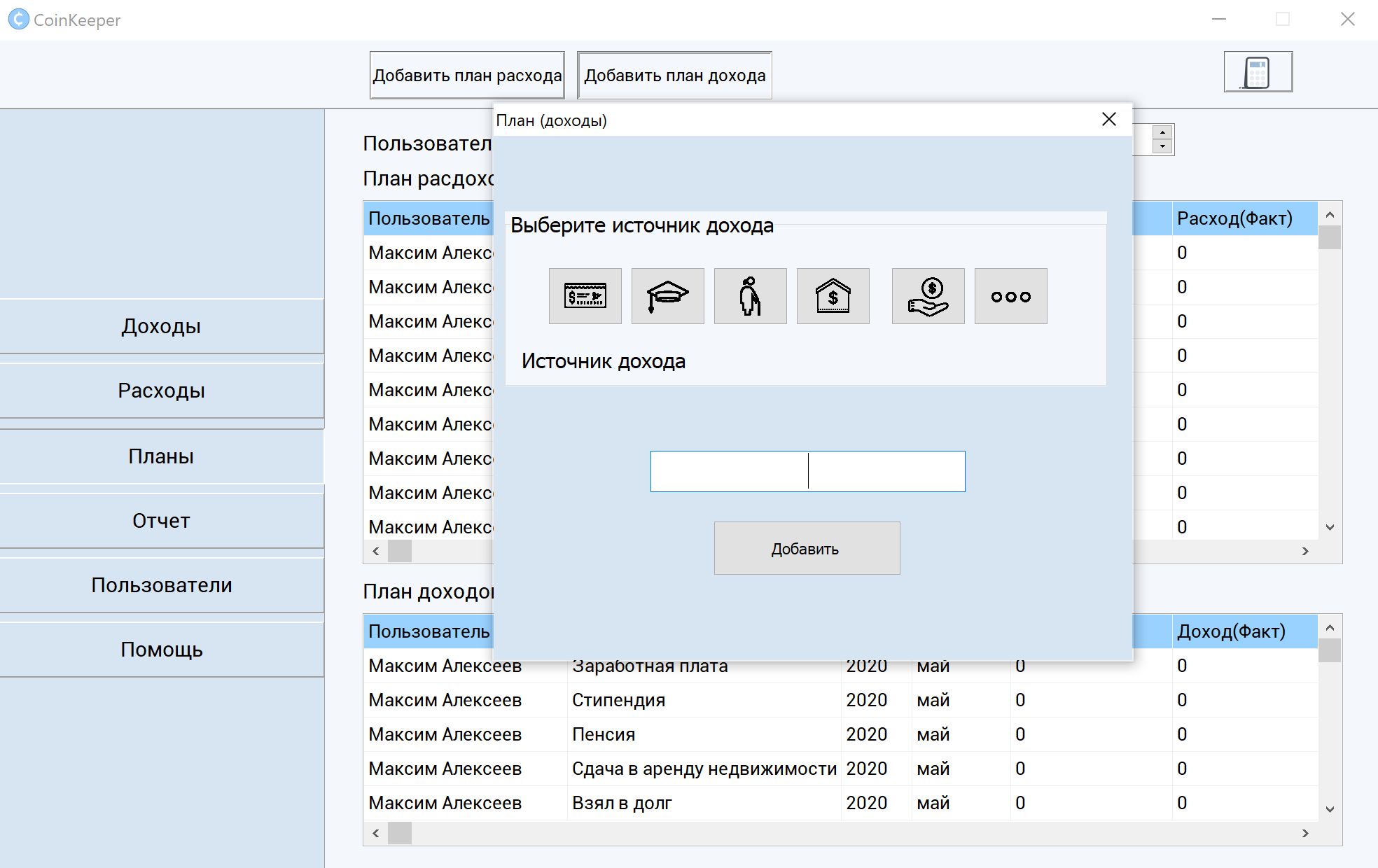


Рисунок 6.8 – добавление планов доходов на месяц

Для получения отчета о бюджете необходимо перейти в раздел «Отчет», выбрать пользователя и тип отчета. Суммы затрат или прибыли за выбранный промежуток времени размещаются в таблице по категориям. Процентное же соотношение расходов (доходов) к общей сумме расходов (доходов) предоставлено на круговой диаграмме.

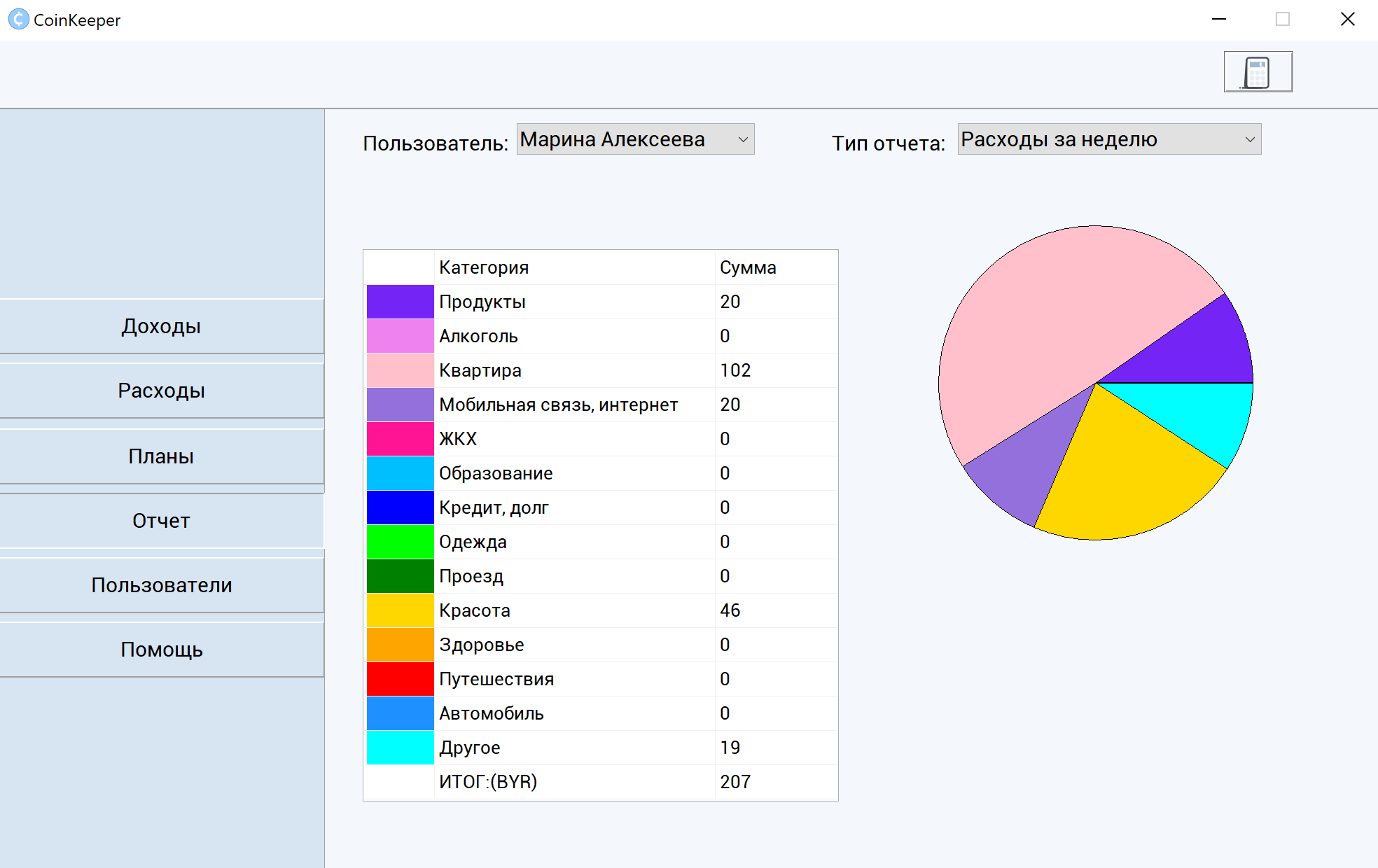


Рисунок 6.6 – отчет о бюджете за временной промежуток

Для получения справочной информации о работе программы и основных фикциях можно перейти в раздел «Помощь».

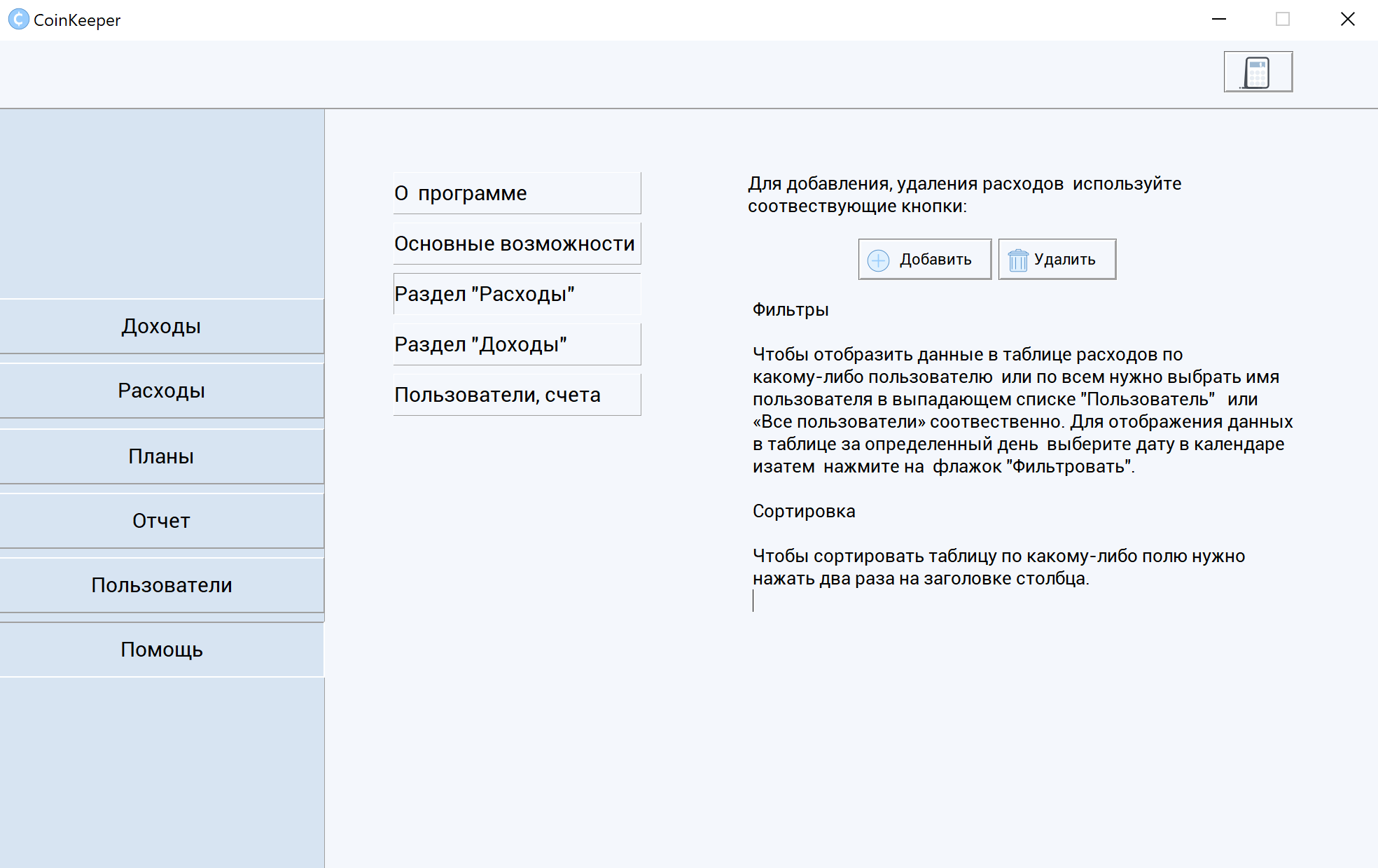


Рисунок 6.6 – раздел «Помощь»

Для удобства пользователей был добавлен калькулятор, которые открываются при нажатии на кнопку с соответствующей.

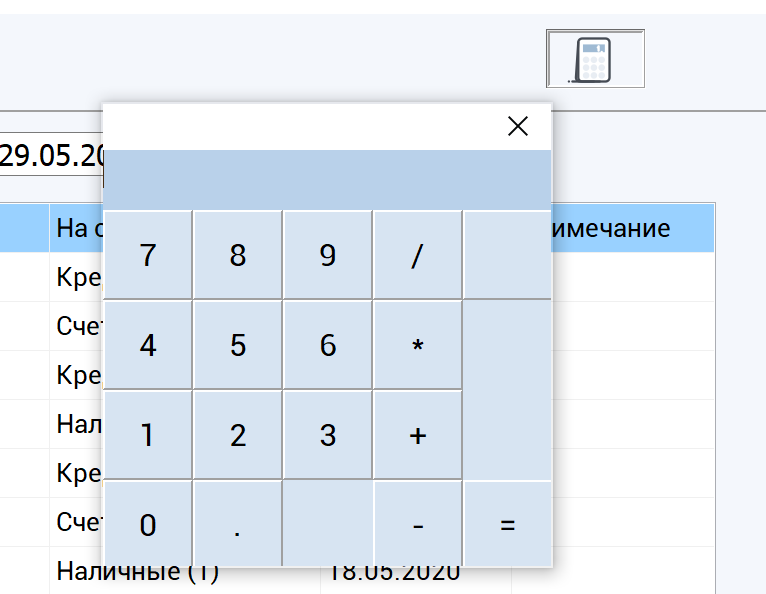


Рисунок 6.7 – калькулятор

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсового проекта была проанализирована предметная область, рассмотрены существующие аналоги, выявлены их преимущества и недостатки.

На этапе проектирования были изучены основные принципы работы с динамическими структурами данных, принципы работы с графикой на языке программирования Delphi, принципы работы с файлами, разработаны блок-схемы алгоритмов и графический интерфейс пользователя.

В соответствии с поставленной целью было разработано приложение, которое представляет собой средство учета семейного бюджета и содержит набор необходимых функций для управления, контроля и анализа бюджета.

Проведено тестирование работоспособности разработанной программной части. Поставленная цель была выполнена в полном объеме, работоспособность подтверждена тестированием программного средства. Разработано руководство пользователя.

# **СПИСОК ЛИТЕРАТУРЫ**

[1] Глухова, Л.А. Основы алгоритмизации и программирования: учебное пособие. В 2 Ч. / Л.А. Глухова. – БГУИР, 2006 – Ч. 1. – 195 с.

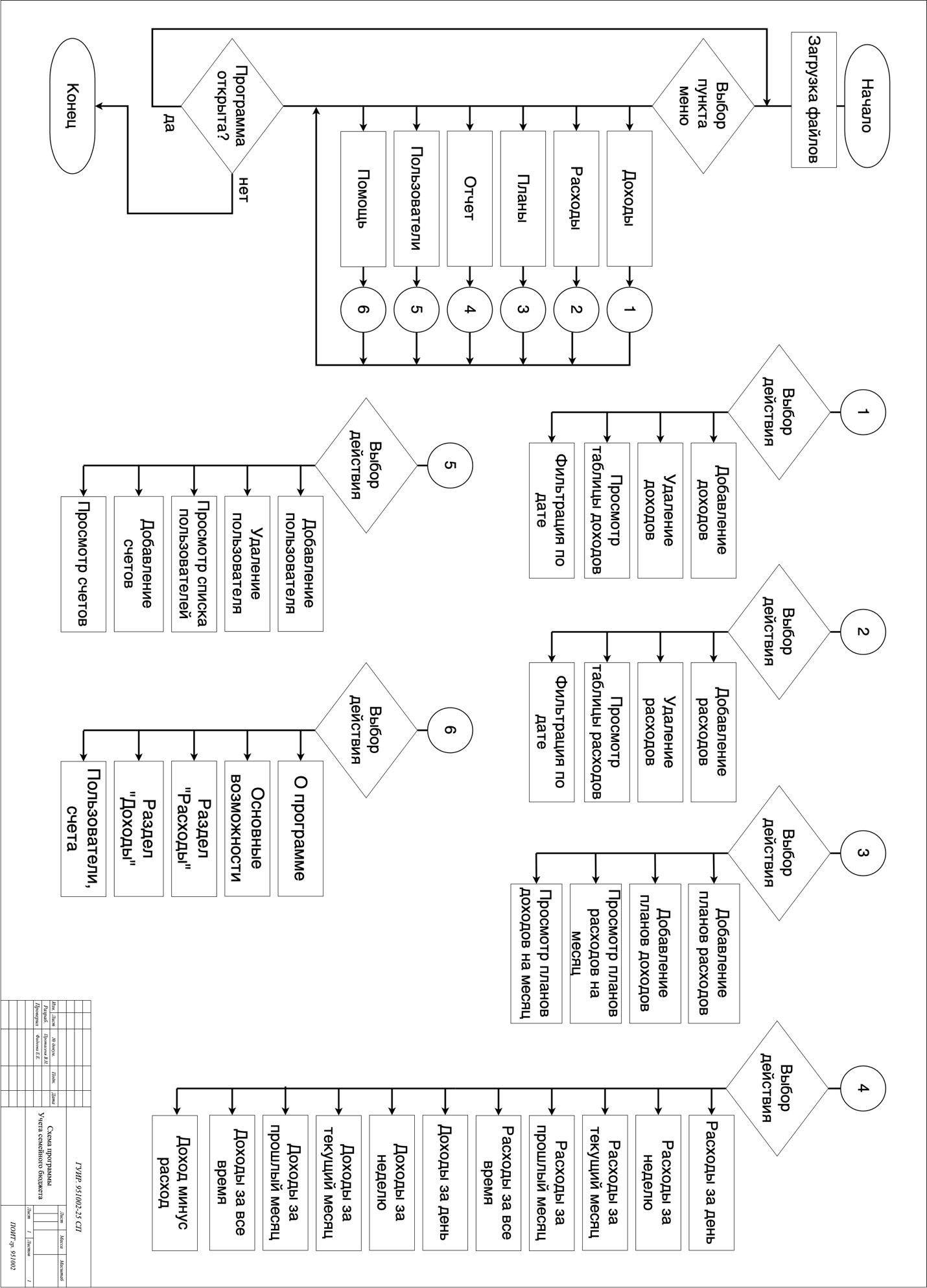
[2] Тюкачев Н.А. Программирование графики в Delphi: учебное пособие. / Н.А. Тюкачев, И.В. Илларионов, В.Г. Хлебострелов. – СПб. : БХВ-Петербург, 2008. – 784с.

[3] Фараонов В.В. Delphi 6: учебное пособие. / В.В.Фараонов. – М.: Издатель Молгачёва С.В., 2003. – 672 с.

[4] Infobank.by [Электронный ресурс]. – Электронные данные. - Режим доступа: https://infobank.by/prosmotr-konkursa/belorusskie-blogery-au-kto-xochet-zarabotatj-700-rublej/chto-takoe-semejnyj-byudzhet-i-zachem-on-nuzhen/

# **ПРИЛОЖЕНИЕ 1**

**Схема алгоритма работы программы**

****

# **ПРИЛОЖЕНИЕ 2**

**Исходный код программы**

**Модуль UnitMain**

unit UnitMain;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls,

Vcl.ComCtrls, DateUtils,

Vcl.Imaging.pngimage, IdBaseComponent, IdNetworkCalculator, Vcl.Grids,

Vcl.Buttons, Vcl.ButtonGroup, Vcl.CategoryButtons, Vcl.Samples.Gauges,

Vcl.Samples.Spin;

type

TfrmMain = class(TForm)

pnlMainMenu: TPanel; // кнопка отчет

pnlMoneySpend: TPanel;

pnlTopMenu: TPanel;

pnlAddBtn: TPanel; // кнопка изменить

pnlDeleteBtn: TPanel; // кнопка удалить

imgAdd: TImage; // кратинка изменить

imgDelete: TImage; // картинка удалить

// --------------------------

// управление пользователями

listboxUsers: TListBox;

pnlAddUserBtn: TPanel;

pnlDeleteUserBtn: TPanel;

sgMoneySpendTable: TStringGrid;

lbDateName: TLabel;

lbUserName: TLabel;

cbFilterForDate: TCheckBox;

cbUsers: TComboBox;

dtDateMoneySpend: TDateTimePicker;

lbUsers: TLabel;

pnlMoneyEarn: TPanel;

sgMoneyEarnTable: TStringGrid;

lblBillsName: TLabel;

sgUsersBills: TStringGrid;

pnlAddBill: TPanel;

pnlHelp: TPanel;

pnlHelpBtns: TPanel;

About: TPanel;

features: TPanel;

AboutSpend: TPanel;

AboutEarn: TPanel;

AboutUsersBills: TPanel;

lbSpendEarn1: TLabel;

imgSpendEarn: TImage;

memoAbout: TMemo;

memoFeatures: TMemo;

memoSpendEarn: TMemo;

memoUserBill: TMemo;

pnlReport: TPanel;

lbReportType: TLabel;

paintboxReport: TPaintBox;

cbReport: TComboBox;

cbUsersReport: TComboBox;

sgReportTable: TStringGrid;

pnlUsersBtn: TPanel;

pnlPlans: TPanel;

sgPlanSpend: TStringGrid;

lbPlanYear: TLabel;

cbMonth: TComboBox;

editYear: TSpinEdit;

pnlAddPlanSpend: TPanel;

pnlAddPlanEarn: TPanel;

lbPlanSpend: TLabel;

lbPlanEarn: TLabel;

sgPlanEarn: TStringGrid;

pnlCalculatorBtn: TPanel;

imgCalculator: TImage;

procedure FormCreate(Sender: TObject);

procedure pnlUsersBtnClick(Sender: TObject);

procedure pnlCalculatorBtnClick(Sender: TObject);

procedure pnlAddBtnClick(Sender: TObject);

procedure pnlDeleteBtnClick(Sender: TObject);

procedure pnlAddUserBtnClick(Sender: TObject);

procedure pnlMoneySpendClick(Sender: TObject);

procedure pnlDeleteUserBtnClick(Sender: TObject);

procedure cbUsersSelect(Sender: TObject);

procedure sgMoneySpendTableSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

procedure cbFilterForDateClick(Sender: TObject);

procedure pnlMoneyEarnClick(Sender: TObject);

procedure pnlAddBillClick(Sender: TObject);

procedure pnlHelpClick(Sender: TObject);

procedure AboutClick(Sender: TObject);

procedure featuresClick(Sender: TObject);

procedure AboutSpendClick(Sender: TObject);

procedure AboutEarnClick(Sender: TObject);

procedure AboutUsersBillsClick(Sender: TObject);

procedure sgMoneyEarnTableSelectCell(Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

procedure listboxUsersClick(Sender: TObject);

procedure pnlReportClick(Sender: TObject);

procedure paintboxReportPaint(Sender: TObject);

procedure cbReportSelect(Sender: TObject);

procedure sgReportTableDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure pnlPlansClick(Sender: TObject);

procedure FormResize(Sender: TObject);

procedure pnlAddPlanSpendClick(Sender: TObject);

procedure cbUsersReportSelect(Sender: TObject);

procedure cbMonthSelect(Sender: TObject);

procedure editYearChange(Sender: TObject);

procedure pnlAddPlanEarnClick(Sender: TObject);

private

procedure LoadUsers;

// процедуры видимости компонентов и эффекта нажатия кнопки

procedure MoneySpendVisible(Value: Boolean);

procedure UsersVisible(Value: Boolean);

procedure MoneyEarnVisible(Value: Boolean);

procedure HelpVisible(Value: Boolean);

procedure AboutVisible(Value: Boolean);

procedure FeaturesVisible(Value: Boolean);

procedure UserBillVisible(Value: Boolean);

procedure AboutSpendVisible(Value: Boolean);

procedure PlansVisible(Value: Boolean);

procedure ReportVisible(Value: Boolean);

// процедуры для таблиц

procedure FilterGridForDate(StringGrid: TStringGrid);

procedure DelRow(StringGrid: TStringGrid);

function IsCellSelected(StringGrid: TStringGrid; X, Y: Longint): Boolean;

procedure SortGrid(SrtingGrid: TStringGrid; Column: Integer; Row: Integer);

procedure SortGridForUser(StringGrid: TStringGrid);

procedure DeletUserFromTable(StringGrid: TStringGrid; UserIndex: Integer);

procedure LoadGridForOneDay(var StringGrid: TStringGrid; UserIndex: Integer;

Category: array of string);

procedure LoadGridForOneWeek(var StringGrid: TStringGrid;

UserIndex: Integer; Category: array of string);

procedure LoadReportGridMoneyEarn(var AllMoney: Integer;

UserIndex: Integer);

procedure LoadReportGridMoneySpend(var AllMoney: Integer;

UserIndex: Integer);

procedure DeleteMoneySpendFromArray(i: Integer);

procedure DeleteMoneyEarnFromArray(i: Integer);

procedure LoadFactPlan(UserIndex: Integer);

procedure LoadGridForMonth(var StringGrid: TStringGrid; UserIndex: Integer;

Category: array of string; MyDate: TDateTime);

procedure DeleteMoney(StringGrid: TStringGrid; TableNumb: Integer);

// подсчет денег

function SumAllForOneUsersBill(StringGrid: TStringGrid; BillName: string;

UserName: string): Integer;

function SearchMoneyForToday(StringGrid: TStringGrid; UserIndex: Integer;

Category: string): Integer;

procedure CountPlanMinusFact(StringGrid: TStringGrid);

function SearchMoneyForMonth(var StringGrid: TStringGrid;

UserIndex: Integer; Category: string; MyDate: TDateTime): Integer;

function SearchMoneyForWeek(StringGrid: TStringGrid; UserIndex: Integer;

Category: string): Integer;

// отрисовка диаграммы

procedure sector(clr: TColor; angle1, angle2: real);

procedure DrawAllMoneyEarnForAllTime(var AllMoney: Integer;

UserIndex: Integer; kind: Integer);

procedure DrawAllMoneySpendForAllTime(var AllMoney: Integer;

UserIndex: Integer; kind: Integer);

procedure ClearDiagram(PaintBox: TPaintBox);

{ Private declarations }

public

{ Public declarations }

end;

procedure SaveStringGrid(StringGrid: TStringGrid; const FileName: TFileName);

procedure LoadStringGrid(StringGrid: TStringGrid; sgNumber: Integer;

const FileName: TFileName);

procedure LoadUsersBills(StringGrid: TStringGrid);

function SumAllMoneySpend(UserNumb: Integer): Integer;

function SumAllMoneyEarn(UserNumb: Integer): Integer;

function CountProcent(var AllMoney: Integer; Category: Integer): real;

procedure GridClean(Sender: TObject);

procedure LoadPlanSpend(UserIndex: Integer; Month: Integer; Year: Integer);

procedure LoadPlanEarn(UserIndex: Integer; Month: Integer; Year: Integer);

procedure SaveChanges;

var

frmMain: TfrmMain;

implementation

Uses

UnitCalculator,

UnitAddMoney,

UnitData,

UnitAddUserInfo;

{$R \*.dfm}

// ------------------------------------------------------------------

// процедуры видимости компонентов

procedure TfrmMain.MoneySpendVisible(Value: Boolean);

begin

// видимость компонентов

sgMoneySpendTable.Visible := Value;

lbDateName.Visible := Value;

dtDateMoneySpend.Visible := Value;

cbFilterForDate.Visible := Value;

sgMoneySpendTable.Visible := Value;

// эффект нажатия

if Value then

pnlMoneySpend.BevelOuter := bvLowered

else

pnlMoneySpend.BevelOuter := bvRaised;

end;

procedure TfrmMain.UsersVisible(Value: Boolean);

begin

// видимость компонентов

lblBillsName.Visible := Value;

listboxUsers.Visible := Value;

lbUsers.Visible := Value;

pnlAddUserBtn.Visible := Value;

pnlDeleteUserBtn.Visible := Value;

pnlAddBill.Visible := Value;

sgUsersBills.Visible := Value;

pnlAddBill.Visible := Value;

// эффект нажатия

if Value then

pnlUsersBtn.BevelOuter := bvLowered

else

pnlUsersBtn.BevelOuter := bvRaised;

end;

procedure TfrmMain.MoneyEarnVisible(Value: Boolean);

begin

// видимость компонентов

sgMoneyEarnTable.Visible := Value;

// эффект нажатия

if Value then

pnlMoneyEarn.BevelOuter := bvLowered

else

pnlMoneyEarn.BevelOuter := bvRaised;

end;

procedure TfrmMain.HelpVisible(Value: Boolean);

begin

pnlHelpBtns.Visible := Value;

// эффект нажатия

if Value then

pnlHelp.BevelOuter := bvLowered

else

pnlHelp.BevelOuter := bvRaised;

end;

procedure TfrmMain.AboutVisible(Value: Boolean);

begin

memoAbout.Visible := Value;

// эффект нажатия

if Value then

About.BevelOuter := bvLowered

else

About.BevelOuter := bvRaised;

end;

procedure TfrmMain.FeaturesVisible(Value: Boolean);

begin

memoFeatures.Visible := Value;

// эффект нажатия

if Value then

features.BevelOuter := bvLowered

else

features.BevelOuter := bvRaised;

end;

procedure TfrmMain.UserBillVisible(Value: Boolean);

begin

memoUserBill.Visible := Value;

// эффект нажатия

if Value then

AboutUsersBills.BevelOuter := bvLowered

else

AboutUsersBills.BevelOuter := bvRaised;

end;

procedure TfrmMain.AboutSpendVisible(Value: Boolean);

begin

lbSpendEarn1.Visible := Value;

imgSpendEarn.Visible := Value;

memoSpendEarn.Visible := Value;

// эффект нажатия

if Value then

AboutSpend.BevelOuter := bvLowered

else

AboutSpend.BevelOuter := bvRaised;

end;

procedure TfrmMain.ReportVisible(Value: Boolean);

begin

cbReport.Visible := Value;

lbReportType.Visible := Value;

cbUsersReport.Visible := Value;

sgReportTable.Visible := Value;

// эффект нажатия

if Value then

pnlReport.BevelOuter := bvLowered

else

pnlReport.BevelOuter := bvRaised;

end;

procedure TfrmMain.PlansVisible(Value: Boolean);

begin

sgPlanSpend.Visible := Value;

lbPlanYear.Visible := Value;

cbMonth.Visible := Value;

editYear.Visible := Value;

pnlAddPlanSpend.Visible := Value;

pnlAddPlanEarn.Visible := Value;

lbPlanSpend.Visible := Value;

lbPlanEarn.Visible := Value;

sgPlanEarn.Visible := Value;

// эффект нажатия

if Value then

pnlPlans.BevelOuter := bvLowered

else

pnlPlans.BevelOuter := bvRaised;

end;

// ------------------------------------------------------------------

// процедуры обработки нажатий в кнопок меню меню

procedure TfrmMain.AboutUsersBillsClick(Sender: TObject);

begin

UserBillVisible(true);

AboutSpendVisible(false);

AboutVisible(false);

FeaturesVisible(false);

AboutEarn.BevelOuter := bvRaised;

AboutSpend.BevelOuter := bvRaised;

// пользователи, счета

memoUserBill.Top := 50;

memoUserBill.Left := 570;

end;

procedure TfrmMain.AboutEarnClick(Sender: TObject);

begin

AboutSpendVisible(true);

AboutEarn.BevelOuter := bvLowered;

AboutSpend.BevelOuter := bvRaised;

lbSpendEarn1.Caption :=

'Для добавления, удаления доходов используйте соотвествующие кнопки:';

FeaturesVisible(false);

AboutVisible(false);

UserBillVisible(false);

end;

procedure TfrmMain.AboutSpendClick(Sender: TObject);

begin

FeaturesVisible(false);

AboutVisible(false);

UserBillVisible(false);

imgSpendEarn.Visible := true;

AboutEarn.BevelOuter := bvRaised;

AboutSpendVisible(true);

lbSpendEarn1.Caption :=

'Для добавления, удаления расходов используйте соотвествующие кнопки:';

lbSpendEarn1.Visible := true;

end;

procedure TfrmMain.featuresClick(Sender: TObject);

var

i: Integer;

begin

// эффект нажатия

AboutVisible(false);

lbSpendEarn1.Visible := false;

AboutSpendVisible(false);

UserBillVisible(false);

FeaturesVisible(true);

// основные возможности

memoFeatures.Top := 50;

memoFeatures.Left := 570;

memoFeatures.Height := 1500;

memoFeatures.Width := 800;

end;

procedure TfrmMain.pnlMoneyEarnClick(Sender: TObject);

begin

fMoneyEarn := true;

fMoneySpend := false;

fPlanSpend := false;

fPlanEarn := false;

// видимые компоненты

MoneyEarnVisible(true);

pnlAddBtn.Visible := true;

pnlDeleteBtn.Visible := true;

cbUsers.Visible := true;

lbUserName.Visible := true;

// невидимые компоненты

ReportVisible(false);

UsersVisible(false);

MoneySpendVisible(false);

HelpVisible(false);

PlansVisible(false);

dtDateMoneySpend.Visible := true;

lbDateName.Visible := true;

cbFilterForDate.Visible := true;

end;

procedure TfrmMain.pnlMoneySpendClick(Sender: TObject); // расходы

begin

fMoneySpend := true;

fMoneyEarn := false;

fPlanSpend := false;

fPlanEarn := false;

// видимые компоненты

MoneySpendVisible(true);

pnlAddBtn.Visible := true;

pnlDeleteBtn.Visible := true;

lbUserName.Visible := true;

cbUsers.Visible := true;

// невидимые компоненты

UsersVisible(false);

MoneyEarnVisible(false);

HelpVisible(false);

ReportVisible(false);

PlansVisible(false);

// диаграмма очистка

ClearDiagram(paintboxReport);

end;

procedure TfrmMain.pnlPlansClick(Sender: TObject);

var

i: Integer;

SearchDate: TDateTime;

MyMonth, UserIndex, SumMoney, AllForMonth, MonthIndex, MyYear, Month,

Year: Integer;

YearNumb, MonthNumb, DayNumb: Word;

UserName: string;

begin

// флаги для модальной формы

fPlanSpend := false;

fMoneyEarn := false;

fMoneySpend := false;

fPlanEarn := false;

// невидимые компоненты

UsersVisible(false);

MoneySpendVisible(false);

MoneyEarnVisible(false);

HelpVisible(false);

ReportVisible(false);

cbUsers.Visible := false;

pnlAddBtn.Visible := false;

pnlDeleteBtn.Visible := false;

// видимые комноненты

PlansVisible(true);

cbUsersReport.Visible := true;

lbUserName.Visible := true;

lbReportType.Visible := true;

lbReportType.Left := 1150;

lbReportType.Caption := 'Месяц:';

sgPlanSpend.RowCount := 15; // кол-во строк в таблице планов расходов

sgPlanEarn.RowCount := 7; // кол-во строк в таблице планов доходов

// заполнение шапок таблиц

for i := 1 to sgPlanSpend.RowCount - 1 do // категории расходов

begin

sgPlanSpend.cells[1, i] := sgCategorySpendPlan[i];

end;

for i := 1 to sgPlanEarn.RowCount - 1 do // категорий доходов

begin

sgPlanEarn.cells[1, i] := sgCategoryEarnPlan[i];

end;

// установка текущего месяца

DecodeDate(now, YearNumb, MonthNumb, DayNumb);

cbMonth.ItemIndex := MonthNumb - 1;

lbPlanSpend.Caption := 'План расдохов:' + ' ' + cbMonth.Text;

lbPlanEarn.Caption := 'План доходов:' + ' ' + cbMonth.Text;

// установка текущего года

editYear.Value := YearNumb;

// месяц для поиска

MyMonth := cbMonth.ItemIndex + 1;

MyYear := editYear.Value;

SearchDate := EncodeDate(MyYear, MyMonth, 5);

Month := frmMain.cbMonth.ItemIndex + 1;

Year := frmMain.editYear.Value;

for i := 0 to (sgPlanSpend.RowCount - 1) do

begin

sgPlanSpend.cells[2, i + 1] := editYear.Text; // год

sgPlanSpend.cells[3, i + 1] := cbMonth.Text; // месяц

end;

for i := 0 to (sgPlanEarn.RowCount - 1) do

begin

sgPlanEarn.cells[2, i + 1] := editYear.Text; // год

sgPlanEarn.cells[3, i + 1] := cbMonth.Text; // месяц

end;

if length(ArrayOfUsers) > 0 then

begin

UserIndex := cbUsersReport.ItemIndex;

UserName := cbUsersReport.Items[cbUsersReport.ItemIndex];

// заполенение таблиц

// ПЛАН РАСХОДОВ

for i := 0 to (sgPlanSpend.RowCount - 1) do

sgPlanSpend.cells[0, i + 1] := UserName; // пользователь

// ПЛАН ДОХОДОВ

for i := 0 to (sgPlanEarn.RowCount - 1) do

sgPlanEarn.cells[0, i + 1] := UserName; // пользователь

// заполнение Расход/Доход (план)

LoadPlanSpend(UserIndex, Month, Year);

LoadPlanEarn(UserIndex, Month, Year);

// заполнение Расход/Доход ( факт)

LoadFactPlan(UserIndex);

// подсчет и заполнение ПЛАН- ФАКТ

CountPlanMinusFact(sgPlanSpend);

CountPlanMinusFact(sgPlanEarn);

end;

// диаграмма очистка

ClearDiagram(paintboxReport);

end;

procedure TfrmMain.pnlReportClick(Sender: TObject);

begin

fReport := true;

// видимые компоненты

ReportVisible(true);

lbReportType.Caption := 'Тип отчета:';

lbUserName.Visible := true;

// невидимые компоненты

UsersVisible(false);

MoneyEarnVisible(false);

HelpVisible(false);

MoneySpendVisible(false);

PlansVisible(false);

pnlAddBtn.Visible := false;

pnlDeleteBtn.Visible := false;

paintboxReport.Invalidate;

cbReport.ItemIndex := 0;

cbReportSelect(cbReport);

if cbUsersReport.Items.Count > 0 then

cbUsersReport.ItemIndex := 0

else

cbUsersReport.ItemIndex := -1;

// диаграмма

paintboxReport.Width := 450;

paintboxReport.Height := 450;

// снятие выделения с первой ячейки

GridClean(sgReportTable);

end;

procedure TfrmMain.pnlUsersBtnClick(Sender: TObject); // пользователи

begin

// очистка таблицы счетов

sgUsersBills.RowCount := 1;

if listboxUsers.Items.Count > 0 then

begin

listboxUsers.ItemIndex := 0;

listboxUsersClick(listboxUsers);

end;

// Видимые компоненты

UsersVisible(true);

// Нeвидимые компоненты

MoneySpendVisible(false);

MoneyEarnVisible(false);

HelpVisible(false);

ReportVisible(false);

PlansVisible(false);

pnlAddBtn.Visible := false;

pnlDeleteBtn.Visible := false;

cbUsers.Visible := false;

lbUserName.Visible := false;

// диаграмма очистка

ClearDiagram(paintboxReport);

end;

procedure TfrmMain.pnlAddUserBtnClick(Sender: TObject); // добавить пользователя

var

i: Integer;

begin

pnlAddUserBtn.BevelOuter := bvLowered;

for i := 0 to listboxUsers.Items.Count - 1 do

listboxUsers.Selected[i] := false;

fBill := false;

frmAddUserInfo.ShowModal;

end;

procedure TfrmMain.AboutClick(Sender: TObject);

begin

AboutVisible(true);

FeaturesVisible(false);

lbSpendEarn1.Visible := false;

AboutSpendVisible(false);

UserBillVisible(false);

end;

// добавить счет

procedure TfrmMain.pnlAddBillClick(Sender: TObject);

var

i: Integer;

begin

if (listboxUsers.Items.Count > 0) then

// проверка наличия пользователей в списке

begin

for i := 0 to listboxUsers.Count - 1 do // проверка выбран ли пользователь

begin

if listboxUsers.Selected[i] then

begin

fBill := true;

pnlAddBill.BevelOuter := bvLowered;

frmAddUserInfo.ShowModal; // открытие модальной формы добавления счета

end;

end;

end

else // список пользователей пустой

Showmessage('Список пользователей пуст. Добавьте пользователя!');

end;

// открытие формы добавления расходов/доходов по клику

procedure TfrmMain.pnlAddBtnClick(Sender: TObject);

var

i: byte;

begin

fSort := false;

// очистка комбобокса с именами пользователей т.к. могут остаться значения предыдущего открытия

frmAddMoney.cbUsersModalUnit.Items.Clear;

// добавление пользователей в комбобокс

for i := 1 to length(ArrayOfUsers) do

begin

frmAddMoney.cbUsersModalUnit.Items.Add(ArrayOfUsers[i - 1].Name + ' ' +

ArrayOfUsers[i - 1].Surname + ' ' + ArrayOfUsers[i - 1].Lastname);

end;

sgReportTable.RowCount := 1;

// установка начального значения в комбобоксе

if (frmAddMoney.cbUsersModalUnit.Items.Count > 0) then

// если есть пользователи, то показывает первого

frmAddMoney.cbUsersModalUnit.ItemIndex := 0

else

frmAddMoney.cbUsersModalUnit.ItemIndex := -1; // иначе ничего не выбрано

// очистка комбобокса с наз. счетов т.к. могут остаться значения предыдущего открытия

frmAddMoney.cbUserBills.Clear;

// если пользователи есть, то добавление счетов выбранного пользователя

if length(ArrayOfUsers) > 0 then

begin

i := 0;

while (ArrayOfUsers[frmAddMoney.cbUsersModalUnit.ItemIndex].Bill[i]

.BillName <> '') do

begin

frmAddMoney.cbUserBills.Items.Add

(ArrayOfUsers[frmAddMoney.cbUsersModalUnit.ItemIndex].Bill[i].BillName);

inc(i);

end;

// если у пользователя есть счета, то показывает название певрого счета и валюту

if frmAddMoney.cbUserBills.Items.Count > 0 then

begin

with frmAddMoney do

begin

cbUserBills.ItemIndex := 0;

editValuta.Text := ArrayOfUsers[frmAddMoney.cbUsersModalUnit.ItemIndex]

.Bill[cbUserBills.ItemIndex].Valuta;

end;

end;

// если форма не открыта, то открываем

if not(frmAddMoney.Visible) then

begin

pnlAddBtn.BevelOuter := bvLowered; // изменение кнопки-->нажата

frmAddMoney.ShowModal; // открытие формы

end

else // если открыта, то закрываем

begin

pnlAddBtn.BevelOuter := bvRaised; // изменение кнопки-->не нажата

frmAddMoney.Close // закрытие формы

end;

end

else

Showmessage('Список пользователей пуст. Добавьте пользователя!');

end;

// добавление плана дохода

procedure TfrmMain.pnlAddPlanEarnClick(Sender: TObject);

begin

fPlanEarn := true;

fPlanSpend := false;

if length(ArrayOfUsers) > 0 then

begin

// если форма не открыта, то открываем

if not(frmAddMoney.Visible) then

begin

pnlAddPlanEarn.BevelOuter := bvLowered; // изменение кнопки-->нажата

frmAddMoney.ShowModal; // открытие формы

end

else // если открыта, то закрываем

begin

pnlAddPlanEarn.BevelOuter := bvRaised; // изменение кнопки-->не нажата

frmAddMoney.Close // закрытие формы

end;

end

else

Showmessage('Добавьте пользователя!');

end;

// Добавлкник плана

procedure TfrmMain.pnlAddPlanSpendClick(Sender: TObject);

begin

fPlanSpend := true;

fPlanEarn := false;

if length(ArrayOfUsers) > 0 then

begin

// если форма не открыта, то открываем

if not(frmAddMoney.Visible) then

begin

pnlAddPlanSpend.BevelOuter := bvLowered; // изменение кнопки-->нажата

frmAddMoney.ShowModal; // открытие формы

end

else // если открыта, то закрываем

begin

pnlAddPlanSpend.BevelOuter := bvRaised; // изменение кнопки-->не нажата

frmAddMoney.Close // закрытие формы

end;

end

else

Showmessage('Добавьте пользователя!');

end;

// процедура открытия калькулятора по клику на кнопку

procedure TfrmMain.pnlCalculatorBtnClick(Sender: TObject);

begin

if not(frmCalculator.Visible) then

begin

frmCalculator.Show;

pnlCalculatorBtn.BevelOuter := bvLowered;

end

else

begin

frmCalculator.Close;

pnlCalculatorBtn.BevelOuter := bvRaised;

end;

end;

// процедура удаления расходов/доходов

procedure TfrmMain.pnlDeleteBtnClick(Sender: TObject);

var

UserName: string;

begin

if fMoneySpend then // расходы

begin

DeleteMoney(sgMoneySpendTable, 1);

end;ы

if fMoneyEarn then // доходы

begin

DeleteMoney(sgMoneyEarnTable, 2);

end;

end;

// ------------------------------------------------------------------

// процедура создания главной формы, инициализация переменных

procedure TfrmMain.FormClose(Sender: TObject; var Action: TCloseAction);

begin

sgReportTable.RowCount := 1;

SaveChanges;

end;

procedure TfrmMain.FormCreate(Sender: TObject);

const

sgCaptionBills: array [0 .. 2] of string = ('Название', 'Ед.изм', 'Остаток');

cbReportname: array [0 .. 10] of string = ('Расходы за день',

'Расходы за неделю', 'Расходы за текущий месяц', 'Расходы за прошлый месяц',

'Расходы за все время', 'Доходы за день', 'Доходы за неделю',

'Доходы за текущий месяц', 'Доходы за прошлый месяц', 'Доходы за все время',

'Доход минус расход');

sgReportName: array [0 .. 2] of string = ('', 'Категория', 'Сумма');

var

i: byte;

j, k, m: byte;

Rect: TRect;

begin

// создание/ открытие файла с информацией о пользователях

AssignFile(UserInfoFile, './User''s info.txt');

if not(FileExists('./User''s info.txt')) then

begin

Rewrite(UserInfoFile);

CloseFile(UserInfoFile);

end;

SetLength(ArrayOfUsers, 0); // инициализации длины массива

LoadUsers; // загрузка данных пользователей из файла в массив

LoadStringGrid(sgMoneySpendTable, 1, './Money spend info.txt');

// загрузка данных из файла в таблицу расходов

LoadStringGrid(sgMoneyEarnTable, 2, './Money earn info.txt');

// загрузка данных из файла в таблицу доходов

LoadStringGrid(sgPlanSpend, 3, './Plan Spend info.txt');

// загрузка данных из файла в таблицу планов расходов

LoadStringGrid(sgPlanEarn, 4, './Plan Earn info.txt');

// загрузка данных из файла в таблицу планов доходов

for i := 1 to length(ArrayOfUsers) do

begin

listboxUsers.Items.Add(ArrayOfUsers[i - 1].Name + ' ' + ArrayOfUsers[i - 1]

.Surname + ' ' + ArrayOfUsers[i - 1].Lastname);

cbUsers.Items.Add(ArrayOfUsers[i - 1].Name + ' ' + ArrayOfUsers

[i - 1].Surname);

cbUsersReport.Items.Add(ArrayOfUsers[i - 1].Name + ' ' + ArrayOfUsers

[i - 1].Surname);

end;

with sgUsersBills do

begin

// присвоение названия ячеек

for k := 0 to 2 do

begin

cells[k, 0] := sgCaptionBills[k];

end;

end;

if cbUsersReport.Items.Count > 0 then

cbUsersReport.ItemIndex := 0

else

cbUsersReport.ItemIndex := -1;

// присвоение названия ячеек таблице отчета

for i := 0 to (sgReportTable.ColCount - 1) do

begin

sgReportTable.cells[i, 0] := sgReportName[i];

end;

USD := 2.403;

RUB := 0.03;

// заполнения комбобокса "Тип отчета"

for m := 0 to length(cbReportname) - 1 do

begin

cbReport.Items.Add(cbReportname[m]);

end;

// Установка в календари текущего времени

dtDateMoneySpend.Date := now;

// clndrMonth.Date := now;

fSort := true;

fMoneyEarn := true;

fMoneySpend := false;

fPlanSpend := false;

fPlanEarn := false;

MoneyEarnVisible(true);

cbUsers.Visible := true;

lbUserName.Visible := true;

dtDateMoneySpend.Visible := true;

lbDateName.Visible := true;

cbFilterForDate.Visible := true;

// ----------------------------------

// размеры и координаты появления форм

// панель верхнего меню

pnlTopMenu.Width := 2500;

pnlTopMenu.Height := 100;

// кнопка добавить

pnlAddBtn.Top := 17;

pnlAddBtn.Left := 570;

imgAdd.Top := 12;

imgAdd.Left := 9;

// кнопка удалить

pnlDeleteBtn.Top := 17;

pnlDeleteBtn.Left := 770;

imgDelete.Top := 12;

imgDelete.Left := 9;

// календарь маленький

dtDateMoneySpend.Left := 1200;

dtDateMoneySpend.Top := 120;

// кпнока калькулятора

pnlCalculatorBtn.Top := 17;

pnlCalculatorBtn.Left := 1750;

imgCalculator.Top := 4;

imgCalculator.Left := 17;

// таблица доходов

sgMoneyEarnTable.Left := 520;

sgMoneyEarnTable.Top := 190;

sgMoneyEarnTable.Width := 1400;

sgMoneyEarnTable.Height := 930;

sgMoneyEarnTable.ColWidths[0] := 390; // ширина валюты

sgMoneyEarnTable.ColWidths[5] := 150; // ширина валюты

sgMoneyEarnTable.ColWidths[2] := 270; // ширина счет

sgMoneyEarnTable.ColWidths[3] := 190; // ширина даты

// комбобокс пользователей

cbUsers.Top := 120;

cbUsers.Left := 740;

// надпись "Пользователь"

lbUserName.Left := 520;

lbUserName.Top := 130;

lbUsers.Left := 520;

lbUsers.Top := 130;

// надпись "дата"

lbDateName.Left := 1100;

lbDateName.Top := 130;

// таблица расходов

sgMoneySpendTable.Left := 520;

sgMoneySpendTable.Top := 190;

sgMoneySpendTable.Width := 1400;

sgMoneySpendTable.Height := 930;

sgMoneySpendTable.ColWidths[0] := 390; // ширина валюты

sgMoneySpendTable.ColWidths[5] := 150; // ширина валюты

sgMoneySpendTable.ColWidths[2] := 270; // ширина счет

sgMoneySpendTable.ColWidths[3] := 190; // ширина даты

// надпись "фильторвать"

cbFilterForDate.Left := 1450;

cbFilterForDate.Top := 120;

// надпись "Счета пользователя"

lblBillsName.Left := 520;

lblBillsName.Top := 584;

// все пользователи список

listboxUsers.Width := 800;

listboxUsers.Height := 400;

listboxUsers.Left := 520;

listboxUsers.Top := 170;

// счета пользователей таблица

sgUsersBills.Width := 800;

sgUsersBills.ColWidths[0] := 400; // название счета

sgUsersBills.ColWidths[1] := 200; // валюта

sgUsersBills.ColWidths[2] := 200; // отстаток

sgUsersBills.Height := 540;

sgUsersBills.Left := 520;

sgUsersBills.Top := 630;

// кнопка добавить пользователя

pnlAddUserBtn.Width := 280;

pnlAddUserBtn.Height := 70;

pnlAddUserBtn.Top := 17;

pnlAddUserBtn.Left := 530;

// кнопка удалить пользователя

pnlDeleteUserBtn.Width := 280;

pnlDeleteUserBtn.Height := 70;

pnlDeleteUserBtn.Top := 17;

pnlDeleteUserBtn.Left := 826;

// добавить план расхода

pnlAddPlanSpend.Width := 280;

pnlAddPlanSpend.Height := 70;

pnlAddPlanSpend.Top := 17;

pnlAddPlanSpend.Left := 530;

// добавить план дохода

pnlAddPlanEarn.Width := 280;

pnlAddPlanEarn.Height := 70;

pnlAddPlanEarn.Top := 17;

pnlAddPlanEarn.Left := 826;

// план расходов

lbPlanSpend.Left := 520;

lbPlanSpend.Top := 180;

// план доходов

lbPlanEarn.Left := 520;

lbPlanEarn.Top := 770;

// кнопка добавить счет

pnlAddBill.Width := 280;

pnlAddBill.Height := 70;

pnlAddBill.Top := 17;

pnlAddBill.Left := 1120;

// тип отчета

lbReportType.Top := 130;

lbReportType.Left := 1190;

// комбобокс "тип отчета"

cbReport.Top := 120;

cbReport.Left := 1370;

cbUsersReport.Top := 120;

cbUsersReport.Left := 740;

// ----------------------------------

// помощь

// панель

pnlHelpBtns.Top := 140;

pnlHelpBtns.Left := 500;

// о программе

memoAbout.Top := 50;

memoAbout.Left := 570;

// раздел расходы

lbSpendEarn1.Top := 50;

lbSpendEarn1.Left := 570;

imgSpendEarn.Top := 130;

imgSpendEarn.Left := 660;

memoSpendEarn.Top := 230;

memoSpendEarn.Left := 570;

// таблица планов расхода

sgPlanSpend.Left := 520;

sgPlanSpend.Top := 230;

sgPlanSpend.Width := 1400;

sgPlanSpend.Height := 520;

sgPlanSpend.ColWidths[0] := 290; // ширина пользователь

sgPlanSpend.ColWidths[1] := 390; // ширина категория

sgPlanSpend.ColWidths[2] := 100; // ширина год

sgPlanSpend.ColWidths[3] := 140; // ширина месяц

sgPlanSpend.ColWidths[4] := 230; // ширина расход (план)

sgPlanSpend.ColWidths[5] := 230; // ширина расход ( факт)

sgPlanSpend.ColWidths[6] := 230; // ширина разница

lbPlanYear.Top := 130;

lbPlanYear.Left := 1480;

// таблица планов доходов

sgPlanEarn.Left := 520;

sgPlanEarn.Top := 820;

sgPlanEarn.Width := 1400;

sgPlanEarn.Height := 333;

sgPlanEarn.ColWidths[0] := 290; // ширина пользователь

sgPlanEarn.ColWidths[1] := 390; // ширина категория

sgPlanEarn.ColWidths[2] := 100; // ширина год

sgPlanEarn.ColWidths[3] := 140; // ширина месяц

sgPlanEarn.ColWidths[4] := 230; // ширина расход (план)

sgPlanEarn.ColWidths[5] := 230; // ширина расход ( факт)

sgPlanEarn.ColWidths[6] := 230; // ширина разница

// таблица отчета

sgReportTable.Left := 520;

sgReportTable.Top := 300;

sgReportTable.Width := 680;

sgReportTable.Height := 400;

sgReportTable.ColWidths[0] := 100;

sgReportTable.ColWidths[1] := 400;

// комбобокс месяц

cbMonth.Top := 120;

cbMonth.Left := 1260;

// год

editYear.Top := 120;

editYear.Left := 1550;

// диаграмма очистка

ClearDiagram(paintboxReport);

end;

procedure TfrmMain.FormResize(Sender: TObject);

begin

// увеличение длинны панели при изменении размеров главной формы

pnlTopMenu.Width := 1000 \* (ClientWidth div 700);

end;

// ------------------------------------------------------------------

// процедуры ТАБЛИЦ расходов, доходов

//процедура для фильтрации таблиц доходов/ расходов по дате

procedure TfrmMain.FilterGridForDate(StringGrid: TStringGrid);

var

i, j: Integer;

TrueDateIndex: Integer;

FalseDateCount: Integer;

begin

if cbFilterForDate.Checked = true then

begin

TrueDateIndex := 0;

FalseDateCount := 0;

for i := 1 to StringGrid.RowCount - 1 do

begin

if dtDateMoneySpend.Date = StrToDate(StringGrid.cells[3, i]) then

begin

if StringGrid.cells[3, i - 1] = '' then

begin

for j := 0 to 6 do

begin

StringGrid.cells[j, TrueDateIndex + 1] := StringGrid.cells[j, i];

end;

StringGrid.Rows[i].Clear;

inc(TrueDateIndex);

end

else

TrueDateIndex := i;

end

else

begin

StringGrid.Rows[i].Clear;

inc(FalseDateCount);

end;

end;

StringGrid.RowCount := StringGrid.RowCount - FalseDateCount;

end;

end;

procedure TfrmMain.SortGrid(SrtingGrid: TStringGrid; Column: Integer;

Row: Integer); // сортировка таблицы расходов по столбцам

var

i, j: Integer;

tmpRow: string;

Swap: Boolean;

begin

// сортируем по возрастанию

for i := 1 to SrtingGrid.RowCount - 1 do

begin

for j := i + 1 to SrtingGrid.RowCount - 1 do

begin

Swap := false;

case Column of

0, 1, 2, 4, 5: // сортировка строк (ФИО, категория, счет, Валюта)

begin

if SrtingGrid.cells[Column, i] > SrtingGrid.cells[Column, j] then

begin // обмен ячеек в столбце

Swap := true;

end;

end;

6: // сортировка по сумме

begin

if strtoint(SrtingGrid.cells[Column, i]) >

strtoint(SrtingGrid.cells[Column, j]) then

begin // обмен ячеек в столбце

Swap := true;

end;

end;

3: // сортировка по дате

begin

if StrToDateTime(SrtingGrid.cells[Column, i]) >

StrToDateTime(SrtingGrid.cells[Column, j]) then

begin // обмен ячеек в столбце

Swap := true;

end;

end;

end;

if Swap then

begin

tmpRow := SrtingGrid.Rows[i].Text;

SrtingGrid.Rows[i].Text := SrtingGrid.Rows[j].Text;

SrtingGrid.Rows[j].Text := tmpRow;

end;

end;

end;

end;

// процедуры сортировки таблицы доходов/расходов для выбранного столбца

procedure TfrmMain.sgMoneyEarnTableSelectCell(Sender: TObject;

ACol, ARow: Integer; var CanSelect: Boolean);

begin

if (ARow < 1) and fSort then // проверка нажатия на ячейки с названиями полей

begin

SortGrid(sgMoneyEarnTable, ACol, ARow); // сортировка для выбранного столбца

end;

end;

procedure TfrmMain.sgMoneySpendTableSelectCell(Sender: TObject;

ACol, ARow: Integer; var CanSelect: Boolean);

begin

if (ARow < 1) and fSort then // проверка нажатия на ячейки с названиями полей

begin

SortGrid(sgMoneySpendTable, ACol, ARow);

// сортировка для выбранного столбца

end;

end;

// процедура раскраски ячеек в соотвествии с категорией

procedure TfrmMain.sgReportTableDrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

// закрашивание первой колонны

if (ACol = 0) then

begin

case cbReport.ItemIndex of

0, 1, 2, 3, 4:

begin

case ARow of

1:

begin

sgReportTable.Canvas.Brush.Color := Rgb(116, 36, 244);

sgReportTable.Canvas.FillRect(Rect);

end;

2:

begin

sgReportTable.Canvas.Brush.Color := Rgb(238, 130, 238);

sgReportTable.Canvas.FillRect(Rect);

end;

3:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 192, 203);

sgReportTable.Canvas.FillRect(Rect);

end;

4:

begin

sgReportTable.Canvas.Brush.Color := Rgb(147, 112, 219);

sgReportTable.Canvas.FillRect(Rect);

end;

5:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 20, 147);

sgReportTable.Canvas.FillRect(Rect);

end;

6:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 191, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

7:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 0, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

8:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 255, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

9:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 128, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

10:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 215, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

11:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 165, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

12:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 0, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

13:

begin

sgReportTable.Canvas.Brush.Color := Rgb(30, 144, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

14:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 255, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

end;

end;

5, 6, 7, 8, 9:

begin

case ARow of

1:

begin

sgReportTable.Canvas.Brush.Color := Rgb(127, 255, 212);

sgReportTable.Canvas.FillRect(Rect);

end;

2:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 191, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

3:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 0, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

4:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 215, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

5:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 69, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

6:

begin

sgReportTable.Canvas.Brush.Color := Rgb(147, 112, 219);

sgReportTable.Canvas.FillRect(Rect);

end;

end;

end;

10:

begin

case ARow of

1:

begin

sgReportTable.Canvas.Brush.Color := Rgb(255, 215, 0);

sgReportTable.Canvas.FillRect(Rect);

end;

2:

begin

sgReportTable.Canvas.Brush.Color := Rgb(0, 191, 255);

sgReportTable.Canvas.FillRect(Rect);

end;

end

end;

end;

end;

end;

// загрузка данных о расходах из файла в таблицу расходов

procedure LoadStringGrid(StringGrid: TStringGrid; sgNumber: Integer;

const FileName: TFileName);

const

sgCaptionSpend: array [0 .. 6] of string = ('Категория расходов',

'Пользователь', 'Списать со счета', 'Дата', 'Примечание', 'Валюта',

'Сумма');

sgCaptionEarn: array [0 .. 6] of string = ('Источник доходов', 'Пользователь',

'На счет', 'Дата', 'Примечание', 'Валюта', 'Сумма');

sgPlanSpendName: array [0 .. 6] of string = ('Пользователь', 'Категория',

'Год', 'Месяц', 'Расход(План)', 'Расход(Факт)', 'Разница');

sgPlanEarnName: array [0 .. 6] of string = ('Пользователь', 'Источник', 'Год',

'Месяц', 'Доход(План)', 'Доход(Факт)', 'Разница');

var

f: TextFile;

Temp, i, k, j: Integer;

strTemp: String;

begin

AssignFile(f, FileName);

if not(FileExists(FileName)) then // создание файла

begin

Rewrite(f);

// заполнение шапки таблиц

with StringGrid do

begin

for j := 0 to ColCount - 1 do

begin

case sgNumber of

1:

cells[j, 0] := sgCaptionSpend[j]; // названия столбцов в таблице

2:

cells[j, 0] := sgCaptionEarn[j]; // доходы

3:

cells[j, 0] := sgPlanSpendName[j]; // планы расходы

4:

cells[j, 0] := sgPlanEarnName[j]; // план доходы

end;

end;

end;

CloseFile(f);

SaveStringGrid(StringGrid, FileName);

end

else

begin

Reset(f); // открытие файла

with StringGrid do

begin

// получение кол-ва столбцов

Readln(f, Temp);

ColCount := Temp;

// получение кол-ва строк

Readln(f, Temp);

RowCount := Temp;

// цикл по всем ячейкам и заполнение их

for i := 0 to ColCount - 1 do

for k := 0 to RowCount - 1 do

begin

Readln(f, strTemp);

cells[i, k] := strTemp;

end;

end;

CloseFile(f);

end;

end;

// сохранение данных из таблицы расходов в файл

procedure SaveStringGrid(StringGrid: TStringGrid; const FileName: TFileName);

var

f: TextFile;

i, k: Integer;

begin

AssignFile(f, FileName);

Rewrite(f);

with StringGrid do

begin

// запись кол-ва столбцов/строк

Writeln(f, ColCount);

Writeln(f, RowCount);

// цикл по всем ячейкам

for i := 0 to ColCount - 1 do

for k := 0 to RowCount - 1 do

Writeln(f, cells[i, k]);

end;

CloseFile(f);

end;

// процедура сохнаения изменений -->> добавление нового пользователя в файл

procedure SaveChanges;

var

i: byte;

begin

Rewrite(UserInfoFile); // открытие файла для записи

for i := 1 to length(ArrayOfUsers) do

begin

write(UserInfoFile, ArrayOfUsers[i - 1]);

end;

CloseFile(UserInfoFile); // закрытие файла

end;

// загрузка плана расходов в таблицу

procedure LoadPlanSpend(UserIndex: Integer; Month: Integer; Year: Integer);

begin

if length(ArrayOfUsers) > 0 then

begin

with frmMain.sgPlanSpend do

begin

cells[4, 1] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Products);

cells[4, 2] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Alcohol);

cells[4, 3] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Flat);

cells[4, 4] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Internet);

cells[4, 5] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.JKH);

cells[4, 6] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Education);

cells[4, 7] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Credit);

cells[4, 8] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Clothes);

cells[4, 9] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.DriveAway);

cells[4, 10] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Beauty);

cells[4, 11] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Health);

cells[4, 12] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Journey);

cells[4, 13] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Car);

cells[4, 14] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Spend.Another);

end;

end;

end;

procedure LoadPlanEarn(UserIndex: Integer; Month: Integer; Year: Integer);

begin

if length(ArrayOfUsers) > 0 then

begin

with frmMain.sgPlanEarn do

begin

cells[4, 1] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Earn.ZP);

cells[4, 2] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Earn.Grant);

cells[4, 3] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Earn.Compensation);

cells[4, 4] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Earn.Rent);

cells[4, 5] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Earn.Debt);

cells[4, 6] := IntToStr(ArrayOfUsers[UserIndex].Plan[Month,

Year].Earn.AnotherMoney);

end;

end;

end;

//

// ФАКТ-ПЛАН

procedure TfrmMain.CountPlanMinusFact(StringGrid: TStringGrid);

var

i: Integer;

begin

if length(ArrayOfUsers) > 0 then

begin

for i := 1 to StringGrid.RowCount - 1 do

begin

with StringGrid do

begin

cells[6, i] := IntToStr(strtoint(cells[4, i]) - strtoint(cells[5, i]));

end;

end;

end;

end;

procedure TfrmMain.ClearDiagram(PaintBox: TPaintBox);

begin

paintboxReport.Width := 0;

paintboxReport.Height := 0;

end;

procedure TfrmMain.cbFilterForDateClick(Sender: TObject);

begin

cbUsers.ItemIndex := 0;

if fMoneySpend then

begin

// загрузка данных из файла в таблицу расходов

LoadStringGrid(sgMoneySpendTable, 1, './Money spend info.txt');

FilterGridForDate(sgMoneySpendTable);

end;

if fMoneyEarn then

begin

// загрузка данных из файла в таблицу расходов

LoadStringGrid(sgMoneyEarnTable, 2, './Money earn info.txt');

FilterGridForDate(sgMoneyEarnTable);

end;

end;

procedure TfrmMain.cbMonthSelect(Sender: TObject);

var

i, UserIndex, Month, MyYear, Year, SumMoney: Integer;

begin

for i := 0 to (sgPlanSpend.RowCount - 1) do

begin

sgPlanSpend.cells[3, i + 1] := cbMonth.Text; // месяц

end;

for i := 0 to (sgPlanEarn.RowCount - 1) do

begin

sgPlanEarn.cells[3, i + 1] := cbMonth.Text; // месяц

end;

Month := frmMain.cbMonth.ItemIndex + 1;

Year := editYear.Value;

lbPlanSpend.Caption := 'План расдохов:' + ' ' + cbMonth.Text;

lbPlanEarn.Caption := 'План доходов:' + ' ' + cbMonth.Text;

UserIndex := cbUsersReport.ItemIndex;

// заполнение Расход/Доход (план)

LoadPlanSpend(UserIndex, Month, Year);

LoadPlanEarn(UserIndex, Month, Year);

// заполнение Расход/Доход ( факт)

LoadFactPlan(UserIndex);

// подсчет и заполнение ПЛАН- ФАКТ

CountPlanMinusFact(sgPlanSpend);

CountPlanMinusFact(sgPlanEarn);

end;

procedure TfrmMain.LoadFactPlan(UserIndex: Integer);

var

SearchDate: TDateTime;

MyMonth, SumMoney, AllForMonth, MonthIndex, MyYear, Month, Year: Integer;

YearNumb, MonthNumb, DayNumb, i: Word;

begin

// установка текущего месяца

DecodeDate(now, YearNumb, MonthNumb, DayNumb);

// месяц для поиска

MyMonth := cbMonth.ItemIndex + 1;

MyYear := editYear.Value;

SearchDate := EncodeDate(MyYear, MyMonth, 5);

if length(ArrayOfUsers) > 0 then

begin

for i := 1 to (sgPlanSpend.RowCount - 1) do

begin

SumMoney := 0;

SumMoney := SearchMoneyForMonth(sgMoneySpendTable, UserIndex,

sgCategorySpendPlan[i], SearchDate);

sgPlanSpend.cells[5, i] := IntToStr(SumMoney);

end;

for i := 1 to (sgPlanEarn.RowCount - 1) do

begin

SumMoney := 0;

SumMoney := SearchMoneyForMonth(sgMoneyEarnTable, UserIndex,

sgCategoryEarnPlan[i], SearchDate);

sgPlanEarn.cells[5, i] := IntToStr(SumMoney);

end;

end;

end;

procedure TfrmMain.cbReportSelect(Sender: TObject);

const

sgCategoryEarn: array [1 .. 7] of string = ('Заработная плата', 'Стипендия',

'Пенсия', 'Сдача в аренду недвижимости', 'Взял в долг',

'Доход от других источников', 'ИТОГ:(BYR)');

sgCategorySpend: array [1 .. 15] of string = ('Продукты', 'Алкоголь',

'Квартира', 'Мобильная связь, интернет', 'ЖКХ', 'Образование',

'Кредит, долг', 'Одежда', 'Проезд', 'Красота', 'Здоровье', 'Путешествия',

'Автомобиль', 'Другое', 'ИТОГ:(BYR)');

sgCategoryErnMinusSpend: array [1 .. 3] of string = ('Доход', 'Расход',

'ИТОГ:(BYR)');

var

i: Integer;

begin

paintboxReport.Width := 450;

paintboxReport.Height := 450;

Case cbReport.ItemIndex of

// расходы

0, 1, 2, 3, 4:

begin

with sgReportTable do

begin

RowCount := 16;

Height := 789;

// заполнение категорий расходов

for i := 1 to 15 do

begin

cells[1, i] := sgCategorySpend[i];

end;

end;

// перерерисовка таблицы

sgReportTable.Refresh;

end;

// доходы

5, 6, 7, 8, 9:

begin

with sgReportTable do

begin

RowCount := 8;

Height := 396;

// заполнение категорий доходов

for i := 1 to 7 do

begin

cells[1, i] := sgCategoryEarn[i];

end;

end;

// перерерисовка таблицы

sgReportTable.Refresh;

End;

// Доход минус расход

10:

begin

with sgReportTable do

begin

RowCount := 4;

Height := 200;

// заполнение категорий доходов

for i := 1 to 3 do

begin

cells[1, i] := sgCategoryErnMinusSpend[i];

end;

end;

// перерерисовка таблицы

sgReportTable.Refresh;

end;

end;

// перерисовка диаграммы

paintboxReport.Invalidate;

end;

procedure TfrmMain.cbUsersReportSelect(Sender: TObject);

var

i, UserIndex, Month, Year: Integer;

UserName: string;

begin

UserIndex := cbUsersReport.ItemIndex;

Month := frmMain.cbMonth.ItemIndex + 1;

Year := editYear.Value;

UserName := cbUsersReport.Items[cbUsersReport.ItemIndex];

for i := 0 to (sgPlanSpend.RowCount - 1) do

begin

sgPlanSpend.cells[0, i + 1].Empty;

sgPlanSpend.cells[0, i + 1] := UserName; // пользователь

end;

for i := 0 to (sgPlanEarn.RowCount - 1) do

begin

sgPlanEarn.cells[0, i + 1].Empty;

sgPlanEarn.cells[0, i + 1] := UserName; // пользователь

end;

// заполнение Расход/Доход (план)

LoadPlanSpend(UserIndex, Month, Year);

LoadPlanEarn(UserIndex, Month, Year);

// заполнение Расход/Доход ( факт)

LoadFactPlan(UserIndex);

// подсчет и заполнение ПЛАН- ФАКТ

CountPlanMinusFact(sgPlanSpend);

CountPlanMinusFact(sgPlanEarn);

end;

procedure TfrmMain.cbUsersSelect(Sender: TObject);

begin

if fMoneySpend then // таблица расходов

begin

// загрузка данных из файла в таблицу расходов

LoadStringGrid(sgMoneySpendTable, 1, './Money spend info.txt');

SortGridForUser(sgMoneySpendTable);

cbFilterForDate.Checked := false;

end;

if fMoneyEarn then // таблица доходов

begin

// загрузка данных из файла в таблицу доходов

LoadStringGrid(sgMoneyEarnTable, 2, './Money earn info.txt');

SortGridForUser(sgMoneyEarnTable);

cbFilterForDate.Checked := false;

end;

end;

procedure TfrmMain.SortGridForUser(StringGrid: TStringGrid);

var

UserIndex, i, j: byte;

TrueUserIndex: Integer; // нужные пользователи

FalseUsersCount: Integer; // кол-во удаленных строк

RowLength: Integer;

begin

RowLength := StringGrid.RowCount;

UserIndex := cbUsers.ItemIndex;

TrueUserIndex := 0;

FalseUsersCount := 0;

if (UserIndex > 0) then

begin

for i := 1 to RowLength - 1 do

begin

if cbUsers.Text = StringGrid.cells[1, i] then

begin

if StringGrid.cells[1, i - 1] = '' then

begin

for j := 0 to 6 do

begin

StringGrid.cells[j, TrueUserIndex + 1] := StringGrid.cells[j, i];

end;

StringGrid.Rows[i].Clear;

inc(TrueUserIndex);

end

else

TrueUserIndex := i;

end

else

begin

StringGrid.Rows[i].Clear;

inc(FalseUsersCount);

end;

end;

StringGrid.RowCount := StringGrid.RowCount - FalseUsersCount;

end;

end;

// ------------------------------------------------------------------

// процедуры верхнего меню

procedure TfrmMain.DeleteMoney(StringGrid: TStringGrid; TableNumb: Integer);

var

i: Integer;

begin

for i := 1 to StringGrid.RowCount - 1 do

begin

if IsCellSelected(StringGrid, 0, i) then

begin

if TableNumb = 1 then

DeleteMoneySpendFromArray(i)

else

DeleteMoneyEarnFromArray(i);

DelRow(StringGrid);

end;

end;

if TableNumb = 1 then

SaveStringGrid(sgMoneySpendTable, './Money spend info.txt')

else

SaveStringGrid(sgMoneyEarnTable, './Money earn info.txt');

end;

procedure TfrmMain.DeleteMoneySpendFromArray(i: Integer);

var

k, j: Integer;

UserName: string;

begin

for j := 0 to length(ArrayOfUsers) - 1 do

begin

UserName := ArrayOfUsers[j].Name + ' ' + ArrayOfUsers[j].Surname;

with sgMoneySpendTable do

begin

if cells[1, i] = UserName then

begin

for k := 1 to length(sgCategorySpendPlan) do

begin

if cells[0, i] = sgCategorySpendPlan[k] then

case k of

1:

ArrayOfUsers[j].MoneySpend.Products :=

ArrayOfUsers[j].MoneySpend.Products - strtoint(cells[6, i]);

2:

ArrayOfUsers[j].MoneySpend.Alcohol :=

ArrayOfUsers[j].MoneySpend.Alcohol - strtoint(cells[6, i]);

3:

ArrayOfUsers[j].MoneySpend.Flat :=

ArrayOfUsers[j].MoneySpend.Flat - strtoint(cells[6, i]);

4:

ArrayOfUsers[j].MoneySpend.Internet :=

ArrayOfUsers[j].MoneySpend.Internet - strtoint(cells[6, i]);

5:

ArrayOfUsers[j].MoneySpend.JKH := ArrayOfUsers[j].MoneySpend.JKH

- strtoint(cells[6, i]);

6:

ArrayOfUsers[j].MoneySpend.Education :=

ArrayOfUsers[j].MoneySpend.Education - strtoint(cells[6, i]);

7:

ArrayOfUsers[j].MoneySpend.Credit :=

ArrayOfUsers[j].MoneySpend.Credit - strtoint(cells[6, i]);

8:

ArrayOfUsers[j].MoneySpend.Clothes :=

ArrayOfUsers[j].MoneySpend.Clothes - strtoint(cells[6, i]);

9:

ArrayOfUsers[j].MoneySpend.DriveAway :=

ArrayOfUsers[j].MoneySpend.DriveAway - strtoint(cells[6, i]);

10:

ArrayOfUsers[j].MoneySpend.Beauty :=

ArrayOfUsers[j].MoneySpend.Beauty - strtoint(cells[6, i]);

11:

ArrayOfUsers[j].MoneySpend.Health :=

ArrayOfUsers[j].MoneySpend.Health - strtoint(cells[6, i]);

12:

ArrayOfUsers[j].MoneySpend.Journey :=

ArrayOfUsers[j].MoneySpend.Journey - strtoint(cells[6, i]);

13:

ArrayOfUsers[j].MoneySpend.Car := ArrayOfUsers[j].MoneySpend.Car

- strtoint(cells[6, i]);

14:

ArrayOfUsers[j].MoneySpend.Another :=

ArrayOfUsers[j].MoneySpend.Another - strtoint(cells[6, i]);

end;

end;

end;

end;

end;

end;

procedure TfrmMain.DeleteMoneyEarnFromArray(i: Integer);

var

k, j: Integer;

UserName: string;

begin

for j := 0 to length(ArrayOfUsers) - 1 do

begin

UserName := ArrayOfUsers[j].Name + ' ' + ArrayOfUsers[j].Surname;

with sgMoneyEarnTable do

begin

if cells[1, i] = UserName then

begin

for k := 1 to length(sgCategoryEarnPlan) do

begin

if cells[0, i] = sgCategoryEarnPlan[k] then

case k of

1:

ArrayOfUsers[j].MoneyEarn.ZP := ArrayOfUsers[j].MoneyEarn.ZP -

strtoint(cells[6, i]);

2:

ArrayOfUsers[j].MoneyEarn.Grant :=

ArrayOfUsers[j].MoneyEarn.Grant - strtoint(cells[6, i]);

3:

ArrayOfUsers[j].MoneyEarn.Compensation :=

ArrayOfUsers[j].MoneyEarn.Compensation -

strtoint(cells[6, i]);

4:

ArrayOfUsers[j].MoneyEarn.Rent := ArrayOfUsers[j].MoneyEarn.Rent

- strtoint(cells[6, i]);

5:

ArrayOfUsers[j].MoneyEarn.Debt := ArrayOfUsers[j].MoneyEarn.Debt

- strtoint(cells[6, i]);

6:

ArrayOfUsers[j].MoneyEarn.AnotherMoney :=

ArrayOfUsers[j].MoneyEarn.AnotherMoney -

strtoint(cells[6, i]);

end;

end;

end;

end;

end;

end;

procedure TfrmMain.pnlDeleteUserBtnClick(Sender: TObject);

var

i, j: Integer;

UserIndex: Integer;

begin

if length(ArrayOfUsers) > 0 then

begin

// проверка выбран ли пользователь

for j := 0 to listboxUsers.Count - 1 do

begin

if listboxUsers.Selected[j] then

begin

// индекс выбранного пользователя

UserIndex := listboxUsers.ItemIndex;

// удаление пользователя из массива

for i := UserIndex to length(ArrayOfUsers) - 2 do

begin

ArrayOfUsers[i] := ArrayOfUsers[i + 1];

end;

end;

end;

// изменение длины массива

SetLength(ArrayOfUsers, length(ArrayOfUsers) - 1);

// удаление из таблицы расходов/доходов

DeletUserFromTable(sgMoneySpendTable, UserIndex);

DeletUserFromTable(sgMoneyEarnTable, UserIndex);

// удаление из комбобоксов

cbUsers.Items.Delete(UserIndex + 1);

cbUsersReport.Items.Delete(UserIndex);

frmAddMoney.cbUsersModalUnit.Items.Delete(UserIndex);

// удаление выделенного пользователя из листбокса пользователей

listboxUsers.DeleteSelected;

// очистка таблицы счетов

sgUsersBills.RowCount := 1;

// очистка таблиц планов

for i := 1 to sgPlanSpend.RowCount - 1 do

sgPlanSpend.Rows[i].Clear;

for i := 1 to sgPlanEarn.RowCount - 1 do

sgPlanEarn.Rows[i].Clear;

// сохранение изменений в таблицах расходов/доходов

SaveStringGrid(sgMoneySpendTable, './Money spend info.txt');

SaveStringGrid(sgMoneyEarnTable, './Money earn info.txt');

// перезаписываю пользователей в файл с учетом удаленного

SaveChanges;

end;

end;

procedure TfrmMain.DeletUserFromTable(StringGrid: TStringGrid;

UserIndex: Integer);

var

i: Integer;

begin

// UserIndex:= listboxUsers.ItemIndex;

i := 1;

// цикл пока по всем строкам таблицы

while i <= StringGrid.RowCount - 1 do

begin

// если яейка с именем пользователя совпадает с выбранным для удаления пользователем

if StringGrid.cells[1, i] = cbUsers.Items[UserIndex + 1] then

begin

// выделение всей строки

StringGrid.Selection := TGridRect(Rect(0, i, 6, i));

// удаление выделенной строки

DelRow(StringGrid);

dec(i);

end;

inc(i);

end;

end;

procedure TfrmMain.pnlHelpClick(Sender: TObject);

begin

// диаграмма очистка

ClearDiagram(paintboxReport);

// видимые компоненты

HelpVisible(true);

AboutVisible(true);

// невидимые комопненты

lbUserName.Visible := false;

UsersVisible(false);

MoneySpendVisible(false);

MoneyEarnVisible(false);

ReportVisible(false);

PlansVisible(false);

FeaturesVisible(false);

UserBillVisible(false);

AboutSpendVisible(false);

AboutEarn.BevelOuter := bvRaised;

cbUsers.Visible := false;

pnlAddBtn.Visible := false;

pnlDeleteBtn.Visible := false;

end;

// процедура загрузки данных пользователей из файла в динамический массив

procedure TfrmMain.LoadUsers;

var

i: byte;

begin

Reset(UserInfoFile); // открытие файла для чтения

i := 0;

while not(eof(UserInfoFile)) do

begin

seek(UserInfoFile, i); // определение позиции в файле для чтения

SetLength(ArrayOfUsers, length(ArrayOfUsers) + 1);

// увеличение длины массива

read(UserInfoFile, ArrayOfUsers[i]); // запись данныe в массив из файла

inc(i);

end;

CloseFile(UserInfoFile); // закрытие файла

end;

// процедура проверки выделенной ячейки таблицы

function TfrmMain.IsCellSelected(StringGrid: TStringGrid;

X, Y: Longint): Boolean;

begin

Result := false;

try

if (X >= StringGrid.Selection.Left) and (X <= StringGrid.Selection.Right)

and (Y >= StringGrid.Selection.Top) and (Y <= StringGrid.Selection.Bottom)

then

Result := true;

except

end;

end;

procedure TfrmMain.listboxUsersClick(Sender: TObject);

var

i, j: Integer;

UserName: string;

BillName: string;

sumSpend, sumEarn: Integer;

begin

// очистка таблицы счетов

sgUsersBills.RowCount := 1;

// загрузка данных в таблицу счетов

with frmAddUserInfo do

LoadUsersBills(sgUsersBills);

// присвоение переменной Имя Пользователя

UserName := ArrayOfUsers[listboxUsers.ItemIndex].Name + ' ' + ArrayOfUsers

[listboxUsers.ItemIndex].Surname;

i := 0;

while (ArrayOfUsers[listboxUsers.ItemIndex].Bill[i].BillName <> '') and

(i <= 9) do

begin

// присвоение переменной Название счета

BillName := ArrayOfUsers[listboxUsers.ItemIndex].Bill[i].BillName;

sumSpend := SumAllForOneUsersBill(sgMoneySpendTable, BillName, UserName);

// расход

sumEarn := SumAllForOneUsersBill(sgMoneyEarnTable, BillName, UserName);

// доход

// запись в таблицу счетов в колонну остаток

sgUsersBills.cells[2, i + 1] := IntToStr(sumEarn - sumSpend);

// Доход-Расход

inc(i);

end;

end;

// ------------------------------------------------------------------

// функция нахождения суммы дохода/ расхода для определенного пользователя и счета

function TfrmMain.SumAllForOneUsersBill(StringGrid: TStringGrid;

BillName: string; UserName: string): Integer;

var

i: Integer;

sum: Integer;

cellBill, cellSum, cellUserName: Integer;

begin

// инициализация суммы

sum := 0;

// 3 столбец - название счета

cellBill := 2;

// 7 столбец - сумма

cellSum := 6;

// 2 столбец- название пользователя

cellUserName := 1;

// загрузка таблицы

for i := 1 to StringGrid.RowCount - 1 do

begin

if (StringGrid.cells[cellUserName, i] = UserName) and

(StringGrid.cells[cellBill, i] = BillName) then

begin

sum := sum + strtoint(StringGrid.cells[cellSum, i]);

end;

end;

// возврашаю значение суммы

Result := sum;

end;

// процедура удаления строки

procedure TfrmMain.DelRow(StringGrid: TStringGrid);

var

n: Integer;

begin

// Если осталась одна строка, операцию удаления не выполнять

if (StringGrid.RowCount = 1) then

exit;

// Сдвиг строки вверх, начиная со строки, содержащей выделенную

for n := StringGrid.Row to StringGrid.RowCount - 2 do

begin

StringGrid.Rows[n] := StringGrid.Rows[n + 1];

end;

// Удаление последней строки таблицы

StringGrid.RowCount := StringGrid.RowCount - 1;

end;

// процедура загрузки счетов пользователей в таблицу

procedure LoadUsersBills(StringGrid: TStringGrid);

var

i: Integer;

begin

i := 0;

while (ArrayOfUsers[frmMain.listboxUsers.ItemIndex].Bill[i].BillName <> '')

and (i <= 9) do

begin

StringGrid.RowCount := StringGrid.RowCount + 1;

StringGrid.cells[0, i + 1] := ArrayOfUsers[frmMain.listboxUsers.ItemIndex]

.Bill[i].BillName;

StringGrid.cells[1, i + 1] := ArrayOfUsers[frmMain.listboxUsers.ItemIndex]

.Bill[i].Valuta;

StringGrid.cells[2, i + 1] := IntToStr(0);;

inc(i);

end;

end;

// функция подсчета общей суммы всех расходов для 1 пользователя

function SumAllMoneySpend(UserNumb: Integer): Integer;

var

sum: Integer;

begin

sum := 0;

with ArrayOfUsers[UserNumb].MoneySpend do

begin

sum := (Products + Alcohol + Clothes + Flat + Health + Internet + Beauty +

DriveAway + Education + Journey + JKH + Car + Credit + Another);

end;

Result := sum;

end;

// функция подсчета общей суммы всех доходов для 1 пользователя

function SumAllMoneyEarn(UserNumb: Integer): Integer;

var

sum: Integer;

begin

sum := 0;

with ArrayOfUsers[UserNumb].MoneyEarn do

begin

sum := (ZP + Grant + Rent + Debt + Compensation + AnotherMoney);

end;

Result := sum;

end;

// функция подсчета процента категории от общей суммы

function CountProcent(var AllMoney: Integer; Category: Integer): real;

begin

Result := (Category / AllMoney) \* 2 \* pi;

end;

// ------------------------------------------------------------

// процедура отрисовки сектора диаграммы

procedure TfrmMain.sector(clr: TColor; angle1, angle2: real);

Var

x1, x2, y1, y2: Integer;

begin

// очистка цвета кисти

if (angle1 <> angle2) then

begin

paintboxReport.Canvas.Brush.Color := clr;

x1 := trunc(cos(angle1) \* paintboxReport.Width / 2 +

paintboxReport.Width / 2);

y1 := paintboxReport.Height - trunc(sin(angle1) \* paintboxReport.Height / 2

+ paintboxReport.Height / 2);

x2 := trunc(cos(angle2) \* paintboxReport.Width / 2 +

paintboxReport.Width / 2);

y2 := paintboxReport.Height - trunc(sin(angle2) \* paintboxReport.Height / 2

+ paintboxReport.Height / 2);

// рисует сектор окружности

paintboxReport.Canvas.Pie(0, 0, paintboxReport.Width, paintboxReport.Height,

x1, y1, x2, y2);

end;

end;

// процедуры рисования диаграммы

procedure TfrmMain.paintboxReportPaint(Sender: TObject);

const

sgCategorySpend: array [0 .. 14] of string = ('Продукты', 'Алкоголь',

'Квартира', 'Мобильная связь, интернет', 'ЖКХ', 'Образование',

'Кредит, долг', 'Одежда', 'Проезд', 'Красота', 'Здоровье', 'Путешествия',

'Автомобиль', 'Другое', 'ИТОГ:');

sgCategoryEarn: array [1 .. 7] of string = ('Заработная плата', 'Стипендия',

'Пенсия', 'Сдача в аренду недвижимости', 'Взял в долг',

'Доход от других источников', 'ИТОГ:');

sgCategoryErnMinusSpend: array [1 .. 2] of string = ('Доход', 'Расход');

var

AllMoneySpend, AllMoneyEarn, AllMoney: Integer;

i: Integer;

procentEarn, procentSpend, Temp: real;

SumMoney: Integer;

CurrentDate, LastMonth: TDateTime;

begin

if (length(ArrayOfUsers) > 0) then

begin

AllMoneyEarn := 0;

AllMoneySpend := 0;

AllMoney := 0;

i := cbUsersReport.ItemIndex;

// подсчет дохода за все время для 1 пользователя

AllMoneyEarn := SumAllMoneyEarn(i);

AllMoneySpend := SumAllMoneySpend(i);

case cbReport.ItemIndex of

0:

begin

// расходы за день

LoadGridForOneDay(sgMoneySpendTable, i, sgCategorySpend);

AllMoney := strtoint(sgReportTable.cells[2, 15]);

DrawAllMoneySpendForAllTime(AllMoney, i, 2);

end;

1:

begin

// расходы за неделю

LoadGridForOneWeek(sgMoneySpendTable, i, sgCategorySpend);

AllMoney := strtoint(sgReportTable.cells[2, 15]);

DrawAllMoneySpendForAllTime(AllMoney, i, 2);

end;

2:

begin

// расходы за месяц ( текущий)

CurrentDate := Date;

LoadGridForMonth(sgMoneySpendTable, i, sgCategorySpend, CurrentDate);

AllMoney := strtoint(sgReportTable.cells[2, 15]);

DrawAllMoneySpendForAllTime(AllMoney, i, 2);

end;

3:

begin

// расходы за месяц ( прошлый)

CurrentDate := Date;

LastMonth := IncMonth(CurrentDate, -1);

LoadGridForMonth(sgMoneySpendTable, i, sgCategorySpend, LastMonth);

AllMoney := strtoint(sgReportTable.cells[2, 15]);

DrawAllMoneySpendForAllTime(AllMoney, i, 2);

end;

4:

begin

// расход за все время

DrawAllMoneySpendForAllTime(AllMoneySpend, i, 1);

// отрисовка диаграммы

LoadReportGridMoneySpend(AllMoneySpend, i);

// загрузка данных в таблицу

end;

5:

begin

// доходы за день

LoadGridForOneDay(sgMoneyEarnTable, i, sgCategoryEarn);

AllMoney := strtoint(sgReportTable.cells[2, 7]);

DrawAllMoneyEarnForAllTime(AllMoney, i, 2);

end;

6:

begin

// доходы за неделю

LoadGridForOneWeek(sgMoneyEarnTable, i, sgCategoryEarn);

AllMoney := strtoint(sgReportTable.cells[2, 7]);

DrawAllMoneyEarnForAllTime(AllMoney, i, 2);

end;

7:

begin

// доходы за месяц ( текущий)

CurrentDate := Date;

LoadGridForMonth(sgMoneyEarnTable, i, sgCategoryEarn, CurrentDate);

AllMoney := strtoint(sgReportTable.cells[2, 7]);

DrawAllMoneyEarnForAllTime(AllMoney, i, 2);

end;

8:

begin

// доходы за месяц ( прошлый)

CurrentDate := Date;

LastMonth := IncMonth(CurrentDate, -1);

LoadGridForMonth(sgMoneyEarnTable, i, sgCategoryEarn, LastMonth);

AllMoney := strtoint(sgReportTable.cells[2, 7]);

DrawAllMoneyEarnForAllTime(AllMoney, i, 2);

end;

9:

begin

// доход за все время

DrawAllMoneyEarnForAllTime(AllMoneyEarn, i, 1); // отрисовка диаграммы

LoadReportGridMoneyEarn(AllMoneyEarn, i); // загрузка данных в таблицу

end;

10:

begin

// доход минус расход

AllMoney := AllMoneyEarn + AllMoneySpend;

// подсчет процентов

procentEarn := CountProcent(AllMoney, AllMoneyEarn);

procentSpend := CountProcent(AllMoney, AllMoneySpend);

Temp := 0;

// отрисовка секторов исходя из проценов( градусов угла)

sector(Rgb(255, 215, 0), 0, Temp + procentEarn);

Temp := Temp + procentEarn;

sector(Rgb(0, 191, 255), Temp, Temp + procentSpend);

with sgReportTable do

begin

cells[2, 1] := IntToStr(AllMoneyEarn);

cells[2, 2] := IntToStr(AllMoneySpend);

cells[2, 3] := IntToStr(AllMoneyEarn - AllMoneySpend);

end;

end;

end;

end;

end;

procedure TfrmMain.LoadGridForOneDay(var StringGrid: TStringGrid;

UserIndex: Integer; Category: array of string);

var

AllForDay, j, SumMoney: Integer;

begin

// инициализация переменной для суммы всех расходов за день

AllForDay := 0;

// добавлениие расходов за день в таблицу отчета по категориям

for j := 0 to (length(Category)) - 1 do

begin

SumMoney := 0;

SumMoney := SearchMoneyForToday(StringGrid, UserIndex, Category[j]);

AllForDay := AllForDay + SumMoney;

sgReportTable.cells[2, j + 1] := IntToStr(SumMoney);

end;

sgReportTable.cells[2, length(Category)] := IntToStr(AllForDay);

end;

// процедуры загрузки всех доходов/расходов в таблицу отчета

procedure TfrmMain.LoadReportGridMoneyEarn(var AllMoney: Integer;

UserIndex: Integer);

begin

with sgReportTable do

begin

cells[2, 1] := IntToStr(ArrayOfUsers[UserIndex].MoneyEarn.ZP);

cells[2, 2] := IntToStr(ArrayOfUsers[UserIndex].MoneyEarn.Grant);

cells[2, 3] := IntToStr(ArrayOfUsers[UserIndex].MoneyEarn.Compensation);

cells[2, 4] := IntToStr(ArrayOfUsers[UserIndex].MoneyEarn.Rent);

cells[2, 5] := IntToStr(ArrayOfUsers[UserIndex].MoneyEarn.Debt);

cells[2, 6] := IntToStr(ArrayOfUsers[UserIndex].MoneyEarn.AnotherMoney);

cells[2, 7] := IntToStr(AllMoney);

end;

end;

procedure TfrmMain.LoadReportGridMoneySpend(var AllMoney: Integer;

UserIndex: Integer);

begin

with sgReportTable do

begin

cells[2, 1] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Products);

cells[2, 2] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Alcohol);

cells[2, 3] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Flat);

cells[2, 4] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Internet);

cells[2, 5] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.JKH);

cells[2, 6] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Education);

cells[2, 7] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Credit);

cells[2, 8] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Clothes);

cells[2, 9] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.DriveAway);

cells[2, 10] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Beauty);

cells[2, 11] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Health);

cells[2, 12] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Journey);

cells[2, 13] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Car);

cells[2, 14] := IntToStr(ArrayOfUsers[UserIndex].MoneySpend.Another);

cells[2, 15] := IntToStr(AllMoney);

end;

end;

// функция отрисовки всех доходов пользователя за все время по категориям

procedure TfrmMain.DrawAllMoneyEarnForAllTime(var AllMoney: Integer;

UserIndex: Integer; kind: Integer);

var

procentZp, procentGrant, procentRent, procentDebt, procentCompensation,

procentAnotherMoney: real;

Temp: real;

begin

procentZp := 0;

procentGrant := 0;

procentRent := 0;

procentDebt := 0;

procentCompensation := 0;

procentAnotherMoney := 0;

if AllMoney > 0 then

// подсчет процента категории от общего дохода = градус сектора

// kind = выбор - считаю по массиву или таблице

begin

case kind of

1:

begin

// зарпалата

procentZp := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneyEarn.ZP);

// стипендия

procentGrant := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneyEarn.Grant);

// пенсия

procentCompensation := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneyEarn.Compensation);

// сдача в аренду

procentRent := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneyEarn.Rent);

// взял в долг

procentDebt := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneyEarn.Debt);

// другое

procentAnotherMoney := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneyEarn.AnotherMoney);

end;

2:

begin

// зарпалата

procentZp := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 1]));

// стипендия

procentGrant := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 2]));

// пенсия

procentCompensation := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 3]));

// сдача в аренду

procentRent := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 4]));

// взял в долг

procentDebt := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 5]));

// другое

procentAnotherMoney := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 6]));

end;

end;

Temp := 0;

// отрисовка секторов исходя из проценов( градусов угла)

// зп ( голубой)

sector(Rgb(127, 255, 212), 0, Temp + procentZp);

Temp := Temp + procentZp;

// стипендия ( синий)

sector(Rgb(0, 191, 255), Temp, Temp + procentGrant);

Temp := Temp + procentGrant;

// пенсия ( розовый)

sector(Rgb(255, 0, 255), Temp, Temp + procentCompensation);

Temp := Temp + procentCompensation;

// аренда

sector(Rgb(255, 215, 0), Temp, Temp + procentRent);

Temp := Temp + procentRent;

// долг ( ораньжевый)

sector(Rgb(255, 69, 0), Temp, Temp + procentDebt);

Temp := Temp + procentDebt;

// другоe ( феолетовый)

sector(Rgb(147, 112, 219), Temp, Temp + procentAnotherMoney);

end

else

sector(Rgb(255, 255, 255), 0, 2 \* pi);

end;

// фукнция поиска доходов/расходов по текущ. дате и нахождение их суммы

function TfrmMain.SearchMoneyForToday(StringGrid: TStringGrid;

UserIndex: Integer; Category: string): Integer;

var

i, sum: Integer;

UserName: string;

CurrentDate: TDateTime;

ConvertIndex: real;

begin

sum := 0;

// установка текущей даты

CurrentDate := Date;

// имя фамилия искомого пользователя

UserName := (ArrayOfUsers[UserIndex].Name + ' ' + ArrayOfUsers

[UserIndex].Surname);

// цикл по всей таблице

for i := 1 to StringGrid.RowCount - 1 do

begin

with StringGrid do

begin

// поиск нужного пользователя и категории и даты

if (cells[1, i] = UserName) and (cells[0, i] = Category) and

(StrToDate(cells[3, i]) = CurrentDate) then

begin

if cells[5, i] = 'USD' then

begin

ConvertIndex := 2.40;

end

else

begin

if cells[5, i] = 'RUB' then

ConvertIndex := 0.034

else

ConvertIndex := 1;

end;

sum := sum + round(strtoint(cells[6, i]) \* ConvertIndex);

end;

end;

end;

Result := sum;

end;

// фукнция поиска доходов/расходов по текущ. дате и нахождение их суммы

function TfrmMain.SearchMoneyForWeek(StringGrid: TStringGrid;

UserIndex: Integer; Category: string): Integer;

var

i, sum: Integer;

UserName: string;

ConvertIndex: real;

CurrentDate, StartDate, FinishDate, TableDate: TDateTime;

begin

// сумма

sum := 0;

// установка текущей даты

CurrentDate := Date;

case DayOfTheWeek(CurrentDate) of

1:

begin

StartDate := CurrentDate;

FinishDate := IncDay(CurrentDate, 6);

end;

2:

begin

StartDate := IncDay(CurrentDate, -1);

FinishDate := IncDay(CurrentDate, 5);

end;

3:

begin

StartDate := IncDay(CurrentDate, -2);

FinishDate := IncDay(CurrentDate, 4);

end;

4:

begin

StartDate := IncDay(CurrentDate, -3);

FinishDate := IncDay(CurrentDate, 3);

end;

5:

begin

StartDate := IncDay(CurrentDate, -5);

FinishDate := IncDay(CurrentDate, 2);

end;

6:

begin

StartDate := IncDay(CurrentDate, -6);

FinishDate := IncDay(CurrentDate, 1);

end;

7:

begin

StartDate := IncDay(CurrentDate, -7);

FinishDate := CurrentDate;

end;

end;

// имя фамилия искомого пользователя

UserName := (ArrayOfUsers[UserIndex].Name + ' ' + ArrayOfUsers

[UserIndex].Surname);

// цикл по всей таблице

for i := 1 to StringGrid.RowCount - 1 do

begin

with StringGrid do

begin

TableDate := StrToDate(cells[3, i]);

// поиск нужного пользователя и категории и даты

if (cells[1, i] = UserName) and (cells[0, i] = Category) and

(TableDate <= FinishDate) and (TableDate >= StartDate) then

begin

if cells[5, i] = 'USD' then

begin

ConvertIndex := 2.40;

end

else

begin

if cells[5, i] = 'RUB' then

ConvertIndex := 0.034

else

ConvertIndex := 1;

end;

sum := sum + round(strtoint(cells[6, i]) \* ConvertIndex);

end;

end;

end;

Result := sum;

end;

procedure TfrmMain.LoadGridForOneWeek(var StringGrid: TStringGrid;

UserIndex: Integer; Category: array of string);

var

AllForWeek, j, SumMoney: Integer;

begin

// инициализация переменной для суммы всех расходов за неделю

AllForWeek := 0;

// добавлениие расходов за день в таблицу отчета по категориям

for j := 0 to (length(Category)) - 1 do

begin

SumMoney := 0;

SumMoney := SearchMoneyForWeek(StringGrid, UserIndex, Category[j]);

AllForWeek := AllForWeek + SumMoney;

sgReportTable.cells[2, j + 1] := IntToStr(SumMoney);

end;

sgReportTable.cells[2, length(Category)] := IntToStr(AllForWeek);

end;

procedure TfrmMain.LoadGridForMonth(var StringGrid: TStringGrid;

UserIndex: Integer; Category: array of string; MyDate: TDateTime);

var

AllForMonth, j, SumMoney: Integer;

begin

// инициализация переменной для суммы всех расходов за месяц

AllForMonth := 0;

// добавлениие расходов за день в таблицу отчета по категориям

for j := 0 to (length(Category)) - 1 do

begin

SumMoney := 0;

SumMoney := SearchMoneyForMonth(StringGrid, UserIndex, Category[j], MyDate);

AllForMonth := AllForMonth + SumMoney;

sgReportTable.cells[2, j + 1] := IntToStr(SumMoney);

end;

sgReportTable.cells[2, length(Category)] := IntToStr(AllForMonth);

end;

function TfrmMain.SearchMoneyForMonth(var StringGrid: TStringGrid;

UserIndex: Integer; Category: string; MyDate: TDateTime): Integer;

var

StartDate, FinishDate, TableDate: TDateTime;

MyYear, MyMonth, myDay, i: Word;

UserName: string;

ConvertIndex: real;

sum, DayCount: Integer;

begin

DayCount := 0;

DecodeDate(MyDate, MyYear, MyMonth, myDay);

// вычисление кол-ва дней в месяце

case MyMonth of

1, 3, 5, 7, 8, 10, 12: // 31 день

begin

DayCount := 30;

end;

4, 6, 9, 11: // 30 дней

begin

DayCount := 31;

end;

2: // февраль

begin

if MyYear div 4 = 0 then

DayCount := 28

else

DayCount := 27;

end;

end;

// вычисление первого и последнего дня искомого месяца

StartDate := EncodeDate(MyYear, MyMonth, 1);

FinishDate := IncDay(StartDate, DayCount);

sum := 0;

// имя фамилия искомого пользователя

UserName := (ArrayOfUsers[UserIndex].Name + ' ' + ArrayOfUsers

[UserIndex].Surname);

// цикл по всей таблице

for i := 1 to StringGrid.RowCount - 1 do

begin

with StringGrid do

begin

TableDate := StrToDate(cells[3, i]);

// поиск нужного пользователя и категории и даты

if (cells[1, i] = UserName) and (cells[0, i] = Category) and

(TableDate < FinishDate) and (TableDate > StartDate) then

begin

if cells[5, i] = 'USD' then

begin

ConvertIndex := 2.40;

end

else

begin

if cells[5, i] = 'RUB' then

ConvertIndex := 0.034

else

ConvertIndex := 1;

end;

sum := sum + round(strtoint(cells[6, i]) \* ConvertIndex);

end;

end;

end;

Result := sum;

end;

// функция отрисовки всех расходов пользователя за все время по категориям

procedure TfrmMain.DrawAllMoneySpendForAllTime(var AllMoney: Integer;

UserIndex: Integer; kind: Integer);

var

procentProducts, procentAlcohol, procentClothes, procentFlat, procentHealth,

procentInternet, procentBeauty, procentDriveAway, procentEducation,

procentJourney, procentJKH, procentCar, procentCredit, procentAnother: real;

Temp: real;

begin

procentProducts:=0;

procentAlcohol:=0;

procentClothes:=0;

procentFlat:=0;

procentHealth:=0;

procentInternet:=0;

procentBeauty:=0;

procentDriveAway:=0;

procentEducation:=0;

procentJourney:=0;

procentJKH:=0;

procentCar:=0;

procentCredit:=0;

procentAnother:=0;

if AllMoney > 0 then

begin

case kind of

1:

begin

// подсчет процента категории от общего дохода = градус сектора

// продукты

procentProducts := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Products);

// алкоголь

procentAlcohol := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Alcohol);

// одежда

procentClothes := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Clothes);

// квартира

procentFlat := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Flat);

// здоровье

procentHealth := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Health);

// интернет, моб. связь

procentInternet := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Internet);

// красота

procentBeauty := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Beauty);

// проезд

procentDriveAway := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.DriveAway);

// образование

procentEducation := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Education);

// путешествие

procentJourney := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Journey);

// ЖКХ

procentJKH := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.JKH);

// машина

procentCar := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Car);

// кредит

procentCredit := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Credit);

// другое

procentAnother := CountProcent(AllMoney,

ArrayOfUsers[UserIndex].MoneySpend.Another);

end;

2:

begin

// продукты

procentProducts := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 1]));

// алкоголь

procentAlcohol := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 2]));

// квартира

procentFlat := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 3]));

// интернет, моб. связь

procentInternet := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 4]));

// ЖКХ

procentJKH := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 5]));

// образование

procentEducation := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 6]));

// кредит

procentCredit := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 7]));

// одежда

procentClothes := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 8]));

// проезд

procentDriveAway := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 9]));

// красота

procentBeauty := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 10]));

// здоровье

procentHealth := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 11]));

// путешествие

procentJourney := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 12]));

// машина

procentCar := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 13]));

// другое

procentAnother := CountProcent(AllMoney,

strtoint(sgReportTable.cells[2, 14]));

end;

end;

Temp := 0;

// отрисовка секторов исходя из проценов( градусов угла)

// продукты ( фиолетовый темный)

sector(Rgb(116, 36, 244), 0, Temp + procentProducts);

Temp := Temp + procentProducts;

// алкоголь ( светло-фиолетовый)

sector(Rgb(238, 130, 238), Temp, Temp + procentAlcohol);

Temp := Temp + procentAlcohol;

// квартира ( розовый)

sector(Rgb(255, 192, 203), Temp, Temp + procentFlat);

Temp := Temp + procentFlat;

// интернет, моб. связь ( феолетовый)

sector(Rgb(147, 112, 219), Temp, Temp + procentInternet);

Temp := Temp + procentInternet;

// ЖКХ (насыщенно розовый)

sector(Rgb(255, 20, 147), Temp, Temp + procentJKH);

Temp := Temp + procentJKH;

// образование ( голубой)

sector(Rgb(0, 191, 255), Temp, Temp + procentEducation);

Temp := Temp + procentEducation;

// кредит ( синий)

sector(Rgb(0, 0, 255), Temp, Temp + procentCredit);

Temp := Temp + procentCredit;

// одежда ( зеленый)

sector(Rgb(0, 255, 0), Temp, Temp + procentClothes);

Temp := Temp + procentClothes;

// проезд ( темно-зеленый)

sector(Rgb(0, 128, 0), Temp, Temp + procentDriveAway);

Temp := Temp + procentDriveAway;

// красота ( желтый)

sector(Rgb(255, 215, 0), Temp, Temp + procentBeauty);

Temp := Temp + procentBeauty;

// здоровье ( ораньжевый)

sector(Rgb(255, 165, 0), Temp, Temp + procentHealth);

Temp := Temp + procentHealth;

// путешествие ( красный)

sector(Rgb(255, 0, 0), Temp, Temp + procentJourney);

Temp := Temp + procentJourney;

// машина ( синий)

sector(Rgb(30, 144, 255), Temp, Temp + procentCar);

Temp := Temp + procentCar;

// другое ( светло-голубой)

sector(Rgb(0, 255, 255), Temp, Temp + procentAnother);

end

else

sector(Rgb(255, 255, 255), 0, 2 \* pi);

end;

procedure TfrmMain.editYearChange(Sender: TObject);

var

i, UserIndex, Month, Year: Integer;

begin

// заполнение года

for i := 0 to (sgPlanSpend.RowCount - 1) do

begin

sgPlanSpend.cells[2, i + 1] := editYear.Text;

end;

for i := 0 to (sgPlanEarn.RowCount - 1) do

begin

sgPlanEarn.cells[2, i + 1] := editYear.Text;

end;

UserIndex := cbUsersReport.ItemIndex;

Month := frmMain.cbMonth.ItemIndex + 1;

Year := editYear.Value;

// заполнение Расход/Доход (план)

LoadPlanSpend(UserIndex, Month, Year);

LoadPlanEarn(UserIndex, Month, Year);

// заполнение Расход/Доход ( факт)

LoadFactPlan(UserIndex);

// подсчет и заполнение ПЛАН- ФАКТ

CountPlanMinusFact(sgPlanSpend);

CountPlanMinusFact(sgPlanEarn);

end;

procedure GridClean(Sender: TObject);

var

hGridRect: TGridRect;

begin

hGridRect.Top := -1;

hGridRect.Left := -1;

hGridRect.Right := -1;

hGridRect.Bottom := -1;

(Sender as TStringGrid).Selection := hGridRect;

end;

end.

UnitAddUserInfo

**Модуль UnitAddUserInfo**

unit UnitAddUserInfo;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, Vcl.Grids;

type

TfrmAddUserInfo = class(TForm)

editName: TEdit;

editSurname: TEdit;

editLastname: TEdit;

pnlDontAdd: TPanel;

rgBills: TRadioGroup;

editMyBill: TEdit;

rgValuta: TRadioGroup;

gbValutCourse: TGroupBox;

lbUSD: TLabel;

lbRYB: TLabel;

editUSD: TEdit;

editRUB: TEdit;

pnlAddUserBtn2: TPanel;

procedure FormCreate(Sender: TObject);

procedure pnlAddUserBtn2Click(Sender: TObject);

procedure pnlDontAddClick(Sender: TObject);

procedure editNameClick(Sender: TObject);

procedure editSurnameClick(Sender: TObject);

procedure editLastnameClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormShow(Sender: TObject);

procedure rgBillsClick(Sender: TObject);

procedure editMyBillClick(Sender: TObject);

private

{ Private declarations }

procedure SetEditStyle;

function IsBillExist(BillName: string; j: integer): boolean;

function IsUserExit(Name, Surname, Lastname: string): boolean;

procedure AddUser;

procedure AddBill;

public

{ Public declarations }

end;

var

frmAddUserInfo: TfrmAddUserInfo;

implementation

uses UnitData,

UnitMain;

{$R \*.dfm}

// ----------------------------------------------------------------

// процедуры для установки стиля полей ввода

procedure TfrmAddUserInfo.SetEditStyle;

begin

editName.Font.Color := clGray;

editName.Text := 'Имя';

editSurname.Font.Color := clGray;

editSurname.Text := 'Фамилия';

editLastname.Font.Color := clGray;

editLastname.Text := 'Отчество';

end;

// пол ввода имени

procedure TfrmAddUserInfo.editNameClick(Sender: TObject);

begin

editName.Clear;

editName.Font.Color := clBlack;

end;

// поле ввода Фамилия

procedure TfrmAddUserInfo.editLastnameClick(Sender: TObject);

begin

editLastname.Clear;

editLastname.Font.Color := clBlack;

end;

// поле ввода Отчетства

procedure TfrmAddUserInfo.editSurnameClick(Sender: TObject);

begin

editSurname.Clear;

editSurname.Font.Color := clBlack;

end;

// поле ввода счета

procedure TfrmAddUserInfo.editMyBillClick(Sender: TObject);

begin

editMyBill.Clear;

editMyBill.Font.Color := clBlack;

end;

// обработка нажатия кнопки "отмена"

procedure TfrmAddUserInfo.pnlDontAddClick(Sender: TObject);

begin

pnlDontAdd.BevelOuter := bvLowered;

frmAddUserInfo.Close;

end;

// процедура открытия/закрытия поля ввода счетаа

procedure TfrmAddUserInfo.rgBillsClick(Sender: TObject);

begin

if rgBills.ItemIndex = 3 then

begin

editMyBill.Visible := true;

editMyBill.Clear;

editMyBill.SetFocus;

end

else

editMyBill.Visible := false;

end;

// ----------------------------------------------------------------

// процедура создания формы

procedure TfrmAddUserInfo.FormCreate(Sender: TObject);

begin

// размер и место появления

frmAddUserInfo.Top := ConstDataManipulateTop;

frmAddUserInfo.Left := ConstDataManipulateLeft;

SetEditStyle;

// поле ввода Имя

editName.Top := 80;

editName.Left := 180;

editName.Width := 360;

editName.Height := 180;

// поле ввода Фамилия

editSurname.Top := 170;

editSurname.Left := 180;

editSurname.Width := 360;

editSurname.Height := 180;

// поле ввода Отчество

editLastname.Top := 260;

editLastname.Left := 180;

editLastname.Width := 360;

editLastname.Height := 180;

// панель счета

rgBills.Top := 50;

rgBills.Left := 20;

rgBills.Width := 350;

rgBills.Height := 300;

// поле ввода счета

editMyBill.Width := 350;

// панель валюты

rgValuta.Top := 50;

rgValuta.Left := 400;

rgValuta.Width := 300;

rgValuta.Height := 210;

// курс валют

gbValutCourse.Top := 280;

gbValutCourse.Left := 400;

gbValutCourse.Width := 300;

gbValutCourse.Height := 139;

editUSD.Text := FloatToStr(USD);

editRUB.Text := FloatToStr(RUB);

end;

// процедура открытия формы

procedure TfrmAddUserInfo.FormShow(Sender: TObject);

begin

if fBill then // добавить счет

begin

frmAddUserInfo.Caption := 'Добавление счета';

// видимые компоненты

rgBills.Visible := true;

rgValuta.Visible := true;

gbValutCourse.Visible := true;

// невидимые компоненты

editName.Visible := false;

editSurname.Visible := false;

editLastname.Visible := false;

// рармер и положение формы/кнопок

frmAddUserInfo.Height := 650;

// кнопка добавить

pnlAddUserBtn2.Top := 470;

pnlAddUserBtn2.Left := 120;

// кнопка отмена

pnlDontAdd.Top := 470;

pnlDontAdd.Left := 380;

end

else // добавить пользователя

begin

frmAddUserInfo.Caption := 'Добавление пользователя';

// видимые компноненты

editName.Visible := true;

editSurname.Visible := true;

editLastname.Visible := true;

// невидимые компоненты

rgBills.Visible := false;

rgValuta.Visible := false;

gbValutCourse.Visible := false;

// кнопка добавить

pnlAddUserBtn2.Top := 380;

pnlAddUserBtn2.Left := 140;

// кнопка отмена

pnlDontAdd.Top := 380;

pnlDontAdd.Left := 400;

// рармер и положение формы/кнопок

frmAddUserInfo.Height := 600;

end;

end;

// ----------------------------------------------------------------

// процедура добавления нового пользователя или счета

procedure TfrmAddUserInfo.pnlAddUserBtn2Click(Sender: TObject);

begin

// эффект нажатой кнопки

pnlAddUserBtn2.BevelOuter := bvLowered;

if not(fBill) then

AddUser // ввод пользователя

else

AddBill; // ввод счета

end;

// добавление пользователя

procedure TfrmAddUserInfo.AddUser;

var

N: integer;

Name, Surname, Lastname: string;

begin

// проверка на ввод

Name := editName.Text;

Surname := editSurname.Text;

Lastname := editLastname.Text;

if not(IsUserExit(Name, Surname, Lastname)) then

begin

if ((Name <> 'Имя') and (Surname <> 'Фамилия') and (Lastname <> 'Отчество'))

and ((Pos(' ', Name) = 0) and (Pos(' ', Surname) = 0) and

(Pos(' ', Lastname) = 0)) and

((length(Name) > 0) and (length(Surname) > 0) and (length(Lastname) > 0))

then

begin

SetLength(ArrayOfUsers, length(ArrayOfUsers) + 1);

// учеличение размера массива на 1

N := length(ArrayOfUsers);

ArrayOfUsers[N - 1].Name := editName.Text; // добавление ФИО в массив

ArrayOfUsers[N - 1].Surname := editSurname.Text;

ArrayOfUsers[N - 1].Lastname := editLastname.Text;

// добавление в листбокс и комбобоксы

FrmMain.listboxUsers.Items.Add(editName.Text + ' ' + editSurname.Text +

' ' + editLastname.Text);

FrmMain.cbUsers.Items.Add(editName.Text + ' ' + editSurname.Text);

FrmMain.cbUsersReport.Items.Add(editName.Text + ' ' + editSurname.Text);

SaveChanges; // запись в файл

frmAddUserInfo.Close;

end

else

begin

ShowMessage('Вы ввели неверное значение, попробуйте еще раз!');

pnlAddUserBtn2.BevelOuter := bvRaised;

end;

end

else

begin

ShowMessage('Данный пользователь уже существует');

SetEditStyle;

end;

end;

// проверкаа наличия введенного ФИО в массиве

function TfrmAddUserInfo.IsUserExit(Name, Surname, Lastname: string): boolean;

var

i: integer;

begin

for i := 0 to length(ArrayOfUsers) - 1 do

begin

if (ArrayOfUsers[i].Name = Name) and (ArrayOfUsers[i].Surname = Surname) and

(ArrayOfUsers[i].Lastname = Lastname) then

result := true

else

result := false;

end;

end;

// процедура добавления счета

procedure TfrmAddUserInfo.AddBill;

var

NumbOfBills, NumbOfExistBill, j, UserIndex: integer;

SelectedBillName: string;

fFound, fAdd: boolean;

begin

fAdd := false;

if (rgBills.ItemIndex >= 0) and (rgValuta.ItemIndex >= 0) then

begin

NumbOfBills := 0;

NumbOfExistBill := 1;

UserIndex := FrmMain.listboxUsers.ItemIndex;

// подсчет количества счетов у пользователя

while (ArrayOfUsers[UserIndex].Bill[NumbOfBills].BillName <> '') and

(NumbOfBills <= 9) do

begin

inc(NumbOfBills);

end;

if (NumbOfBills <= 9) then

begin

if (rgBills.ItemIndex = 3) then

begin

SelectedBillName := editMyBill.Text;

if (Pos(' ', SelectedBillName) > 0) or (length(SelectedBillName) = 0)

then

ShowMessage('Некорректный ввод!')

else

begin

fFound := false;

for j := 0 to NumbOfBills - 1 do

begin

if (IsBillExist(SelectedBillName, j)) then

fFound := true;

end;

if not(fFound) then

begin

ArrayOfUsers[UserIndex].Bill[NumbOfBills].BillName :=

editMyBill.Text;

fAdd := true;

end

else

begin

ShowMessage('Такой счет уже существует!');

frmAddUserInfo.Close;

end;

end;

end

else

begin

SelectedBillName := rgBills.Items[rgBills.ItemIndex] + ' ' + '(' +

inttostr(NumbOfExistBill) + ')';

for j := 0 to NumbOfBills - 1 do

begin

if IsBillExist(SelectedBillName, j) then

begin

inc(NumbOfExistBill);

SelectedBillName := rgBills.Items[rgBills.ItemIndex] + ' ' + '(' +

inttostr(NumbOfExistBill) + ')';;

end;

end;

ArrayOfUsers[UserIndex].Bill[NumbOfBills].BillName := SelectedBillName;

fAdd := true;

end;

if fAdd then

begin

// добавление валюты счета

ArrayOfUsers[UserIndex].Bill[NumbOfBills].Valuta :=

rgValuta.Items[rgValuta.ItemIndex];

// очистка таблицы счетов

FrmMain.sgUsersBills.RowCount := 1;

// добавление в таблицу счетов

LoadUsersBills(FrmMain.sgUsersBills);

ShowMessage('Счет добавлен!');

end;

end

else

begin

ShowMessage('Максимальное количество счетов 10.');

frmAddUserInfo.Close;

end;

end

else

ShowMessage('Не все пункты выбраны!');

rgBills.ItemIndex := -1;

rgValuta.ItemIndex := -1;

editMyBill.Visible := false;

pnlAddUserBtn2.BevelOuter := bvRaised;

end;

// проверка наличия введенного счета в массиве

function TfrmAddUserInfo.IsBillExist(BillName: string; j: integer): boolean;

var

i: integer;

begin

if BillName = ArrayOfUsers[FrmMain.listboxUsers.ItemIndex].Bill[j].BillName

then

result := true

else

result := false;

end;

// ----------------------------------------------------------------

// процедура закрытия формы

procedure TfrmAddUserInfo.FormClose(Sender: TObject; var Action: TCloseAction);

begin

SetEditStyle;

pnlAddUserBtn2.BevelOuter := bvRaised;

pnlDontAdd.BevelOuter := bvRaised;

FrmMain.pnlAddUserBtn.BevelOuter := bvRaised;

FrmMain.pnlAddBill.BevelOuter := bvRaised;

rgBills.ItemIndex := -1;

rgValuta.ItemIndex := -1;

editMyBill.Visible := false;

pnlAddUserBtn2.SetFocus;

end;

end.

**Модуль UnitAddMoney**

unit UnitAddMoney;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls,

Vcl.ExtCtrls,

Vcl.Buttons, Vcl.WinXPickers, Vcl.Grids;

type

TfrmAddMoney = class(TForm)

lbUserName: TLabel;

cbUsersModalUnit: TComboBox;

gbCategory: TGroupBox;

lblNameOfCategory: TLabel;

editSumSpend: TEdit;

gbDate: TGroupBox;

btnAdd: TBitBtn;

btnProducts: TSpeedButton;

btnAlcohol: TSpeedButton;

btnFlat: TSpeedButton;

btnClothes: TSpeedButton;

btnDriweAway: TSpeedButton;

btnBeauty: TSpeedButton;

btnHealth: TSpeedButton;

btnInternet: TSpeedButton;

btnJKH: TSpeedButton;

btnJourney: TSpeedButton;

btnCredit: TSpeedButton;

btnAnother: TSpeedButton;

btnEducation: TSpeedButton;

btnCar: TSpeedButton;

dtCalendar: TDateTimePicker;

btnZP: TSpeedButton;

gbMoneySource: TGroupBox;

btnGrant: TSpeedButton;

btnCompensation: TSpeedButton;

btnAnotherMoney: TSpeedButton;

btnRent: TSpeedButton;

btnDebt: TSpeedButton;

lblNameOfSource: TLabel;

cbUserBills: TComboBox;

gbBills: TGroupBox;

gbNotes: TGroupBox;

editNotes: TEdit;

gbValuta: TGroupBox;

editValuta: TEdit;

procedure FormCreate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure btnProductsClick(Sender: TObject);

procedure btnAlcoholClick(Sender: TObject);

procedure btnFlatClick(Sender: TObject);

procedure editSumSpendKeyPress(Sender: TObject; var Key: Char);

procedure btnAddClick(Sender: TObject);

procedure btnInternetClick(Sender: TObject);

procedure btnJKHClick(Sender: TObject);

procedure btnEducationClick(Sender: TObject);

procedure btnCreditClick(Sender: TObject);

procedure btnClothesClick(Sender: TObject);

procedure btnDriweAwayClick(Sender: TObject);

procedure btnBeautyClick(Sender: TObject);

procedure btnHealthClick(Sender: TObject);

procedure btnJourneyClick(Sender: TObject);

procedure btnCarClick(Sender: TObject);

procedure btnAnotherClick(Sender: TObject);

procedure editSumSpendClick(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure btnZPClick(Sender: TObject);

procedure btnGrantClick(Sender: TObject);

procedure btnCompensationClick(Sender: TObject);

procedure btnAnotherMoneyClick(Sender: TObject);

procedure btnRentClick(Sender: TObject);

procedure btnDebtClick(Sender: TObject);

procedure cbUsersModalUnitChange(Sender: TObject);

procedure cbUserBillsChange(Sender: TObject);

private

procedure SetFocusOn;

procedure NameOfCategorySet;

procedure NameOfSourceSet;

procedure AddInfoToSpendTable(var BtnNumber: byte);

procedure AddInfoTоEarnTable(var BtnNumber: byte);

procedure AddPlanMoney;

procedure AddMoneySpendEarn;

{ Private declarations }

public

{ Public declarations }

end;

const

NameOfCategorys: array [1 .. 20] of string = ('Продукты', 'Алкоголь',

'Квартира', 'Мобильная связь, интернет', 'ЖКХ', 'Образование',

'Кредит, долг', 'Одежда', 'Проезд', 'Красота', 'Здоровье', 'Путешествия',

'Автомобиль', 'Другое', 'Заработная плата', 'Стипендия', 'Пенсия',

'Сдача в аренду недвижимости', 'Взял в долг', 'Доход от других источников');

var

frmAddMoney: TfrmAddMoney;

BtnNumber: byte; // номер кнопки категории

BtnWasCliked: boolean;

implementation

Uses UnitMain,

UnitData, UnitAddUserInfo;

{$R \*.dfm}

// ----------------------------------------------------------------

// процедуры установки цвета и надписи в поле сумма платежа

procedure TfrmAddMoney.SetFocusOn;

begin

editSumSpend.Font.Color := clBlack;

editSumSpend.Clear;

editSumSpend.SetFocus;

end;

// ----------------------------------------------------------------

// процедуры обработки кнопок категорий

procedure TfrmAddMoney.btnProductsClick(Sender: TObject);

begin

BtnNumber := 1;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnRentClick(Sender: TObject);

begin

BtnNumber := 18;

NameOfSourceSet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnZPClick(Sender: TObject);

begin

BtnNumber := 15;

NameOfSourceSet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnAlcoholClick(Sender: TObject);

begin

BtnNumber := 2;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnAnotherClick(Sender: TObject);

begin

BtnNumber := 14;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnAnotherMoneyClick(Sender: TObject);

begin

BtnNumber := 20;

NameOfSourceSet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnBeautyClick(Sender: TObject);

begin

BtnNumber := 10;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnCarClick(Sender: TObject);

begin

BtnNumber := 13;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnClothesClick(Sender: TObject);

begin

BtnNumber := 8;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnCompensationClick(Sender: TObject);

begin

BtnNumber := 17;

NameOfSourceSet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnCreditClick(Sender: TObject);

begin

BtnNumber := 7;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnDebtClick(Sender: TObject);

begin

BtnNumber := 19;

NameOfSourceSet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnDriweAwayClick(Sender: TObject);

begin

BtnNumber := 9;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnEducationClick(Sender: TObject);

begin

BtnNumber := 6;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnFlatClick(Sender: TObject);

begin

BtnNumber := 3;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnGrantClick(Sender: TObject);

begin

BtnNumber := 16;

NameOfSourceSet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnHealthClick(Sender: TObject);

begin

BtnNumber := 11;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnInternetClick(Sender: TObject);

begin

BtnNumber := 4;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnJKHClick(Sender: TObject);

begin

BtnNumber := 5;

NameOfCategorySet;

SetFocusOn;

end;

procedure TfrmAddMoney.btnJourneyClick(Sender: TObject);

begin

BtnNumber := 12;

NameOfCategorySet;

SetFocusOn;

end;

// ----------------------------------------------------------------

// изменение названия валюты при изменении счета

procedure TfrmAddMoney.cbUserBillsChange(Sender: TObject);

begin

editValuta.Text := ArrayOfUsers[frmAddMoney.cbUsersModalUnit.ItemIndex].Bill

[cbUserBills.ItemIndex].Valuta;

end;

// изменение названия категории расхода при клике на кнопку

procedure TfrmAddMoney.NameOfCategorySet;

var

i: integer;

begin

for i := 1 to 14 do

begin

if BtnNumber = i then

lblNameOfCategory.Caption := NameOfCategorys[i];

end;

BtnWasCliked := True;

end;

// изменение названия источника дохода при клике на кнопку

procedure TfrmAddMoney.NameOfSourceSet;

var

i: integer;

begin

for i := 15 to 20 do

begin

if BtnNumber = i then

lblNameOfSource.Caption := NameOfCategorys[i];

end;

BtnWasCliked := True;

end;

// ----------------------------------------------------------------

// заполнение выпадающего списка счетов при изм. пользователя

procedure TfrmAddMoney.cbUsersModalUnitChange(Sender: TObject);

var

i: integer;

begin

cbUserBills.Clear;

if Length(ArrayOfUsers) > 0 then

begin

i := 0;

while (ArrayOfUsers[cbUsersModalUnit.ItemIndex].Bill[i].BillName <> '') do

begin

cbUserBills.Items.Add(ArrayOfUsers[cbUsersModalUnit.ItemIndex].Bill[i]

.BillName);

inc(i);

end;

if frmAddMoney.cbUserBills.Items.Count > 0 then

cbUserBills.ItemIndex := 0;

editValuta.Text := ArrayOfUsers[frmAddMoney.cbUsersModalUnit.ItemIndex].Bill

[cbUserBills.ItemIndex].Valuta;

end;

SetFocusOn;

end;

// ----------------------------------------------------------------

// процедура создания формы

procedure TfrmAddMoney.FormCreate(Sender: TObject);

var

i: integer;

begin

// инициализация номера кнопки

BtnNumber := 0;

dtCalendar.Date := now;

// расположение компонентов

gbBills.Top := 450;

gbDate.Top := 450;

editSumSpend.Top := 600;

btnAdd.Top := 700;

// название категории

lblNameOfCategory.Caption := 'Категория';

lblNameOfSource.Caption := 'Источник дохода';

end;

// процедура показа формы

procedure TfrmAddMoney.FormShow(Sender: TObject);

begin

if (fMoneySpend) then // расходы

begin

frmAddMoney.Caption := 'Расходы (добавление)';

// видимые компоненты

cbUsersModalUnit.Visible := True;

SetFocusOn;

editSumSpend.SetFocus;

gbCategory.Visible := True;

gbCategory.Top := 90;

lbUserName.Visible := True;

gbBills.Visible := True;

gbValuta.Visible := True;

gbNotes.Visible := True;

gbDate.Visible := True;

// невидимые компоненты

gbMoneySource.Visible := false;

// расположение компнонентов

gbBills.Top := 450;

gbDate.Top := 450;

editSumSpend.Top := 750;

btnAdd.Top := 850;

gbNotes.Top := 570;

gbValuta.Top := 570;

frmAddMoney.Height := 1000;

// определение положение формы при открытии

frmAddMoney.Top := ConstDataManipulateTop;

frmAddMoney.Left := ConstDataManipulateLeft;

end;

if (fMoneyEarn) then // доходы

begin

frmAddMoney.Caption := 'Доходы (добавление)';

// видимые компоненты

SetFocusOn;

editSumSpend.SetFocus;

gbMoneySource.Visible := True;

cbUsersModalUnit.Visible := True;

lbUserName.Visible := True;

gbBills.Visible := True;

gbValuta.Visible := True;

gbNotes.Visible := True;

gbDate.Visible := True;

// невидимые компоненты

gbCategory.Visible := false;

// расположение компнонентов

gbBills.Top := 400;

gbDate.Top := 400;

editSumSpend.Top := 700;

gbNotes.Top := 527;

gbValuta.Top := 527;

btnAdd.Top := 800;

frmAddMoney.Height := 1000;

// определение положение формы при открытии

frmAddMoney.Top := ConstDataManipulateTop;

frmAddMoney.Left := ConstDataManipulateLeft;

end;

if fPlanSpend then // планы расходов

begin

frmAddMoney.Caption := 'План (расходы)';

// видимые компоненты

gbCategory.Visible := True;

SetFocusOn;

// невидимые компоненты

cbUsersModalUnit.Visible := false;

gbMoneySource.Visible := false;

lbUserName.Visible := false;

gbBills.Visible := false;

gbValuta.Visible := false;

gbNotes.Visible := false;

gbDate.Visible := false;

// расположение компнонентов

gbCategory.Top := 35;

gbBills.Top := 450;

gbDate.Top := 450;

editSumSpend.Top := 450;

btnAdd.Top := 550;

gbNotes.Top := 570;

gbValuta.Top := 570;

frmAddMoney.Height := 800;

// определение положение формы при открытии

frmAddMoney.Top := ConstDataManipulateTop;

frmAddMoney.Left := ConstDataManipulateLeft;

end;

if fPlanEarn then // планы доходов

begin

frmAddMoney.Caption := 'План (доходы)';

SetFocusOn;

// видимые компоненты

gbMoneySource.Visible := True;

// невидимые компоненты

cbUsersModalUnit.Visible := false;

gbCategory.Visible := false;

lbUserName.Visible := false;

gbBills.Visible := false;

gbValuta.Visible := false;

gbNotes.Visible := false;

gbDate.Visible := false;

// расположение компнонентов

gbCategory.Top := 35;

gbBills.Top := 450;

gbDate.Top := 450;

editSumSpend.Top := 450;

btnAdd.Top := 550;

gbNotes.Top := 570;

gbValuta.Top := 570;

frmAddMoney.Height := 800;

// определение положение формы при открытии

frmAddMoney.Top := ConstDataManipulateTop;

frmAddMoney.Left := ConstDataManipulateLeft;

end;

end;

// ----------------------------------------------------------------

// ввод ссумы платежа

procedure TfrmAddMoney.editSumSpendClick(Sender: TObject);

begin

SetFocusOn;

end;

// проверка ввода ( только цифры и backspace)

procedure TfrmAddMoney.editSumSpendKeyPress(Sender: TObject; var Key: Char);

begin

// разрешен ввод цифр,точки/ запятой, backspace

if not(Key in ['0' .. '9']) and not(Key in [#8]) then

Key := #0;

end;

// ----------------------------------------------------------------

// добавление расхода в таблицу расходов

procedure TfrmAddMoney.AddInfoToSpendTable(var BtnNumber: byte);

var

i: byte;

begin

FrmMain.cbUsers.ItemIndex := 0;

FrmMain.cbUsersSelect(FrmMain.cbUsers);

// увеличение количества строк в таблице расходов

FrmMain.sgMoneySpendTable.RowCount := FrmMain.sgMoneySpendTable.RowCount + 1;

with FrmMain.sgMoneySpendTable do

begin

cells[0, FrmMain.sgMoneySpendTable.RowCount - 1] :=

NameOfCategorys[BtnNumber]; // категория

cells[1, FrmMain.sgMoneySpendTable.RowCount - 1] :=

ArrayOfUsers[cbUsersModalUnit.ItemIndex].Name + ' ' + ArrayOfUsers

[cbUsersModalUnit.ItemIndex].Surname; // пользователь

cells[2, FrmMain.sgMoneySpendTable.RowCount - 1] :=

cbUserBills.Items[cbUserBills.ItemIndex];

cells[3, FrmMain.sgMoneySpendTable.RowCount - 1] :=

DateToStr(dtCalendar.Date);

cells[4, FrmMain.sgMoneySpendTable.RowCount - 1] := editNotes.Text;

cells[5, FrmMain.sgMoneySpendTable.RowCount - 1] :=

ArrayOfUsers[cbUsersModalUnit.ItemIndex].Bill

[cbUserBills.ItemIndex].Valuta;

cells[6, FrmMain.sgMoneySpendTable.RowCount - 1] :=

IntToStr(StrToInt(editSumSpend.Text));

end;

// сохранение данных из таблицы расходов в файл

with FrmMain do

SaveStringGrid(sgMoneySpendTable, './Money spend info.txt');

ShowMessage('Платеж добавлен!');

end;

// добавление дохода в таблицу доходов

procedure TfrmAddMoney.AddInfoTоEarnTable(var BtnNumber: byte);

var

i: integer;

begin

FrmMain.cbUsers.ItemIndex := 0;

FrmMain.cbUsersSelect(FrmMain.cbUsers);

// увеличение количества строк в таблице доходов

FrmMain.sgMoneyEarnTable.RowCount := FrmMain.sgMoneyEarnTable.RowCount + 1;

with FrmMain.sgMoneyEarnTable do

begin

cells[0, FrmMain.sgMoneyEarnTable.RowCount - 1] :=

NameOfCategorys[BtnNumber]; // категория

cells[1, FrmMain.sgMoneyEarnTable.RowCount - 1] :=

ArrayOfUsers[cbUsersModalUnit.ItemIndex].Name + ' ' + ArrayOfUsers

[cbUsersModalUnit.ItemIndex].Surname; // пользователь

cells[2, FrmMain.sgMoneyEarnTable.RowCount - 1] := cbUserBills.Items

[cbUserBills.ItemIndex];

cells[3, FrmMain.sgMoneyEarnTable.RowCount - 1] :=

DateToStr(dtCalendar.Date);

cells[4, FrmMain.sgMoneyEarnTable.RowCount - 1] := editNotes.Text;

cells[5, FrmMain.sgMoneyEarnTable.RowCount - 1] :=

ArrayOfUsers[cbUsersModalUnit.ItemIndex].Bill

[cbUserBills.ItemIndex].Valuta;

cells[6, FrmMain.sgMoneyEarnTable.RowCount - 1] :=

IntToStr(StrToInt(editSumSpend.Text));

end;

// сохранение данных из таблицы расходов в файл

with FrmMain do

SaveStringGrid(sgMoneyEarnTable, './Money earn info.txt');

ShowMessage('Платеж добавлен!');

end;

// обработка нажатия кнопки "добавить"

procedure TfrmAddMoney.btnAddClick(Sender: TObject);

var

fAdd: boolean;

begin

fAdd := false;

// проверка ввода

if (fPlanSpend) or (fPlanEarn) then // планы

begin

if BtnWasCliked = false then

ShowMessage('Выберите категорию!')

else

begin

if (Length(editSumSpend.Text) = 0) then

ShowMessage('Введите сумму плана!')

else

AddPlanMoney;

end;

end

else // доходы/расходы

begin

if cbUserBills.Items.Count > 0 then

begin

if (Length(editSumSpend.Text) = 0) then

begin

ShowMessage('Введите сумму платежа!');

end

else

begin

// проверка выбрана ли категория

if BtnWasCliked = false then

ShowMessage('Выберите категорию!')

else

fAdd := True;

end;

end

else

begin

ShowMessage('Добавьте счет!');

frmAddMoney.Close;

end;

if fAdd then

AddMoneySpendEarn;

end;

editNotes.Clear;

SetFocusOn;

end;

// ввод доходов/расходов

procedure TfrmAddMoney.AddMoneySpendEarn;

var

fAdd: boolean;

UserIndex, BillIndex, Sum: integer;

ValutConvert: double;

begin

Sum := 0;

ValutConvert := 1;

// индекс пользователя

UserIndex := cbUsersModalUnit.ItemIndex;

BillIndex := cbUserBills.ItemIndex;

// исхояд из валюты счета, выбираю переменную для конвертации в BYR

if ArrayOfUsers[UserIndex].Bill[BillIndex].Valuta = 'USD' then

ValutConvert := 2.403

else

begin

if ArrayOfUsers[UserIndex].Bill[BillIndex].Valuta = 'RUB' then

ValutConvert := 0.03;

end;

// введенная сумма

Sum := round(StrToFloat(editSumSpend.Text) \* ValutConvert);

// выбор по категории

case BtnNumber of

1:

begin

ArrayOfUsers[UserIndex].MoneySpend.Products := ArrayOfUsers[UserIndex]

.MoneySpend.Products + Sum;

end;

2:

begin

ArrayOfUsers[UserIndex].MoneySpend.Alcohol := ArrayOfUsers[UserIndex]

.MoneySpend.Alcohol + Sum;

end;

3:

begin

ArrayOfUsers[UserIndex].MoneySpend.Flat := ArrayOfUsers[UserIndex]

.MoneySpend.Flat + Sum;

end;

4:

begin

ArrayOfUsers[UserIndex].MoneySpend.Internet := ArrayOfUsers[UserIndex]

.MoneySpend.Internet + Sum;

end;

5:

begin

ArrayOfUsers[UserIndex].MoneySpend.JKH := ArrayOfUsers[UserIndex]

.MoneySpend.JKH + Sum;

end;

6:

begin

ArrayOfUsers[UserIndex].MoneySpend.Education := ArrayOfUsers[UserIndex]

.MoneySpend.Education + Sum;

end;

7:

begin

ArrayOfUsers[UserIndex].MoneySpend.Credit := ArrayOfUsers[UserIndex]

.MoneySpend.Credit + Sum;

end;

8:

begin

ArrayOfUsers[UserIndex].MoneySpend.Clothes := ArrayOfUsers[UserIndex]

.MoneySpend.Clothes + Sum;

end;

9:

begin

ArrayOfUsers[UserIndex].MoneySpend.DriveAway := ArrayOfUsers[UserIndex]

.MoneySpend.DriveAway + Sum;

end;

10:

begin

ArrayOfUsers[UserIndex].MoneySpend.Beauty := ArrayOfUsers[UserIndex]

.MoneySpend.Beauty + Sum;

end;

11:

begin

ArrayOfUsers[UserIndex].MoneySpend.Health := ArrayOfUsers[UserIndex]

.MoneySpend.Health + Sum;

end;

12:

begin

ArrayOfUsers[UserIndex].MoneySpend.Journey := ArrayOfUsers[UserIndex]

.MoneySpend.Journey + Sum;

end;

13:

begin

ArrayOfUsers[UserIndex].MoneySpend.Car := ArrayOfUsers[UserIndex]

.MoneySpend.Car + Sum;

end;

14:

begin

ArrayOfUsers[UserIndex].MoneySpend.Another := ArrayOfUsers[UserIndex]

.MoneySpend.Another + Sum;

end;

15:

begin

ArrayOfUsers[UserIndex].MoneyEarn.ZP := ArrayOfUsers[UserIndex]

.MoneyEarn.ZP + Sum;

end;

16:

begin

ArrayOfUsers[UserIndex].MoneyEarn.Grant := ArrayOfUsers[UserIndex]

.MoneyEarn.Grant + Sum;

end;

17:

begin

ArrayOfUsers[UserIndex].MoneyEarn.Compensation :=

ArrayOfUsers[UserIndex].MoneyEarn.Compensation + Sum;

end;

18:

begin

ArrayOfUsers[UserIndex].MoneyEarn.Rent := ArrayOfUsers[UserIndex]

.MoneyEarn.Rent + Sum;

end;

19:

begin

ArrayOfUsers[UserIndex].MoneyEarn.Debt := ArrayOfUsers[UserIndex]

.MoneyEarn.Debt + Sum;

end;

20:

begin

ArrayOfUsers[UserIndex].MoneyEarn.AnotherMoney :=

ArrayOfUsers[UserIndex].MoneyEarn.AnotherMoney + Sum;

end;

end;

if BtnNumber in [1 .. 14] then

AddInfoToSpendTable(BtnNumber)

else

AddInfoTоEarnTable(BtnNumber);

end;

// ввод планов

procedure TfrmAddMoney.AddPlanMoney;

var

fAdd: boolean;

UserIndex, BillIndex, Sum, Month, Year: integer;

begin

Sum := StrToInt(editSumSpend.Text);

UserIndex := FrmMain.cbUsersReport.ItemIndex;

Month := FrmMain.cbMonth.ItemIndex + 1;

Year := FrmMain.editYear.Value;

case BtnNumber of

1:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Products := Sum;

end;

2:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Alcohol := Sum;

end;

3:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Flat := Sum;

end;

4:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Internet := Sum;

end;

5:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.JKH := Sum;

end;

6:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Education := Sum;

end;

7:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Credit := Sum;

end;

8:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Clothes := Sum;

end;

9:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.DriveAway := Sum;

end;

10:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Beauty := Sum;

end;

11:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Health := Sum;

end;

12:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Journey := Sum;

end;

13:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Car := Sum;

end;

14:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Spend.Another := Sum;

end;

15:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Earn.ZP := Sum;

end;

16:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Earn.Grant := Sum;

end;

17:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Earn.Compensation := Sum;

end;

18:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Earn.Rent := Sum;

end;

19:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Earn.Debt := Sum;

end;

20:

begin

ArrayOfUsers[UserIndex].Plan[Month, Year].Earn.AnotherMoney := Sum;

end;

end;

// сохранение изменений данных пользователей в файл

if BtnNumber in [1 .. 14] then

begin

LoadPlanSpend(UserIndex, Month, Year);

ShowMessage('План расхода добавлен!');

end

else

begin

LoadPlanEarn(UserIndex, Month, Year);

ShowMessage('План дохода добавлен!');

end;

editSumSpend.Font.Color := clGray;

editSumSpend.Text := 'Сумма (BYR)';

ActiveControl := nil;

end;

// ----------------------------------------------------------------

// процедура закрытия формы

procedure TfrmAddMoney.FormClose(Sender: TObject; var Action: TCloseAction);

begin

FrmMain.pnlAddBtn.BevelOuter := bvRaised;

FrmMain.pnlAddPlanSpend.BevelOuter := bvRaised;

FrmMain.pnlAddPlanEarn.BevelOuter := bvRaised;

editNotes.Clear;

fSort := True;

btnAdd.SetFocus;

end;

end.

**Модуль UnitData**

unit UnitData;

interface

const

ConstDataManipulateTop = 570;

ConstDataManipulateLeft = 1400;

sgCategorySpendPlan: array [1 .. 14] of string = ('Продукты', 'Алкоголь',

'Квартира', 'Мобильная связь, интернет', 'ЖКХ', 'Образование',

'Кредит, долг', 'Одежда', 'Проезд', 'Красота', 'Здоровье', 'Путешествия',

'Автомобиль', 'Другое');

sgCategoryEarnPlan: array [1 .. 6] of string = ('Заработная плата',

'Стипендия', 'Пенсия', 'Сдача в аренду недвижимости', 'Взял в долг',

'Доход от других источников');

type // запись с категориями расходов

TMoneySpend = record

Products: integer;

Alcohol: integer;

Clothes: integer;

Flat: integer;

Health: integer;

Internet: integer;

Beauty: integer;

DriveAway: integer; // проезд

Education: integer;

Journey: integer;

JKH: integer; // ЖКХ

Car: integer;

Credit: integer;

Another: integer;

end;

// запись доходов

TMoneyEarn = record

ZP: integer; // зарплата

Grant: integer; // стипедния

Rent: integer; // сдача аренды в недвижимость

Debt: integer; // взял долг

Compensation: integer; // пенсия

AnotherMoney: integer; // доход от других источников

end;

TBill = record // счета

BillName: string[20];

Valuta: string[20];

end;

TPlan = record // поле для хранения планируемых расходов/ доходов по месяцам

Spend: TMoneySpend;

Earn: TMoneyEarn;

end;

TUser = record // запись пользователей

Name, Surname, Lastname: string[40]; // ФИО пользователя

MoneySpend: TMoneySpend; // расходы

MoneyEarn: TMoneyEarn; // доходы

Bill: array [0 .. 9] of TBill;

Plan: Array [1 .. 12, 2000 .. 2050] of TPlan;

end;

var

UserInfoFile: file of TUser; // файл для записи пользователей

ArrayOfUsers: array of TUser; // массив записей

USD, RUB: real;

// флаги

fMoneyEarn, fMoneySpend, fBill, fSort, fReport, fPlanSpend,

fPlanEarn: Boolean;

implementation

end.

**Модуль UnitCalculator**

unit UnitCalculator;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,

Vcl.Imaging.pngimage;

type

TfrmCalculator = class(TForm)

editVvodNumber: TEdit;

pnlCalculator: TPanel;

pnl7: TPanel;

pnl8: TPanel;

pnl9: TPanel;

pnl6: TPanel;

pnl4: TPanel;

pnl5: TPanel;

pnl1: TPanel;

pnl2: TPanel;

pnl3: TPanel;

pnl0: TPanel;

pnlPoint: TPanel;

pnlDelit: TPanel;

pnlUmnozit: TPanel;

pnlPlus: TPanel;

pnlMinus: TPanel;

pnlRovno: TPanel;

pnlDelete: TPanel;

Image1: TImage;

procedure FormCreate(Sender: TObject);

procedure pnl1Click(Sender: TObject);

procedure pnl3Click(Sender: TObject);

procedure pnl4Click(Sender: TObject);

procedure pnl5Click(Sender: TObject);

procedure pnl6Click(Sender: TObject);

procedure pnl9Click(Sender: TObject);

procedure pnl8Click(Sender: TObject);

procedure pnl7Click(Sender: TObject);

procedure pnl0Click(Sender: TObject);

procedure pnlPointClick(Sender: TObject);

procedure pnl2Click(Sender: TObject);

procedure pnlDeleteClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure pnlDelitClick(Sender: TObject);

procedure pnlPlusClick(Sender: TObject);

procedure pnlUmnozitClick(Sender: TObject);

procedure pnlMinusClick(Sender: TObject);

procedure pnlRovnoClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmCalculator: TfrmCalculator;

kod: char;

x, y, z: real;

implementation

{$R \*.dfm}

Uses UnitMain;

procedure TfrmCalculator.FormClose(Sender: TObject; var Action: TCloseAction);

begin

frmMain.pnlCalculatorBtn.BevelOuter := bvRaised;

frmCalculator.Top := 570;

frmCalculator.Left := 2000;

editVvodNumber.Text := '';

end;

procedure TfrmCalculator.FormCreate(Sender: TObject);

var

s: string;

a, b, c, i, p: integer;

op: char;

begin

frmCalculator.Top := 570;

frmCalculator.Left := 2000;

editVvodNumber.Text := '';

end;

procedure TfrmCalculator.pnl0Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '0';

end;

procedure TfrmCalculator.pnl1Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '1';

end;

procedure TfrmCalculator.pnl2Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '2';

end;

procedure TfrmCalculator.pnl3Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '3';

end;

procedure TfrmCalculator.pnl4Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '4';

end;

procedure TfrmCalculator.pnl5Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '5';

end;

procedure TfrmCalculator.pnl6Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '6';

end;

procedure TfrmCalculator.pnl7Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '7';

end;

procedure TfrmCalculator.pnl8Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '8';

end;

procedure TfrmCalculator.pnl9Click(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '9';

end;

procedure TfrmCalculator.pnlDeleteClick(Sender: TObject);

begin

editVvodNumber.Clear;

end;

procedure TfrmCalculator.pnlDelitClick(Sender: TObject);

begin

if editVvodNumber.Text <> '' then

x := strtofloat(editVvodNumber.Text);

kod := '/';

editVvodNumber.Clear;

end;

procedure TfrmCalculator.pnlMinusClick(Sender: TObject);

begin

if editVvodNumber.Text <> '' then

x := strtofloat(editVvodNumber.Text);

kod := '-';

editVvodNumber.Clear;

end;

procedure TfrmCalculator.pnlPlusClick(Sender: TObject);

begin

if editVvodNumber.Text <> '' then

x := strtofloat(editVvodNumber.Text);

kod := '+';

editVvodNumber.Clear;

end;

procedure TfrmCalculator.pnlPointClick(Sender: TObject);

begin

editVvodNumber.Text := editVvodNumber.Text + '.';

end;

procedure TfrmCalculator.pnlRovnoClick(Sender: TObject);

begin

if editVvodNumber.Text <> '' then

y := strtofloat(editVvodNumber.Text);

// в переменную y кладем editVvodNumber.text

case kod of

'+':

z := x + y;

'-':

z := x - y;

'\*':

z := x \* y;

'/':

z := x / y

end;

editVvodNumber.Text := floattostr(z);

// вывод в поле editVvodNumber значение переменной z

end;

procedure TfrmCalculator.pnlUmnozitClick(Sender: TObject);

begin

if editVvodNumber.Text <> '' then

x := strtofloat(editVvodNumber.Text);

kod := '\*';

editVvodNumber.Clear;

end;

end.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КР 1–40 01 01 225 ПЗ | | | | Пояснительная записка | | | | 122 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 951002 225 СП | | | | «Учет семейного бюджета»,А1, схема программы, чертеж | | | | **Формат А1** | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | **БГУИР КР 1-40 01 01 225 Д1** | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | «Учет семейного бюджета»  Ведомость курсовой  работы |  | | | | Лист | Листов |
| Разраб. | | Протасеня. В.Н. |  |  | Т |  | |  | 122 | 122 |
| Пров. | | Фадеева Е.Е. |  |  | Кафедра ПОИТ  гр. 951002 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |