**Введение**

На практику по программированию была поставлена тема по предметной области «Учет внутриофисных расходов».

Цель проекта заключается в создании консольного приложения.

Приведем краткое описание разделов пояснительной записки.

Первый раздел под названием «Анализ предметной области и формулировка требований к программе» предназначен для ознакомления с предметной областью и инструментами разработки, которые поспособствовали созданию данного проекта. В этом разделе можно узнать о круге задач, которые должны быть автоматизированы; об необходимости компьютерной обработки информации или процессов;

Раздел «Проектирование» имеет цель показать варианты использования системы в виде диаграммы. Также в разделе представлена диаграмма деятельности, чтобы показать один из процессов, выполняемый данным приложением. Еще раздел содержит описание тестов, проведенных в ходе разработки приложения.

Третий раздел под названием «Построение программы» познакомит с классами, которые реализуют все функции приложения, их атрибутами и методами. Присутствует диаграмма классов, для понимания зависимостей между ними.

Четвертый раздел «Тестирование» посвящен результатам тестирования приложения.

Пятый раздел – это «Применение». В данном разделе описана область применения, инструкция по установке приложения и его запуска, а также показаны функции и системные требования.

В разделе «Заключение» предоставлены выводы о проделанной работе.

В «Списке использованных источников» приводится список используемых при разработке источников.

В приложении представлен листинг программы

**1 Анализ предметной области и формулировка требований к программе**

* 1. **Исследование предметной области**

Наименование задачи: Предметная область «Учет внутриофисных расходов».

Определение круга задач, которые должны быть автоматизированы: задачей проекта

является ежемесячное отслеживание внутриофисных расходов и совершение закупок от лица сотрудников, не превышая предельную сумму затрат.

Обоснование необходимости компьютерной обработки информации или процессов: компьютерный процесс заключается в записи данных о покупке(отдел, дата, сумма, вид расхода) и выводе информации о затраченных денежных средствах за текущий месяц; приложение не должно допускать превышение затраченных средств в месяц в соответствии с предельной суммой закупок в зависимости от вида расходов, приложение должно учитывать введенную сумму закупки и уже потраченные средства на данный вид расходов в данном месяце, для расчета итоговой суммы затрат, которая будет сравниваться с предельной суммой; приложение будет выводить информацию о том, на сколько денежных единиц превышает сумма закупок, если такая ситуация произойдет.

**1.2Инструменты разработки**

Для оптимального функционирования разрабатываемого приложения необходимо наличие у компьютера следующих параметров:

− процессор ARM64 или x64; Рекомендуется четырехядерный или более качественный;

− от 4 GB RAM;

− монитор с минимальным разрешением 1366х768.

Требования к операционной системе:

− Windows 11 версии 21H2 или более поздних версий;

− Windows 10 версии 1909 или более поздних версий;

Для разработки данного проекта будет использоваться среда разработки Microsoft Visual Studio 2022, которая является целевой средой для создания консольного приложения.

Иные инструменты, используемые при разработке и написании сопутствующей документации:

- Microsoft Word 2016 – для написания документации к программному продукту;

- Microsoft Visio 2016 – для создания графической части и разработки UML-диаграмм;

- Google-браузер – для нахождения информации, картинок;

- Smart Install Maker – нужен для создания инсталлятора.

**2 Проектирование**

**2.1 Диаграмма вариантов использования**

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. Актером или действующим лицом является любая сущность, взаимодействующая с системой извне (Рисунок 1). Это может быть, как живое существо, так и любая друга система, которая может служить источником воздействия на моделируемую систему так, как определяет сам разработчик.

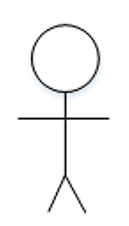


Рисунок 1 – Актер

Вариант использования является стандартным языком UML и применяется для спецификаций общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название (Рисунок 2).

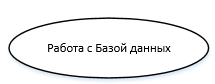


Рисунок 2 – Вариант использования

Отношение ассоциации является главным понятием языка UML и используется при построении всех графических моделей. Оно служит для обозначения роли актера и отдельном варианте использования. На диаграмме отношение ассоциации обозначается сплошной линией между актером и вариантом использования (Рисунок 3).

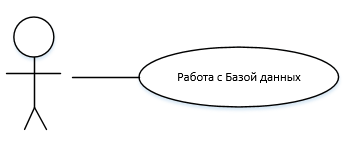


Рисунок 3 – Обозначение отношения ассоциации

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов использования (Рисунок 4).

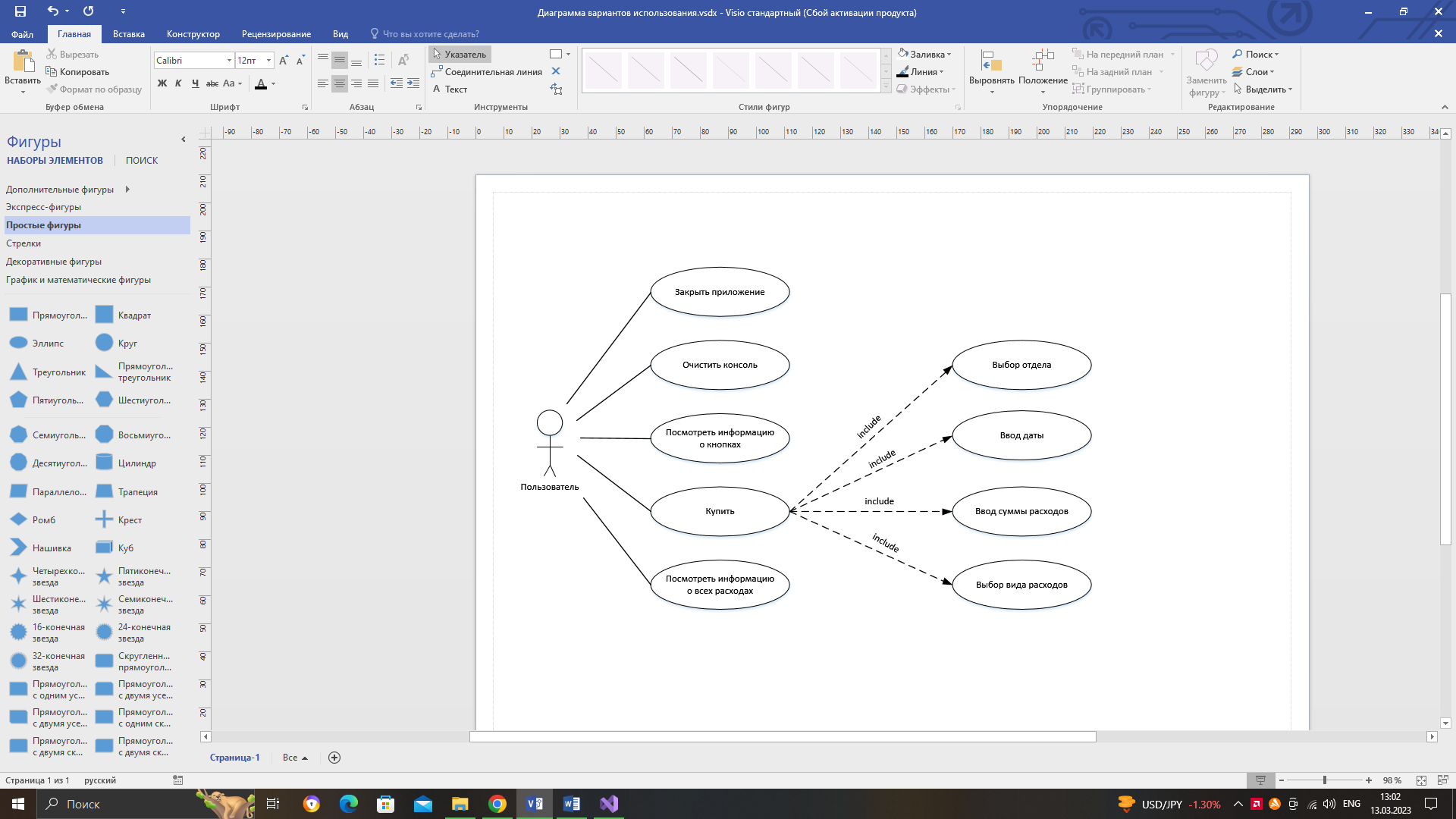


Рисунок 4 – Диаграмма вариантов использования

**2.2 Диаграмма деятельности**

В ходе создания проекта требовалось создать UML-диаграмму, что помогла бы понять определенный процесс самого консольного приложения. Этой диаграммой является диаграмма деятельности (Рисунок 5). Диаграмма деятельности ([анг.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык) activity diagram) — UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Диаграмма состояний — это, по существу, диаграмма состояний из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до бизнеспроцессов.

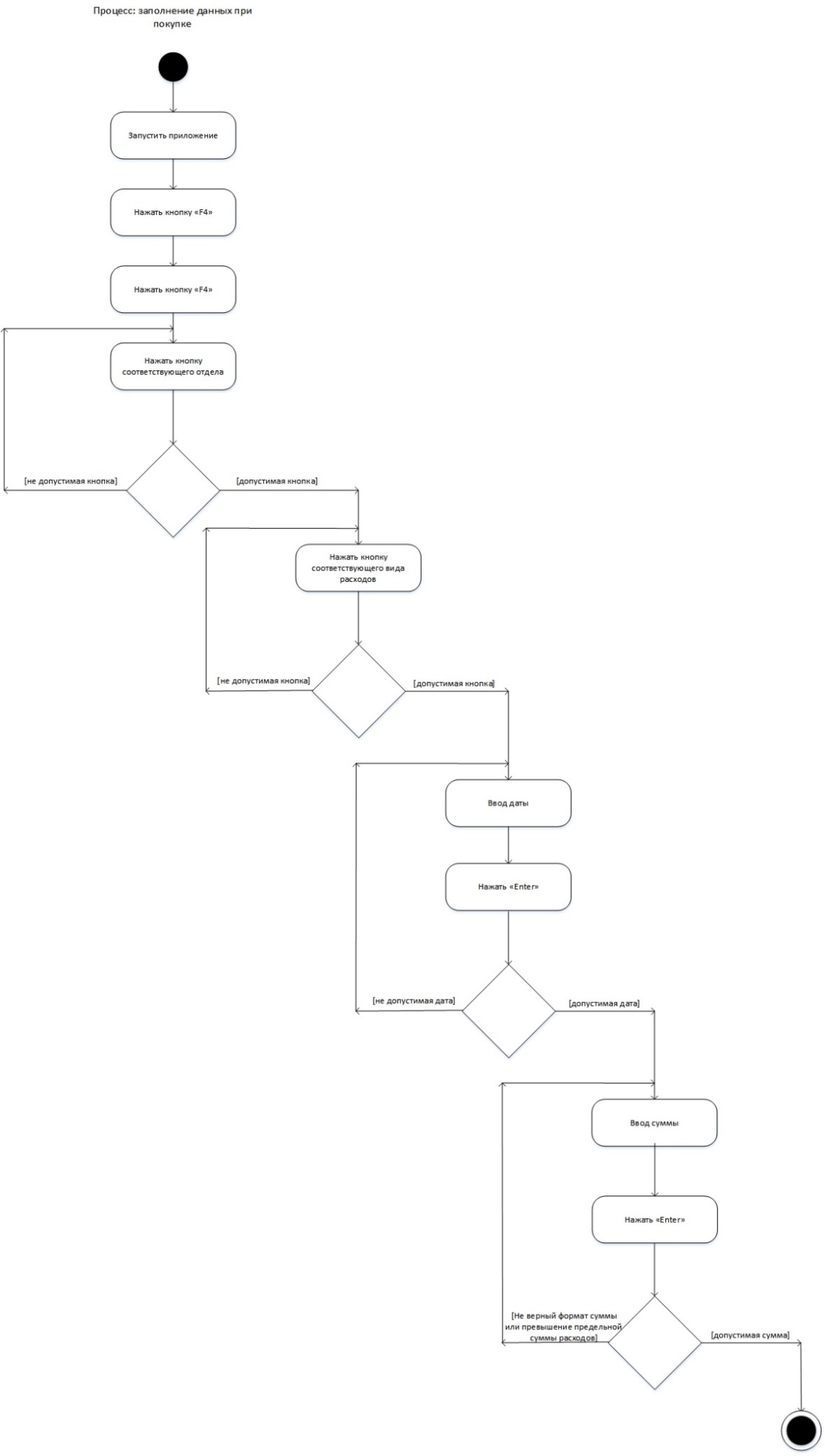
****

Рисунок 5 – Диаграмма деятельности

**2.3 Описание тестов**

На финальной стадии разработки проекта были созданы тесты, предусматривающие различные варианты действий. Тесты составлены таким образом, чтобы предусмотреть максимальное количество возможных действий.

Таблица 3 – Тесты на использование.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название теста | Действия | Исходная информация | Ожидаемая информация |
| T1 | Закрытие приложения по кнопке | Нажать «F1», когда приложение запрашивает «Ввод действия» | - | Закрытие приложения |
| T2 | Очистка консоли | Заполнить консоль информацией, нажать кнопку «F2», когда приложение запрашивает «Ввод действия» | - | Очистка консоли |
| T3 | Просмотр информации о кнопках | Нажать «F3», когда приложение запрашивает «Ввод действия» | - | Вывод информации о кнопках |
| T4 | Оформление отчета о покупке | Нажать «F4», когда приложение запрашивает «Ввод действия», пройти по всем шагам оформления покупки | Выбранные и введенные данные | Запись информации о покупке в текстовый файл в соответствии с видом расходов |
| T5 | Нажатие не допустимой кнопки | Нажать любую недопустимую кнопку | - | Вывод сообщения о нажатии недопустимой кнопки |
| T6 | Проверка неправильного ввода даты покупки | Нажать кнопку Нажать «F4», когда приложение запрашивает «Ввод действия», пройти по шагам до ввода даты, ввести некорректную дату | Введенная дата | Вывод сообщения о неправильно введенной дате |
| T7 | Проверка ввода суммы покупки, которая не подходит под условия | Нажать кнопку Нажать «F4», когда приложение запрашивает «Ввод действия», пройти по шагам до ввода суммы, ввести сумму меньше 0 или большую, так чтобы превышала предельную сумму | Введенная сумма | Сообщения об некорректности суммы или сообщение о превышении суммы, с выводом числа на сколько именно сумма превышена |
| T8 | Просмотр информации о всех расходах | Нажать кнопку «F5», когда приложение запрашивает «Ввод действия» | - | Вывод информации о всех расходах и о расходах по их виду |

**3 Построение программы**

Для построения данного приложения было использованы классы, представленные в таблице 1:

Таблица 1.

|  |  |  |
| --- | --- | --- |
| **Класс** | **Атрибуты** | **Назначение** |
| Departaments | NameDepartament;(Название отдела)  KolvoSotrudnikov;(Количество сотрудников) | Используется для определения отдела, из которого сотрудник совершает покупку |
| TypesOfExpenses | NameExpenses;(Название расходов)  Description;(Описание)  PredelNorma;(Предельная норма) | Используется для определения вида расходов, при покупке сотрудником |
| Expenses | TypeExp;(Вид расходов)  Depart;(Отдел)  Summa;(Сумма)  Date;(Дата) | Используется для оформления покупки, совершаемой сотрудником |
| Program |  | Используется для осуществления управления приложением |

Каждый класс имеет свои методы, помогающие работе приложения. Методы классов и их назначение представлены в таблице 2.

Таблица 2.

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Назначение** |
| Departaments | NDep | Определение отдела по нажатию соответствующей кнопки. |
| TypesOfExpenses | TOExp | Определение вида расходов по нажатию соответствующей кнопки. |
| Expenses | Expens | Определение даты и суммы (затраченной сотрудником), с проверкой на корректность и доступность. Запись потраченной суммы на соответствующую дату в текстовый файл. |
| Program | Info | Статический метод для выведения информации о доступных действиях и кнопках, активирующих эти действия. |
| Buy | Статический метод, который хранит объекты всех классов, которые участвуют в оформлении покупки. Это объекты классов Departaments, TypesOfExpenses, Expenses |
| Show | Статический метод для вывода информации о расходах по каждому виду и общей суммы всех расходов. |
| Main | Точка входа приложения. С выбором действий над данным приложением. |

Диаграмма классов представлена на ниже (Рисунок 6).

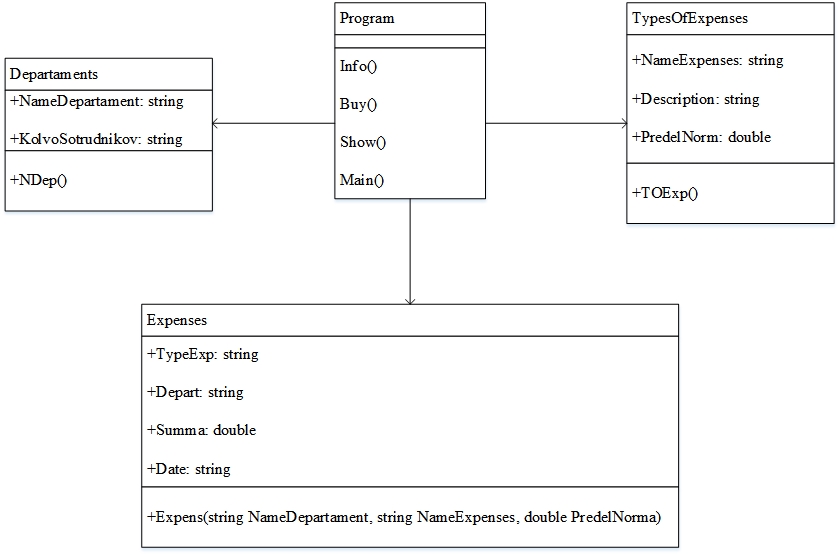


Рисунок 6 – Диаграмма классов

**4 Тестирование**

Подробности о тестах можно посмотреть в таблице 3.

Таблица 4 – Отчет о результатах тестирования

|  |  |
| --- | --- |
| № | Статус |
| T1 | Выполнено успешно |
| T2 | Выполнено успешно |
| T3 | Выполнено успешно |
| T4 | Выполнено успешно |
| T5 | Выполнено успешно |
| T6 | Выполнено успешно |
| T7 | Выполнено успешно |
| T8 | Выполнено успешно |

**5 Применение**

**5.1 Назначение и условия применения программы**

Консольное приложение «Учет внутриофисных расходов» предназначено для отслеживания покупок сотрудниками чего-либо за счет компании и оформления этих покупок.

Функции приложения:  
 − Закрытие приложения по нажатию одной кнопки;

− Очистка консоли приложения по нажатию одной кнопки;

− Вывод информации о кнопках в приложении по нажатию одной кнопки;

− Оформление покупок;

− Вывод всех расходов по их видам и общая сумма нажатию одной кнопки;

Для работы с приложением понадобится система со следующими параметрами:

− процессор ARM64 или x64; Рекомендуется четырехядерный или более качественный;

− от 4 GB RAM;

− монитор с минимальным разрешением 1366х768.

**5.2 Инсталляция**

Для установки приложения необходимо запустить установщик под названием UchetRas.exe (Рис. 6).



Рисунок 6 – Установщик приложения

**5.3 Выполнение приложения**

Для запуска приложения необходимо пройти по пути Учет внутриофисных расходов\bin\Debug\net6.0\UchetRas.exe. Откроется главное окно (Рис. 7)

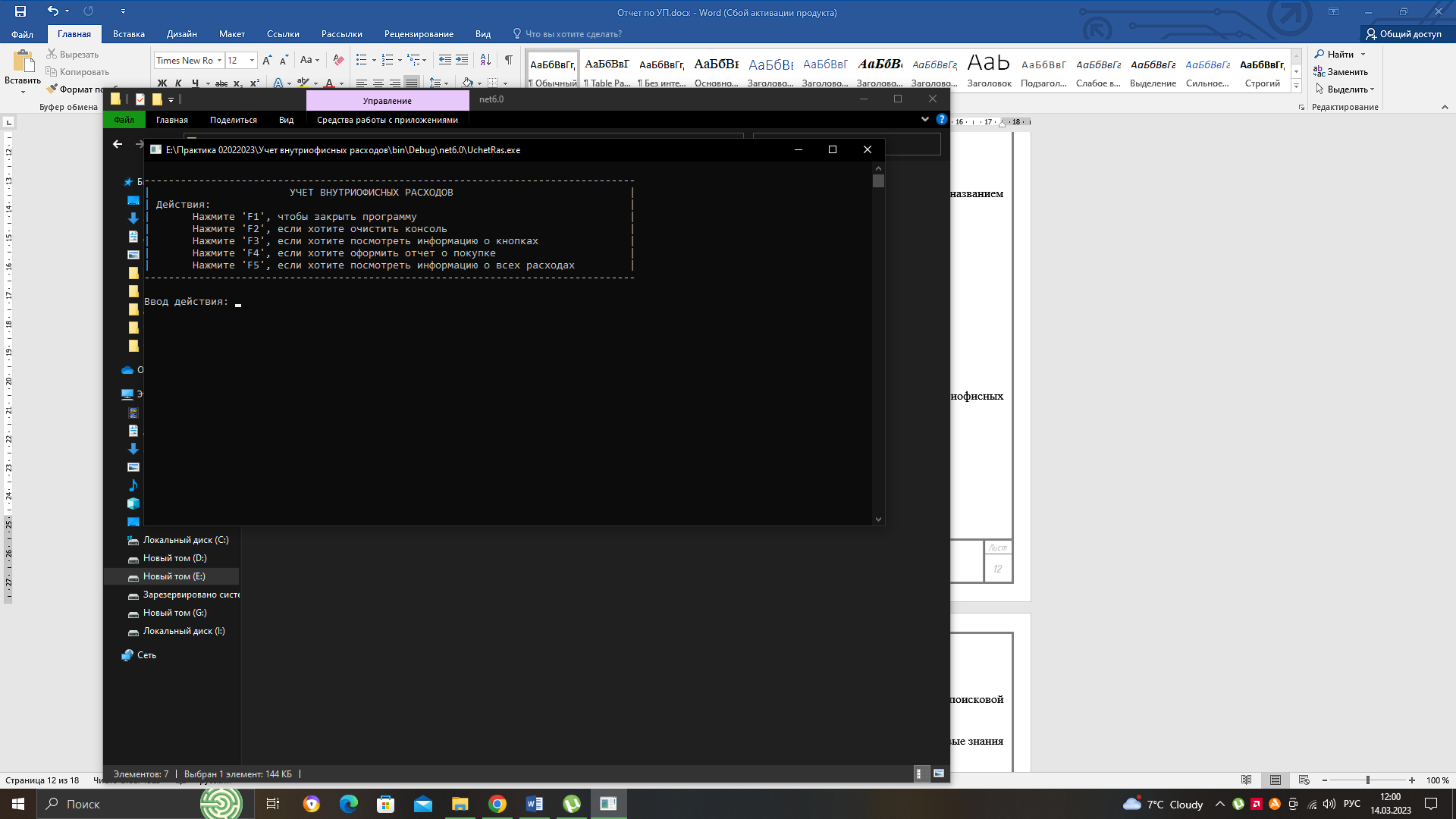


Рисунок 7 – Главное окно приложения

Внутри приложения есть вся информация о его функциях (Рис. 8), в основном все функции выполняются по нажатию горячих клавиш.

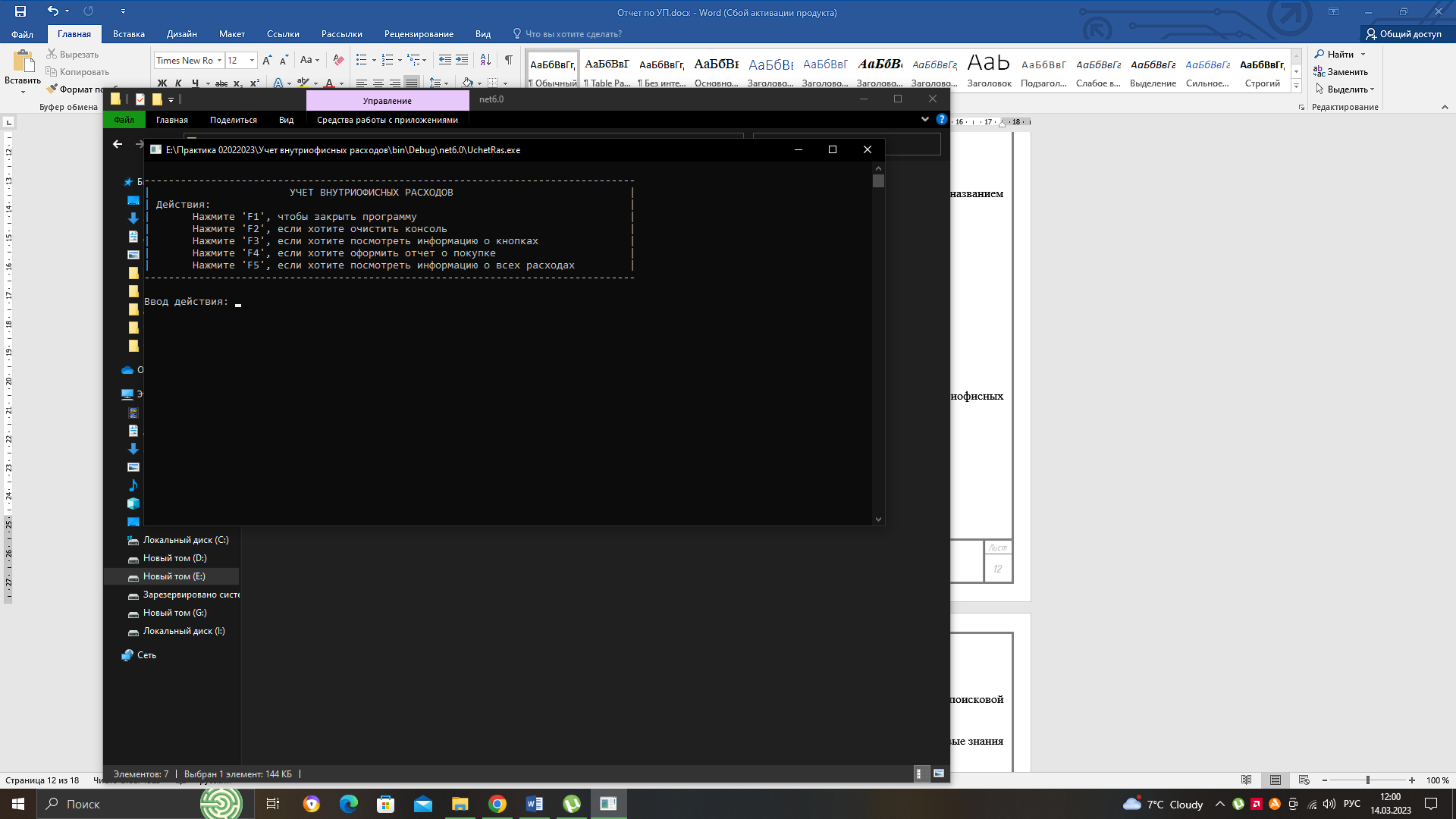


Рисунок 8 - вся информация о его функциях приложения

**Заключение**

Целью данного учебного проекта являлась разработка консольного приложения по предметной области «Учет внутриофисных расходов».

В ходе реализации поставленной задачи были укреплены старые и усвоены новые знания по использованию Microsoft Visual Studio 2022, а также курс дисциплины «Конструирование программ и языки программирования».

Хотелось бы выделить библиотеку классов System.Globalization, которая помогла внедрить в приложение проверку на корректность даты, при оформлении покупки.

После тщательного тестирования системы не было выявлено некоторые недоработки, которые были частично исправлены на стадии проектирования, и полностью исключены на стадии тестирования программы. В целом при реализации программы, были выполнены все условия, перечисленные в описании предметной области. Таким образом, можно сказать, что система была реализована.

**Список использованных источников**

1. Поддержка создателя среды разработки MS Visual Studio 2022 при работе с ней:

https://support.microsoft.com/

2. Помощь при работе с языком программирования C#:

https://learn.microsoft.com/

https://www.cyberforum.ru/

<https://www.youtube.com/>

https://stackoverflow.com/

**Приложение А**

using System;

using System.Data;

using System.Diagnostics;

using System.Reflection.Metadata;

using System.Runtime.CompilerServices;

using System.Security.AccessControl;

using System.Security.Cryptography;

using System.Threading.Channels;

using static System.Collections.Specialized.BitVector32;

using static System.Net.Mime.MediaTypeNames;

using System.Text.RegularExpressions;

using System.Globalization;

using System.Linq.Expressions;

using System.Reflection.Metadata.Ecma335;

namespace MyProgram

{

class Departaments//Класс отделов

{

public string NameDepartament;//Название отдела

public int KolvoSotrudnikov;//Количество сотрудников

public void NDep()//определение отдела

{

ConsoleKeyInfo depart;//переменная для выбора отдела

Console.WriteLine("Выберите ваш отдел:\n\tF1 - Коммерческий\n\tF2 - Рекламный\n\tF3 - Технический\n\tF4 - Финансовый\n\tF5 - Юридический");

Console.WriteLine("\nНажмите кнопку вашего отдела: ");

while (true)

{

depart = Console.ReadKey(true);

if ((depart.Key == ConsoleKey.F1) || (depart.Key == ConsoleKey.F2) || (depart.Key == ConsoleKey.F3) || (depart.Key == ConsoleKey.F4) || (depart.Key == ConsoleKey.F5))

{

if (depart.Key == ConsoleKey.F1)

{

NameDepartament = "Коммерческий";

KolvoSotrudnikov = 25;

break;

}

if (depart.Key == ConsoleKey.F2)

{

NameDepartament = "Рекламный";

KolvoSotrudnikov = 16;

break;

}

if (depart.Key == ConsoleKey.F3)

{

NameDepartament = "Технический";

KolvoSotrudnikov = 18;

break;

}

if (depart.Key == ConsoleKey.F4)

{

NameDepartament = "Финансовый";

KolvoSotrudnikov = 30;

break;

}

if (depart.Key == ConsoleKey.F5)

{

NameDepartament = "Юридический";

KolvoSotrudnikov = 27;

break;

}

}

else

{

Console.WriteLine("Неверная кнопка, посмотрите информацию о доступных кнопках!\n");

}

}

Console.WriteLine("Отдел - " + NameDepartament + ", Количество сотрудников - " + KolvoSotrudnikov);

}

}

class TypesOfExpenses//Класс видов расходов

{

public string NameExpenses;//Название расходов

public string Description;//Описание

public double PredelNorma;//Предельная норма

public void TOExp()//определение вида расходов

{

ConsoleKeyInfo typeexp;//переменная для выбора вида расходов

Console.WriteLine("\nВыберите вид расходов:\n\tF1 - Канцелярия\n\tF2 - Ремонт офисной техники\n\tF3 - Сеть\n\tF4 - Мебель\n\tF5 - Удобства сотрудников");

Console.WriteLine("\nНажмите кнопку вида расходов: ");

while (true)

{

typeexp = Console.ReadKey(true);

if ((typeexp.Key == ConsoleKey.F1) || (typeexp.Key == ConsoleKey.F2) || (typeexp.Key == ConsoleKey.F3) || (typeexp.Key == ConsoleKey.F4) || (typeexp.Key == ConsoleKey.F5))

{

if (typeexp.Key == ConsoleKey.F1)

{

NameExpenses = "Канцелярия";

Description = "Расходы на канцтовары(бумага, ручки, карандаши, клей, степлеры, дыроколы, и т.д.)";

PredelNorma = 500;

break;

}

if (typeexp.Key == ConsoleKey.F2)

{

NameExpenses = "Ремонт офисной техники";

Description = "\tРасходы на починку офисной техники(компьютеры, принтеры, ксероксы, телефоны, пылесосы и т.д.)";

PredelNorma = 800;

break;

}

if (typeexp.Key == ConsoleKey.F3)

{

NameExpenses = "Сеть";

Description = "Расходы на сеть(покупка новых тарифов, оплата хостинга, интернета, антивируса и т.д.)";

PredelNorma = 700;

break;

}

if (typeexp.Key == ConsoleKey.F4)

{

NameExpenses = "Мебель";

Description = "Расходы на мебель(покупка новой мебели, ремонт мебели)";

PredelNorma = 1000;

break;

}

if (typeexp.Key == ConsoleKey.F5)

{

NameExpenses = "Удобства сотрудников";

Description = "Расходы на удобства сотрудников(покупка кондиционеров, кулеров, бутилированной воды, кофемашин, электрочайников и т.д.)";

PredelNorma = 550;

break;

}

}

else

{

Console.WriteLine("Неверная кнопка, посмотрите информацию о доступных кнопках!\n");

}

}

Console.WriteLine("\tВид расходов - " + NameExpenses + "\n\tОписание: " + Description + "\n\tПредельная норма расходов - " + PredelNorma);

}

}

class Expenses//Класс расходов

{

public string TypeExp;//Вид расходов

public string Depart;//Отдел

public double Summa;//Сумма

public string Date;//Дата

public void Expens(string NameDepartament, string NameExpenses, double PredelNorma)

{

TypeExp = NameExpenses;

Depart = NameDepartament;

DateTime adate;//переменная для проверки правильности даты

CultureInfo provider = CultureInfo.InvariantCulture;//проверка даты в соответствии с регионом

string format = "dd.MM.yyyy";//формат даты

//Ввод и проверка даты

while (true)

{

Console.Write("\nВведите дату в формате 'DD.MM.YYYY': ");

Date = Console.ReadLine();

try

{

adate = DateTime.ParseExact(Date, format, provider);//Попытка привести введенную дату к правильному формату даты

break;

}

catch

{

Console.WriteLine("Дата введена некорректно! Попробуйте еще раз.");

}

}

StreamReader filedate = new StreamReader("data\\Отделы\\"+NameExpenses +".txt");//поток чтения для определения наличия месяца в записи данных

string checkdate;//переменная даты в текстовом файле

int k = 0;//переменная для определения наличия даты в текстовом файле данных

while (!filedate.EndOfStream)

{

checkdate = (filedate.ReadLine()).Substring(0, 7);//берем номера символов, которые хранят дату

if (Date.Substring(3) == checkdate) k = 1;

}//определение есть ли данный месяц и год в текстовом файле данных

filedate.Close();

//добавление в текстовый файл данных даты если она отсутствует

if (k == 0)

{

string y1 = "\n" + Date.Substring(3) + " мес:0";

File.AppendAllText("data\\Отделы\\" + NameExpenses + ".txt", y1);

string m1 = "data\\Отделы\\Канцелярия.txt";

string m2 = "data\\Отделы\\Мебель.txt";

string m3 = "data\\Отделы\\Ремонт офисной техники.txt";

string m4 = "data\\Отделы\\Сеть.txt";

string m5 = "data\\Отделы\\Удобства сотрудников.txt";

string[] massiv = { m1, m2, m3, m4, m5 };

for (int i = 0; i < 5; i++)//цикл чтения суммы из 5 разных файлов

{

File.AppendAllText(massiv[i], y1);

}

Console.WriteLine("Добавлена новая дата");

}

//Ввод и проверка суммы расходов

string hj;//переменная хранения прошлой суммы

while (true)

{

try

{

Console.Write("\nВведите сумму расходов: ");

Summa = double.Parse(Console.ReadLine());

if ((Summa + double.Parse(File.ReadAllLines("data\\Отделы\\" + NameExpenses + ".txt").Last().Substring(13))) > PredelNorma)

{

Console.WriteLine("\nСумма расходов превышает предел на {0}! Введите другое значение.", (Summa + double.Parse((File.ReadAllLines("data\\Отделы\\" + NameExpenses + ".txt").Last()).Substring(13))) - PredelNorma);

}

else

{

if (Summa < 0) Console.WriteLine("Неккоректный ввод суммы!");

else

{

hj = File.ReadAllLines("data\\Отделы\\" + NameExpenses + ".txt").Last().Substring(13);

hj = File.ReadAllLines("data\\Отделы\\" + NameExpenses + ".txt").Last().Substring(0, 13) + Convert.ToString(double.Parse(hj) + Summa);

var lines = File.ReadAllLines("data\\Отделы\\" + NameExpenses + ".txt");

File.WriteAllLines("data\\Отделы\\" + NameExpenses + ".txt", lines.Take(lines.Length - 1).ToArray());//перезапись того же файла, но без последней строки

File.AppendAllText("data\\Отделы\\" + NameExpenses + ".txt", hj);

break;

}

}

}

catch

{

continue;

}

}

Console.WriteLine("\n\tПокупка оформлена!");

}

}

class Program

{

static void Info()

{

Console.WriteLine();

Console.WriteLine("---------------------------------------------------------------------------------");

Console.WriteLine("|\t\t\tУЧЕТ ВНУТРИОФИСНЫХ РАСХОДОВ\t\t\t\t|");

Console.WriteLine("| Действия:\t\t\t\t\t\t\t\t\t|");

Console.WriteLine("|\tНажмите 'F1', чтобы закрыть программу\t\t\t\t\t|");

Console.WriteLine("|\tНажмите 'F2', если хотите очистить консоль\t\t\t\t|");

Console.WriteLine("|\tНажмите 'F3', если хотите посмотреть информацию о кнопках\t\t|");

Console.WriteLine("|\tНажмите 'F4', если хотите оформить отчет о покупке\t\t\t|");

Console.WriteLine("|\tНажмите 'F5', если хотите посмотреть информацию о всех расходах\t\t|");

Console.WriteLine("---------------------------------------------------------------------------------");

}

static void Buy()//метод для покупки

{

Departaments dep = new Departaments();

TypesOfExpenses toexpen = new TypesOfExpenses();

Expenses expen = new Expenses();

Console.WriteLine("купить");

dep.NDep();

toexpen.TOExp();

expen.Expens(dep.NameDepartament, toexpen.NameExpenses, toexpen.PredelNorma);

}

static void Show()//Метод для вывода расходов

{

double money = 0;

string ras;

string m1 = "data\\Отделы\\Канцелярия.txt";

string m2 = "data\\Отделы\\Мебель.txt";

string m3 = "data\\Отделы\\Ремонт офисной техники.txt";

string m4 = "data\\Отделы\\Сеть.txt";

string m5 = "data\\Отделы\\Удобства сотрудников.txt";

string[] mass = {m1, m2, m3, m4, m5};//массив путей хранения файлов содержащих суммы затрат

ras = File.ReadAllLines(mass[0]).Last();

char[] Mychar = { '.', 't', 'x', 't' };//Символы которые будут удалены

Console.WriteLine("\nСумма расходов на {0}:", ras.Substring(0, 7));

for (int i = 0; i < 5; i++)//цикл чтения суммы из 5 разных файлов

{

ras = File.ReadAllLines(mass[i]).Last();//считывание последней строки из файлов

Console.WriteLine("\tРасходы на {0}: {1}", (mass[i].Substring(12)).TrimEnd(Mychar), ras.Substring(13));//Вывод суммы расходов на конкретные виды расходов

money += double.Parse(ras.Substring(13));//прибавление суммы расходов из текстового файла на начиная с 7 символа(откуда начинается сумма)

}

Console.WriteLine("\nОбщие внутриофисные расходы составляют: " + money);

}

static void Main()

{

ConsoleKeyInfo action;//Действие на нажатие клавиши

Info();

while (true)

{

Console.Write("\nВвод действия: ");

action = Console.ReadKey(true);

if ((action.Key == ConsoleKey.F1) || (action.Key == ConsoleKey.F2) || (action.Key == ConsoleKey.F3) || (action.Key == ConsoleKey.F4) || (action.Key == ConsoleKey.F5))

{

if (action.Key == ConsoleKey.F1) break;

if (action.Key == ConsoleKey.F2)

{

Console.Clear();

Info();

Console.WriteLine("\nКонсоль очищена!");

}

if (action.Key == ConsoleKey.F3) Info();

if (action.Key == ConsoleKey.F4) Buy();

if (action.Key == ConsoleKey.F5) Show();

}

else

{

Console.WriteLine("Неверная кнопка, посмотрите информацию о доступных кнопках!");

}

}

}

}

}