МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет» Факультет автоматики и вычислительной техники Кафедра электронных вычислительных машин

# Дневник

по производственной практике на АО КБ Хлынов,

г. Киров

Выполнил студент группы ИВТ-31 /Категов А.Д. /

Руководитель практики от ВятГУ /Исупов К. С./ Руководитель практики от предприятия /Савина В. Н./

Киров 2023

|  |  |
| --- | --- |
| Дата | Вид деятельности |
| 24.06.2023 | Получение задания на практику |
| 25.06.2023  26.07.2023 | Выполнение задания |
| 27.07.2023 | Подготовить и оформить отчет |

Введение

Производственная практика является неотъемлемой и важной частью учебного процесса. В ходе неё не только закрепляются теоретические знания, полученные во время обучения, но и отрабатываются необходимые практические умения для эффективной работы в профессиональной сфере, осваивается инструментарий для работы и взаимодействия в команде, оттачиваются коммуникативные навыки для общения с коллегами.

Данный документ представляет собой отчет по производственной практике, проходившей в период с 24 июня по 26 июля АО ББ «Хлынов», г. Киров.

1. Общие сведения о предприятии

«Хлынов» - универсальный банк с широким спектром услуг для частных лиц и бизнеса (лицензия ЦБ РФ № 254).

На финансовом рынке с 1990 года и занимает лидирующее место в Кировской области.

Акционерное общество (АО) — предприятие, нацеленное на получение прибыли, с уставным капиталом, разделенным на акционеров. Количество акционеров определяется уставом, где закрепляются их права, правила распределения прибыли, величина влияния и другие правила работы.

Сегодня это динамично развивающийся банк с большим потенциалом и развитой сетью офисов в Кировской области. Он входит в 10-ку банков Приволжского федерального округа и в сотню ведущих банков страны по вкладам и кредитам физических лиц.

1. Общие сведения о структурном подразделении

Практика проходится в подразделении управления информационных технологий. Данное подразделение занимается банковской деятельностью.

Банковская деятельность включает в себя анализ данных клиентов, оценка клиентов и определение их платежеспособности, и выполнение обязанностей банка по предоставлению всей необходимой информации.

1. Описание выполненной работы

В соответствии с заданием от организации во время прохождения практики необходимо выполнить следующие работы:

– Разработать техническое задание приложения для расчета полной стоимости потребительского кредита

– Разработать структуру программы

– Разработать интерфейс пользователя

– Осуществить программную реализацию

3.1. Разработка технического задания

Для разработки технического задания необходимо определить:

– Назначение системы

– Цели создания системы

– Характеристику объекта автоматизации

3.1.1. Назначение системы

Приложение "Расчет ПСК" предназначено для информационного обеспечения процессов АО КБ "Хлынов", в части исполнения следующих процессов:

– произведение расчета полной стоимости кредита;

– произведение расчета ежемесячного платежа;

– построение графика погашения кредита;

– произведение расчета переплаты.

"Расчет ПСК" предполагается использовать в отделениях банка "Хлынов" при работе с клиентами, для выполнения вышеперечисленных процессов.

3.1.2. Цели создания системы

Основными целями создания приложения "Расчет ПСК" являются:

– Повышение эффективности исполнения процессов, перечисленных выше, путем сокращения операций, выполняемых «вручную»;

– Повышение качества оказания услуг за счет полноты, достоверности и удобства форматов отображения информации;

– Повышение информационной открытости и прозрачности оказания услуг АО КБ "Хлынов", повышение удобства и комфорта клиентов банка.

Для реализации поставленных целей система должна решать следующие задачи:

– Ввод данных о кредите;

– Произведение расчетов;

– Вывод информации на экран;

3.1.3. Характеристика объекта автоматизации

Объектом автоматизации являются процессы расчета информации о кредите.

Процессы расчета информации включают в себя:

– Вычисление дат платежей

– Расчет ежемесячного платежа

– Расчет графика погашения кредита

– Расчет полной стоимости кредита

– Расчет переплаты

Данная информация необходима менеджеру банка при работе с клиентами.

* + 1. Перечень подсистем, их назначение и основные характеристики

В состав приложения Расчет ПСК должны входить следующие подсистемы:

* Подсистема вычисления дат платежей
* Подсистема расчета ежемесячного платежа
* Подсистема построения графика погашения кредита
* Подсистема расчета дополнительной информации о кредите

Подсистема вычисления дат платежей предназначена для расчета дат платежей по дате получения и сроку выдачи кредита. Подсистема должна учитывать, что платеж должен выпадать только на рабочий день.

Подсистема расчета ежемесячного платежа предназначена для расчета ежемесячного платежа аннуитетного графика погашения по сумме кредита и процентной ставке.

Подсистема построения графика по гашению кредита предназначена для расчета аннуитетного графика погашения кредита и вывода его на экран.

Подсистема расчета дополнительной информации о кредите предназначена для расчета полной стоимости кредита и переплаты

3.2. Разработка структуры программы

Для разработки структурной модели необходимо создать формальные модели и на их основе определить спецификации разрабатываемого приложения.

Для этой цели была выбрана методология IDEF0. Она обладает высокой наглядностью и призвана помочь выявить основные функции и составные части проектируемой системы и, по возможности, обнаружить и устранить существенные ошибки.

Опираясь на техническое задание была построена контекстная диаграмма. Контекстная диаграмма представлена на Рисунке 1.

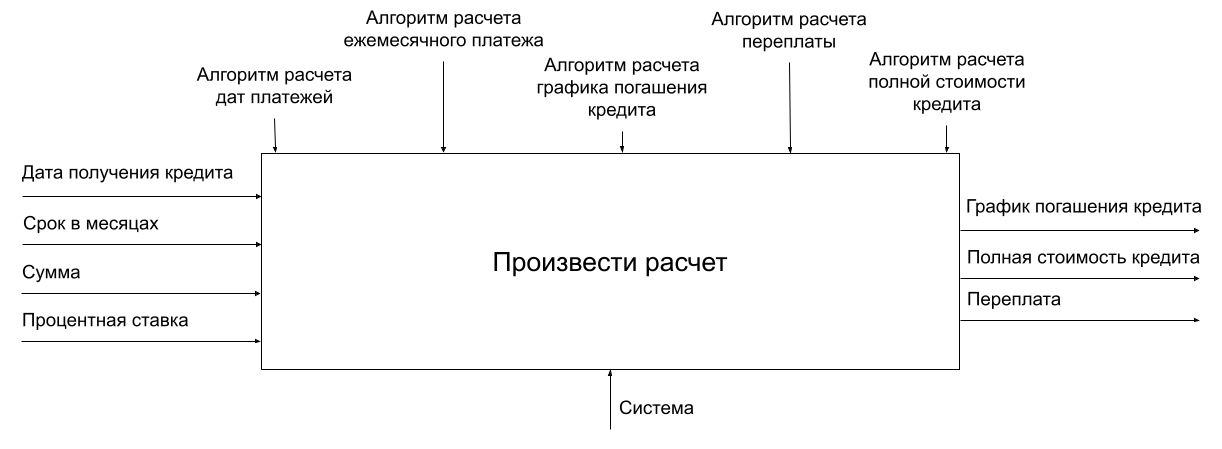


Рисунок 1 – Контекстная диаграмма IDEF0

На основе контекстной диаграммы построена детализирующая. Детализирующая диаграмма представлена на Рисунке 2.

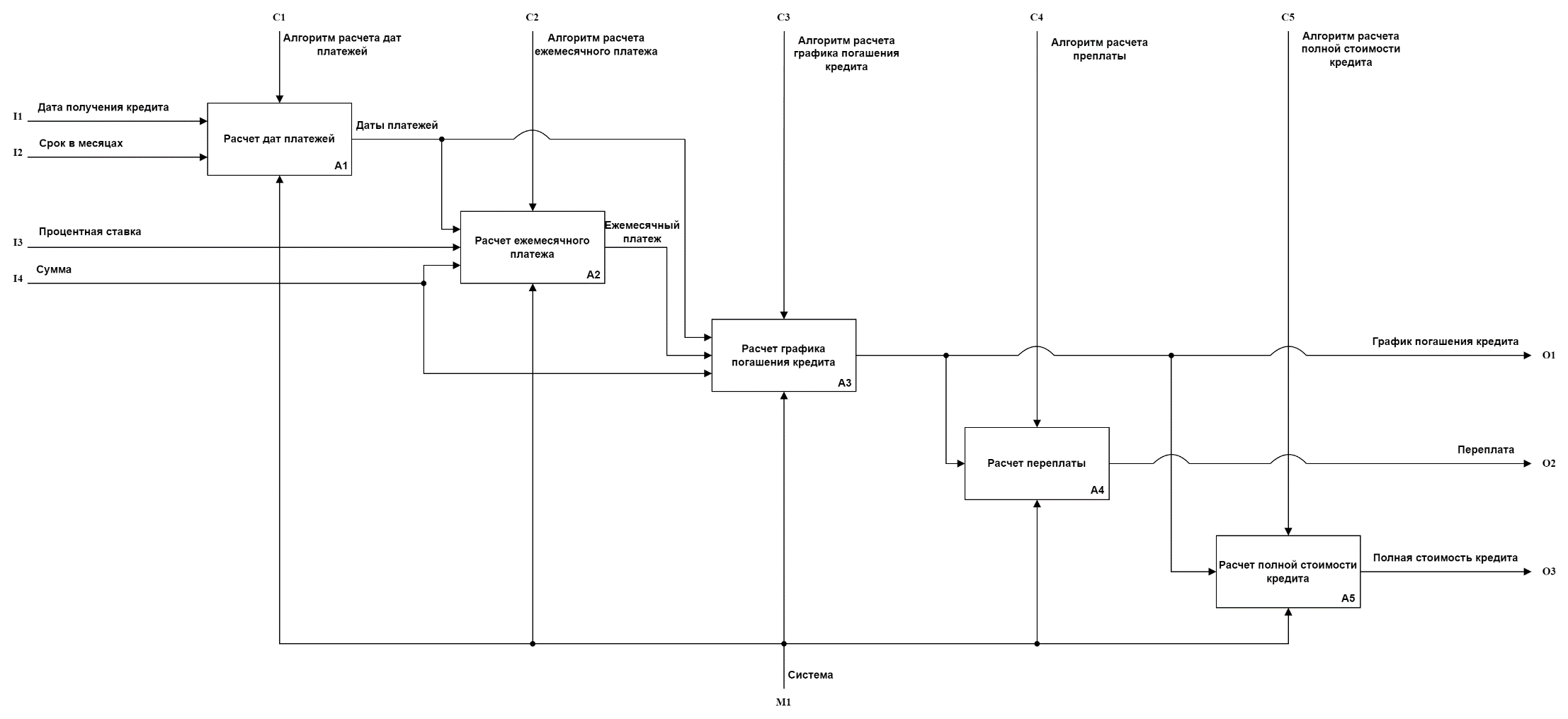


Рисунок 2 – Детализирующая диаграмма IDEF0

Для спецификации каждого процесса была выбрана методология IDEF3. Диаграмма спецификации процесса «Расчет дат платежей» представлена на Рисунке 3.

Так как кредит выдается на срок в месяцах, а платежи осуществляются один раз в месяц, то работа А1.1.1 (Расчет даты следующего платежа) выполняется смещением текущей даты на один месяц вперед.

Диаграмма спецификации процесса «Расчет ежемесячного платежа» представлена на Рисунке 4.

Диаграмма спецификации процесса «Расчет графика погашения кредита» представлена на Рисунке 5.

Диаграмма спецификации процесса «Расчет переплаты» представлена на Рисунке 6.

Диаграмма спецификации процесса «Расчет полной стоимости кредита» представлена на Рисунке 7.

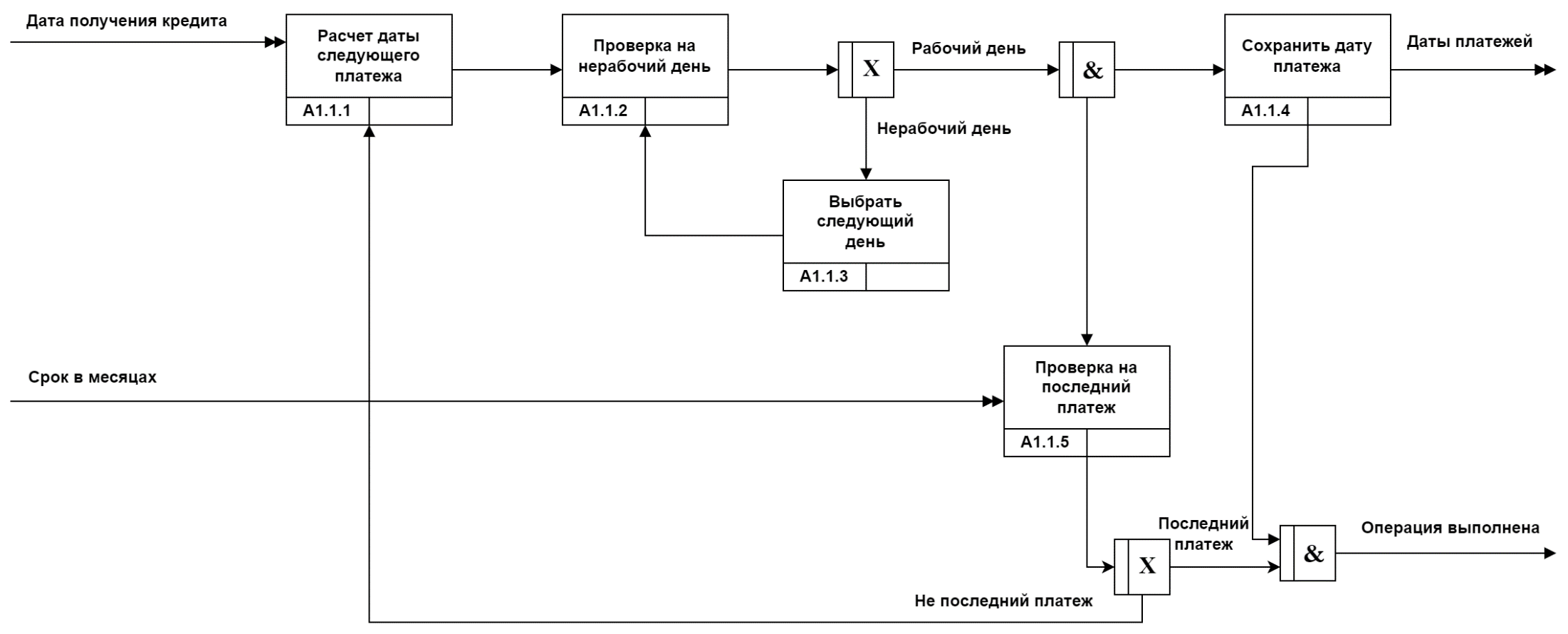


Рисунок 3 – Диаграмма Расчет дат платежей

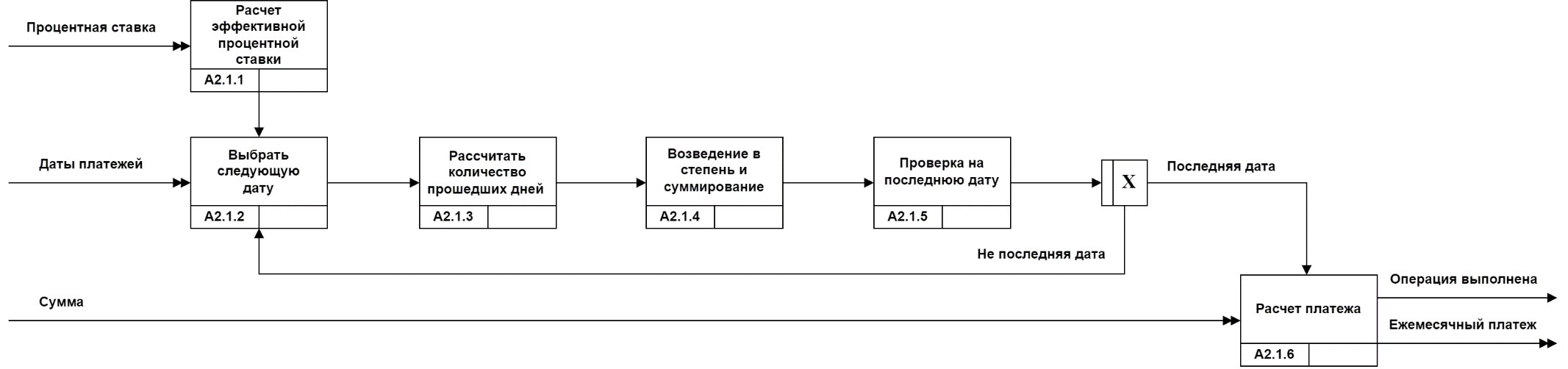


Рисунок 4 – Диаграмма Расчет ежемесячного платежа

Расчет эффективной процентной ставки I, в процессе А2.1.1, производится по формуле: I = i \ 365, где i – номинальная процентная ставка. Так как согласно ЦБ РФ период конвертации процентов равен 1 дню, а продолжительность календарного года признается равной 365 дням [2].

Для расчета ежемесячного платежа аннуитета, выплачиваемого с меньшей частотой, чем начисляются проценты, в теории процента существует формула. Она имеет следующий вид:

где k – количество периодов конвертации процентов за 1 платежный период, n – срок аннуитета измеряемый в периодах конвертации процента, i – эффективная процентная ставка за период конвертации процентов, S – сумма кредита. Однако она применима только для аннуитетов с постоянным платежным периодом k, для переменного платежного периода необходимо модифицировать формулу расчета параметра А.

Для этого заметим, что формула параметра А является частичной суммой геометрического числового ряда, тогда, при его развертывании, формула будет иметь вид:

теперь, зная что параметр k – переменный, формулу следует привести к виду:

где k – количество периодов конвертации процентов прошедших с последнего платежа, n – срок аннуитета измеряемый в периодах конвертации процента.

Таким образом, мы получаем формулу параметра А для аннуитетов с переменным платежным периодом. При этом не стоит забывать, что первым платежом считается само получение крадита, тогда k1 – это количество периодов конвертации процентов прошедших с даты взятия кредита.

Диаграмма IDEF3, описывающая сценарий процесса А3 (Расчет графика погашения кредита) представлена на рисунке 5.

В работе А3.1.3 для определения количества дней между двумя датами используется их перевод в юлианский день с последующим вычитанием. Формула перевода даты григорианского календаря в юлианский день имеет следующий вид:

где Y – григорианский год, M – григорианский месяц, D – григорианский день.

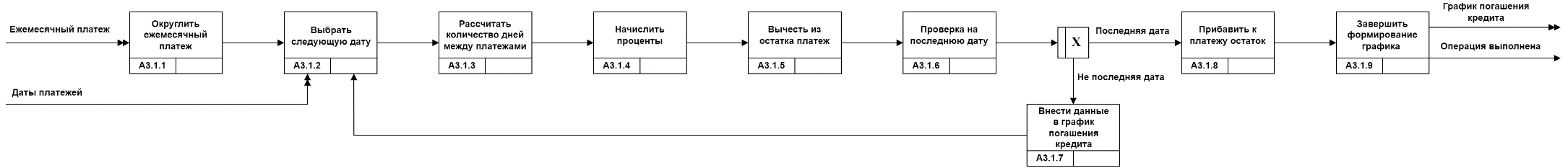


Рисунок 5 – Диаграмма Расчет графика погашения кредита

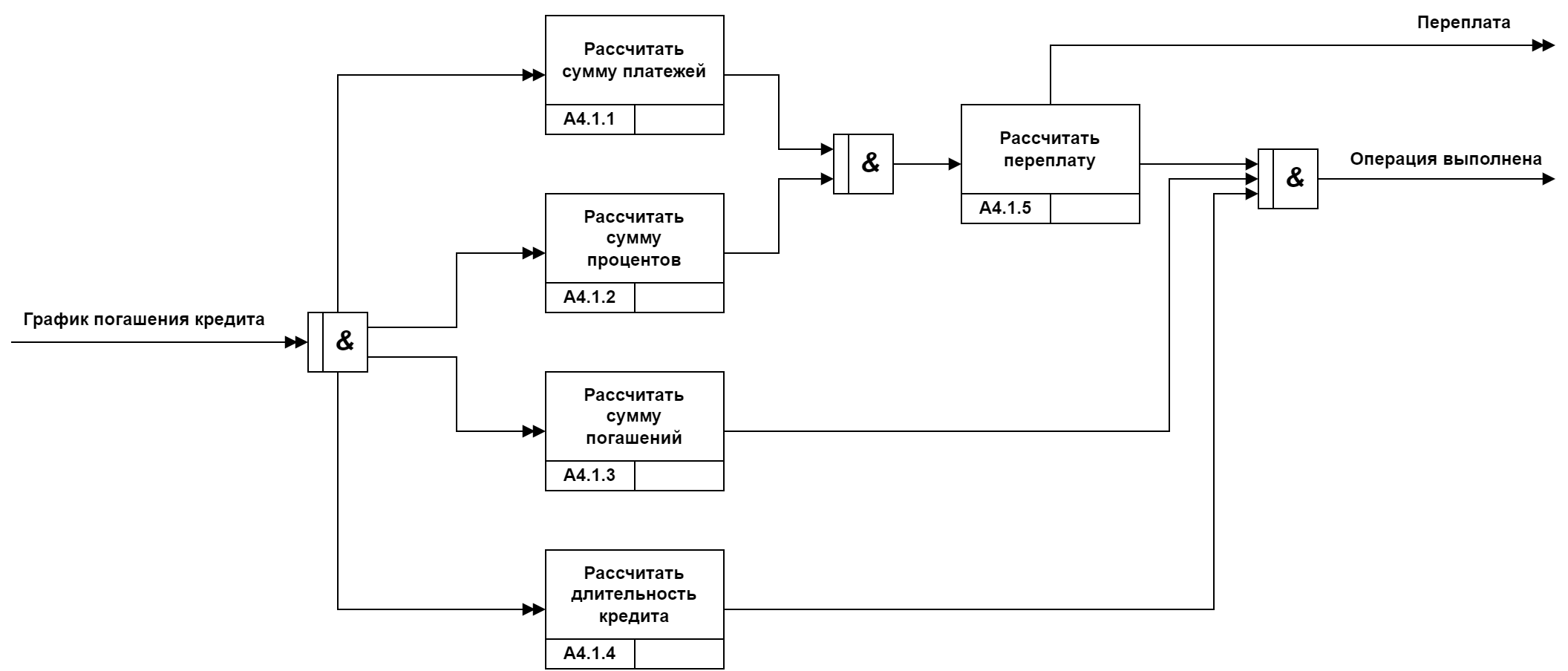


Рисунок 6 – Диаграмма Расчет переплаты

В работе А4.1.3 расчет переплаты с процентах от суммы платежей производится по формуле: G = (P \* 100) \ R, где P – сумма платежей, R – сумма процентов.

Диаграмма IDEF3, описывающая сценарий процесса А5 (Расчет ПСК) представлена на рисунке 7.

Порядок расчета ПСК участниками финансового рынка устанавливается ЦБ РФ [2]. ПСК определяется в процентах годовых по формуле:

где i – процентная ставка базового периода, C – число базовых периодов в календарном году, расчет ПСК необходимо производить с точностью до третьего знака после запятой. Продолжительность календарного года признается ЦБ РФ равной тремстам шестидесяти пяти дням [2].

где В – базовый платежный период (среднее количество периодов конвертации процентов в платежном периоде), Тk - количество периодов конвертации процентов в k-м платежном периоде (денежном потоке), n – количество денежных потоков.

Процентная ставка базового периода i определяется как наименьшее положительное решение уравнения:

где – сумма k-го денежного потока, qk – количество полных базовых периодов с момента выдачи кредита до даты k-го денежного потока, еk – срок, выраженный в долях базового периода, с момента завершения k-го базового периода до даты k-го денежного потока, n – количество денежных потоков, i – процентная ставка базового периода.

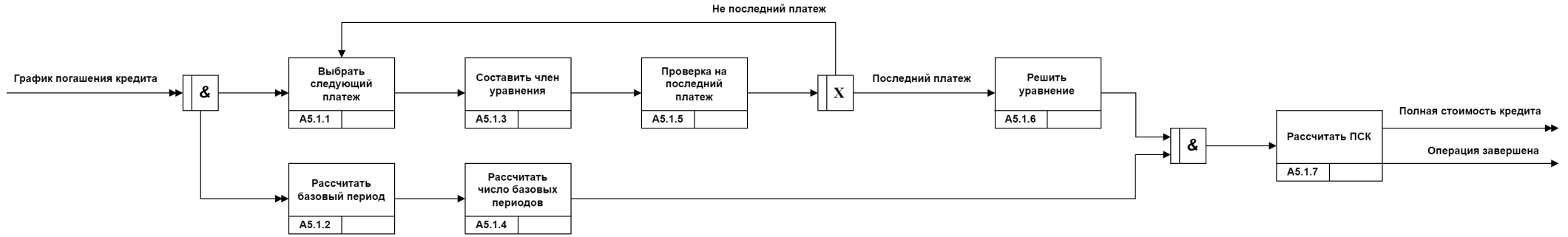


Рисунок 7 – Диаграмма Расчет полной стоимости кредита

В работе А5.1.6 решение уравнения выполняется методом Regula falsi. Данный метод имеет сверхлинейную сходимость, порядок равен примерно 1,6, и не требует вычисление производной.

Метод Ньютона, хотя и имеет квадратичную сходимость, не используется, так как вычисление производной сложной функции затруднительно и может отнять много времени. Также метод Ньютона, в отличии от Regula falsi, не имеет безусловной сходимости.

При решении одного уравнения с помощью компьютера адекватным выбором является метод деления пополам. Хотя деление пополам выполняется не так быстро, как другие методы, оно имеет гарантированую сходимость с полезной скоростью. Компьютер, используя деление пополам, решит уравнение с желаемой точностью настолько быстро, что нет необходимости экономить время, используя менее надежный метод [3].

Если компьютерной программе приходиться решать уравнения очень много раз. То время, сэкономленное более быстрыми методами, может быть значительным [3]. Исходя из этого использование метода Regula falsi является обоснованным, так как он не уступает в надежности методу деления пополам, но превосходит его в скорости сходимости. Формула метода Regula falsi имеет вид:

где – интервал изоляции, f – функция.

3.3. Разработка интерфейса пользователя

Взаимодействие пользователя с приложением должно осуществляться посредством визуального графического интерфейса. Интерфейс системы должен быть понятным и удобным, не должен быть перегружен графическими элементами. Интерфейс должен соответствовать современным эргономическим требованиям и обеспечивать удобный доступ к основным функциям и операциям системы.

Все надписи экранных форм, а также сообщения, выдаваемые пользователю должны быть на русском языке.

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей, неверным форматом или недопустимыми значениями входных данных.

Экранные формы должны проектироваться с учетом требований унификации:

– все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

– для обозначения сходных операций должны использоваться сходные графические значки, кнопки и другие управляющие элементы. Термины, используемые для обозначения типовых операций, а также последовательности действий пользователя при их выполнении, должны быть унифицированы;

Макет интерфейса приложения представлен на Рисунке 8.

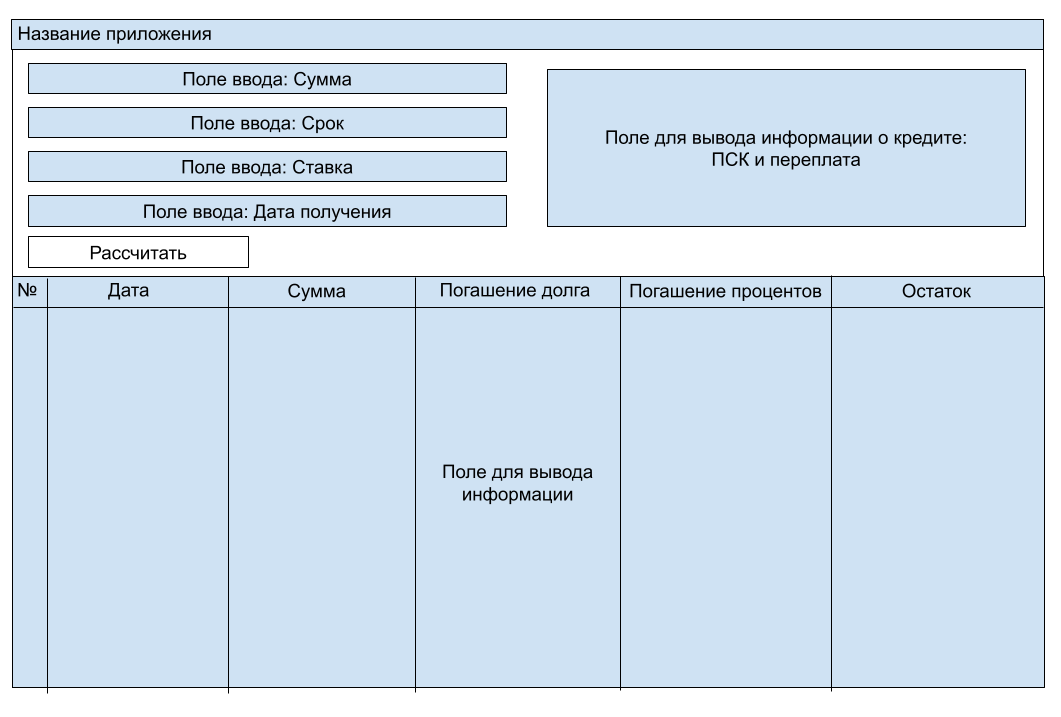


Рисунок 8 – Макет интерфейса приложения

3.4. Программная реализация

Листинг исходного кода программы Представлен в приложении А.

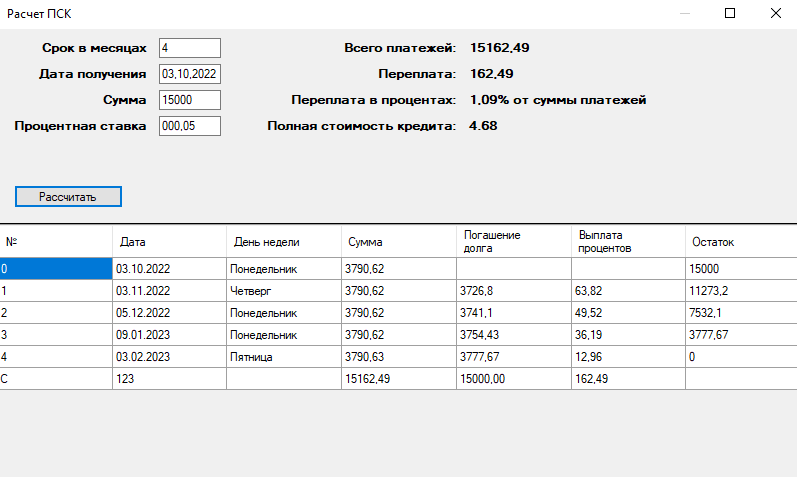
Экранная форма приложения представлена на Рисунке 9.

Рисунок 8 – Макет интерфейса приложения

**Вывод:** в ходе производственной практики было разработано приложение для расчета полной стоимости кредита. В процессе работы были получены навыки работы с языком программирования С#. Также были получены знания о функционировании современной банковской системы, финансовых операциях и основах теории перспектив. Все полученные навыки и знания были применены для разработки приложения. На основании всего сказанного выше можно сделать вывод о том, что цель достигнута, необходимые знания получены, задание выполнено верно, что доказывается полученными результатами.

Приложение А

Листинг кода

using System;

using System.Windows.Forms;

namespace PSK

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

struct TGraphic

{

public string Paymentsdate;

public int Quantitydays;

public string DayName;

public decimal Amount;

public decimal Repayment;

public decimal Percentages;

public decimal Loanbalance;

}

void СonstructionGraphic(int payments, TGraphic[] G)

{

Grid1.Rows.Clear();

Grid1.Rows.Add(0, G[0].Paymentsdate, G[0].DayName, G[0].Amount, "", "", -G[0].Loanbalance);

for (int i = 1; i <= payments; i++)

{

Grid1.Rows.Add(i, G[i].Paymentsdate, G[i].DayName, G[i].Amount, G[i].Repayment, G[i].Percentages, G[i].Loanbalance);

}

Grid1.Rows.Add("C", G[payments].Quantitydays, "", G[payments + 1].Amount, G[payments + 1].Repayment, G[payments + 1].Percentages, "");

}

TGraphic[] СalculationDates(int payments, string receipt, TGraphic[] G)

{

TDate Original = new TDate(receipt);

TDate Next = new TDate(receipt);

G[0].Paymentsdate = Original.String();

G[0].DayName = Original.DayName();

for (int i = 1; i <= payments; i++)

{

Next.day = Original.day;

Next.month = Next.month + 1;

if (Next.month == 13)

{

Next.year = Next.year + 1;

Next.month = 1;

}

int tmp = Next.DaysinMonth();

if (Next.day > tmp) { Next.day = Next.day - (Next.day - tmp); }

while (Next.Nonworking())

{

if (Next.DaysinMonth() > Next.day) { Next.day++; }

else

{

Next.day = 1;

if (Next.month != 12) { Next.month++; }

else

{

Next.month = 1;

Next.year++;

}

}

}

G[i].Paymentsdate = Next.String();

G[i].DayName = Next.DayName();

G[i].Quantitydays = Original.Quantity(Next.day, Next.month, Next.year);

}

return G;

}

decimal Payment(decimal S, decimal i, int n, TGraphic[] G)

{

decimal I = i / 365;

decimal V = 1 / (1 + I);

decimal A = 0;

for (int j = 1; j <= n; j++)

{

decimal tmp = (decimal)Math.Pow((double)V, G[j].Quantitydays);

A = A + tmp;

}

return S / A;

}

TGraphic[] СalculationGraphic(TGraphic[] G, int n, decimal i)

{

TDate date = new TDate();

decimal payment = Rounding.RoundDown(G[0].Amount, 2);

i = i / 365;

for (int j = 1; j <= n - 1; j++)

{

date.Int(G[j - 1].Paymentsdate);

decimal tmp = G[j - 1].Loanbalance \* (decimal)Math.Pow((double)i + 1, date.Quantity(G[j].Paymentsdate));

G[j].Amount = payment;

G[j].Percentages = Rounding.RoundDown(tmp - G[j - 1].Loanbalance, 2);

G[j].Repayment = Rounding.RoundDown(payment - G[j].Percentages, 2);

G[j].Loanbalance = Rounding.RoundDown(tmp - payment, 2);

}

date.Int(G[n - 1].Paymentsdate);

int yy = date.Quantity(G[n].Paymentsdate);

decimal tt = (decimal)Math.Pow((double)i + 1, yy);

decimal tmp2 = G[n - 1].Loanbalance \* (decimal)Math.Pow((double)i + 1, date.Quantity(G[n].Paymentsdate)); // <----------- тут хуйню исправить

G[n].Amount = Rounding.RoundUp(tmp2, 2);

G[n].Percentages = Rounding.RoundUp(tmp2 - G[n - 1].Loanbalance, 2);

G[n].Repayment = Rounding.RoundUp(tmp2 - G[n].Percentages, 2);

G[n].Loanbalance = 0;

G[0].Amount = payment;

return G;

}

TGraphic[] CalculationOverpayment(TGraphic[] G, int n)

{

G[0].Loanbalance = -G[0].Loanbalance;

for (int j = 1; j <= n; j++)

{

G[n + 1].Amount = G[n + 1].Amount + G[j].Amount;

G[n + 1].Repayment = G[n + 1].Repayment + G[j].Repayment;

G[n + 1].Percentages = G[n + 1].Percentages + G[j].Percentages;

}

labelPayment.Text = Convert.ToString(G[n + 1].Amount);

labelRepayment.Text = Convert.ToString(G[n + 1].Percentages);

labelPercent.Text = Convert.ToString(Rounding.RoundUp((G[n + 1].Percentages \* 100) / G[n + 1].Repayment, 2)) + "% от суммы платежей";

return G;

}

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

{

if (int.TryParse(TextBox1.Text, out int n) == false) { n = 0; }

if (decimal.TryParse(TextBox3.Text, out decimal S) == false) { S = 0; }

if (decimal.TryParse(TextBox4.Text, out decimal i) == false) { i = 0; }

TGraphic[] Graphic = new TGraphic[n + 2];

Graphic = СalculationDates(n, TextBox2.Text, Graphic);

Graphic[0].Amount = Payment(S, i, n, Graphic);

Graphic[0].Loanbalance = S;

Graphic = СalculationGraphic(Graphic, n, i);

Graphic = CalculationOverpayment(Graphic, n);

СonstructionGraphic(n, Graphic);

}

}

}

class Rounding

{

public static decimal RoundUp(decimal number, int places)

{

decimal factor = RoundFactor(places);

number \*= factor;

number = Math.Ceiling(number);

number /= factor;

return number;

}

public static decimal RoundDown(decimal number, int places)

{

decimal factor = RoundFactor(places);

number \*= factor;

number = Math.Floor(number);

number /= factor;

return number;

}

public static decimal RoundFactor(int places)

{

decimal factor = 1m;

if (places < 0)

{

places = -places;

for (int i = 0; i < places; i++) { factor /= 10m; }

}

else

{

for (int i = 0; i < places; i++) { factor \*= 10m; }

}

return factor;

}

}

public class TDate

{

public int day;

public int month;

public int year;

private int LeapYear(int Y)

{

return (1 - (Y % 4 + 2) % (Y % 4 + 1)) \* ((Y % 100 + 2) % (Y % 100 + 1)) + (1 - (Y % 400 + 2) % (Y % 400 + 1));

}

private int DaysinMonth(int M, int Y)

{

return (28 + (M + M / 8) % 2 + 2 % M + (1 + LeapYear(Y)) / M + 1 / M - LeapYear(Y) / M);

}

private int DayofWeek(int D, int M, int Y)

{

if (M > 2) { M -= 2; }

else

{

M += 10;

Y -= 1;

}

return (D + (13 \* M - 1) / 5 + Y + Y / 4 - Y / 100 + Y / 400) % 7;

}

private string DayName(int D)

{

string result = "";

switch (D)

{

case 1:

result = "Понедельник";

break;

case 2:

result = "Вторник";

break;

case 3:

result = "Среда";

break;

case 4:

result = "Четверг";

break;

case 5:

result = "Пятница";

break;

case 6:

result = "Суббота";

break;

case 0:

result = "Воскресенье";

break;

}

return result;

}

private bool Nonworking(int D, int M, int Y)

{

int name = DayofWeek(D, M, Y);

if ((name == 6) || (name == 0) ||

((M == 1) && (D < 9)) || ((M == 2) && (D == 23)) || ((M == 3) && (D == 8)) || ((M == 5) && ((D == 1) || (D == 9))) ||

((M == 6) && (D == 12)) || ((M == 11) && (D == 4))) { return true; }

else { return false; }

}

private int JulianDay(int D, int M, int Y)

{

return ((1461 \* (Y + 4800 + (M - 14) / 12)) / 4 + (367 \* (M - 2 - 12 \* ((M - 14) / 12))) / 12 - (3 \* ((Y + 4900 + (M - 14) / 12) / 100)) / 4 + D - 32075);

}

public int Quantity(int D, int M, int Y)

{

return (JulianDay(D, M, Y) - JulianDay(day, month, year));

}

public int Quantity(string date)

{

date = date.Replace(".", "");

if ((date.Length > 1) && (int.TryParse(date.Substring(0, 2), out int D) == true)) ;

else { D = 1; }

if ((date.Length > 2) && (int.TryParse(date.Substring(2, 2), out int M) == true)) ;

else { M = 1; }

if ((date.Length > 3) && (int.TryParse(date.Substring(4), out int Y) == true)) ;

else { Y = 1; }

return (JulianDay(D, M, Y) - JulianDay(day, month, year));

}

public string DayName()

{

return DayName(DayofWeek(day, month, year));

}

public void Int(string date)

{

date = date.Replace(".", "");

if ((date.Length > 1) && (int.TryParse(date.Substring(0, 2), out day) == true)) ;

else { day = 1; }

if ((date.Length > 2) && (int.TryParse(date.Substring(2, 2), out month) == true)) ;

else { month = 1; }

if ((date.Length > 3) && (int.TryParse(date.Substring(4), out year) == true)) ;

else { year = 1; }

}

public string String()

{

string tmpD, tmpM;

if (month < 10) { tmpM = "0" + Convert.ToString(month); }

else { tmpM = Convert.ToString(month); }

if (day < 10) { tmpD = "0" + Convert.ToString(day); }

else { tmpD = Convert.ToString(day); }

return (tmpD + "." + tmpM + "." + Convert.ToString(year));

}

public bool Nonworking()

{

return Nonworking(day, month, year);

}

public int DaysinMonth()

{

return DaysinMonth(month, year);

}

public TDate(string DS)

{

Int(DS);

}

public TDate(int D = 1, int M = 1, int Y = 1)

{

day = D;

month = M;

year = Y;

}

}