**КАЗАНСКИЙ(ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ «ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И**

**ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

**СПЕЦИАЛЬНОСТЬ «БИЗНЕС-ИНФОРМАТИКА»**

**КАФЕДРА АНАЛИЗА ДАННЫХ И ИССЛЕДОВАНИЯ ОПЕРАЦИЙ**

**ОТЧЕТ**

по предмету

Система поддержки принятия решений

Тема: «Эффективное распределение бюджетных средств на рекламу для ювелирного магазина»

Выполнила:

Преподаватель:

Шустова Евгения Петровна

Казань 2022

# Введение

Ювелирный магазин занимается продажей ювелирных изделий в розницу, как в онлайн-магазине, так и в оффлайн-магазине г.Казань. Для повышения узнаваемости и привлечения новых потенциальных клиентов, ювелирный магазин закупает рекламу на различных площадках (Telegram каналы, VK сообщества, баннеры в общественных местах, флаеры около торговых точек и тд).

СППР будет помогать ЛПР – управляющему ювелирным магазином – принимать релевантное решение в распределении бюджетных средств, выделенных директором на маркетинговую деятельность, для получения наибольшей выгоды от покупки различной рекламы. Такое решение будет приниматься на основе проведения анализа на оптимальное распределение денежных средств из ограниченного бюджета, реализованного в данной работе.

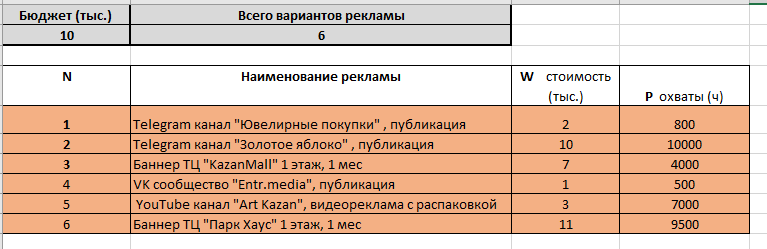
Данная программа позволяет оптимально распределить ограниченные бюджетные средства так, чтобы получить наибольшую выгоду от покупки рекламы (получить наибольшие охваты). Для этого пользователю необходимо указать выделенный на маркетинговую деятельность бюджет и заполнить таблицу всех вариантов рекламы с наименованием, стоимостью покупки и средними охватами рекламы. Далее следует нажать на кнопку «Рассчитать». Сформированная рядом таблица будет содержать такой набор рекламы, покупка которых принесет наибольшие охваты и привлечет новых клиентов в ювелирный магазин при минимальных затратах бюджетных средств.

Разработка СППР будет реализована на языке С#.

# База данных

В качестве базы данных используется файл расширением .xlsx. В базе данных определен набор рекламы, с соответствующими для них параметрами: стоимость и охваты, а также весь выделенный бюджет.

Структура файла следующая:



В ячейке “A2” задается общий бюджет, далее заполняется таблица с данными о стоимости покупки (в столбцах С[i] для i от 1 до N ) рекламы и об охватах (в столбцах D[i] для i от 1 до N ), которые гарантирует такая реклама.

# Модель базы знаний

Задачу оптимального распределения денежных средств из ограниченного бюджета можно рассматривать как задачу о ранце, которая звучит следующим образом:

Пусть имеется набор предметов, каждый из которых имеет два параметра - вес и ценность. И есть рюкзак, определенной вместимости. Задача заключается в том, чтобы собрать рюкзак с максимальной ценностью предметов внутри, соблюдая при этом весовое ограничение рюкзака.

Для выбранной мной предметной области воспринимать задачу буквально нет смысла, поэтому условия задачи я сформулировала следующим образом:

Пусть имеется некоторый известный список всевозможных вариантов рекламы, каждая из которых имеет два параметра – **стоимость покупки** и **охваты** (зависит от среднего количества просмотров). И есть **ограниченный бюджет**, определенный директором. Задача заключается в том, чтобы управляющему выделить такой набор рекламы, покупка которой принесет **максимально возможные охваты**, соблюдая при этом денежное ограничение по бюджету.

Для решения поставленной задачи существует несколько эффективных алгоритмов. В данной программе **используется алгоритм динамического программирования**.

В основе метода динамического программирования лежит следующий принцип:

Оптимальное решение на i шаге находится исходя из найденных ранее оптимальных решений на предшествующих шагах. Из этого следует, что для того чтобы найти оптимальное решение на последнем шаге надо сначала найти оптимальное решения для первого, затем для второго и так далее пока не пройдем все шаги до последнего.

**Постановка задачи о ранце:**

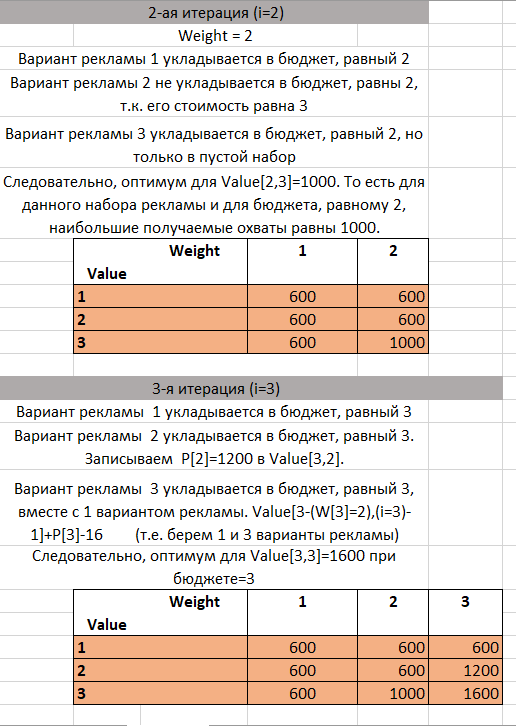
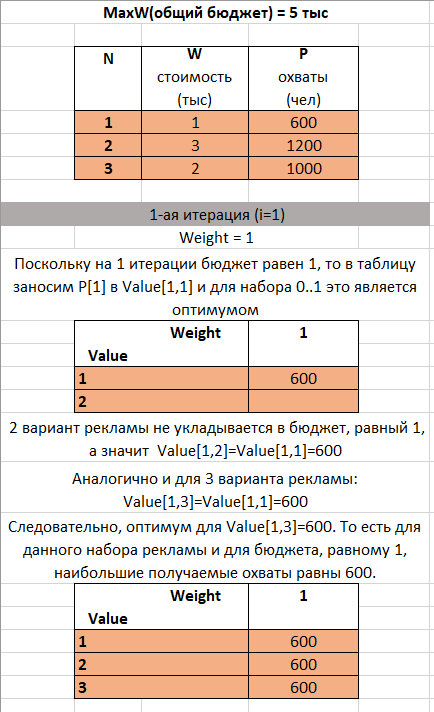
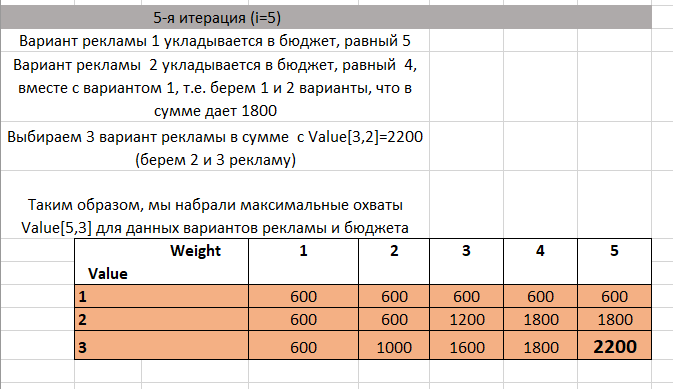
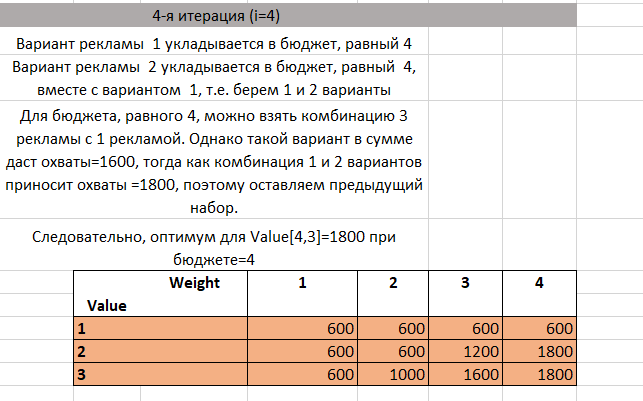
Имеется набор из N предметов. Пусть MaxW - объем рюкзака, Pi – стоимость i-го предмета, Wi – вес i-го предмета. Value[W, i] – максимальная сумма, которую надо найти. Суть метода динамического программирования – на каждом шаге по весу 1<Wi<W находим максимальную загрузку Value[Wi, i], для веса Wi. Допустим мы уже нашли Value[1..W, 1..i-1], то есть для веса меньше либо равного W и с предметами, взятыми из 1..N-1.

Рассмотрим предмет N, если его вес WN меньше W проверим стоит ли его брать.

Если его взять, то вес станет W-Wi , тогда Value[W, i] = Value[W – Wi , i-1] + Pi (для Value[W – Wi , i-1]) решение уже найдено остается только прибавить Pi.

Если его не брать, то вес останется тем же и Value[W , i] = Value[W – Wi, i-1]. Из двух вариантов выбирается тот, который дает наибольший результат.

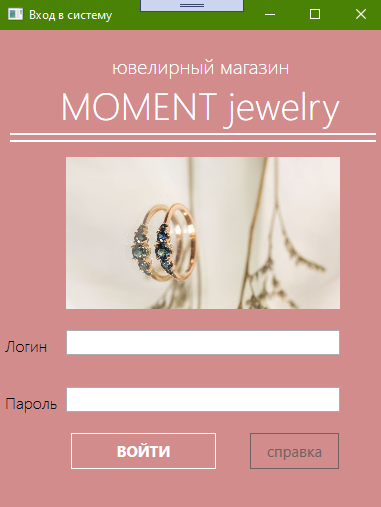
Рассмотрим алгоритм подробнее с подходящей формулировкой:

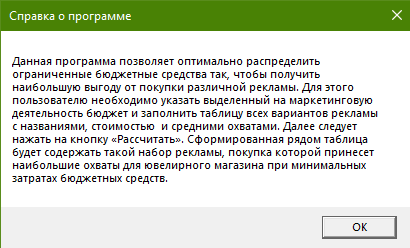
Алгоритм динамического программирования задачи о рюкзаке дает точное решение путем использования дополнительной памяти O(N\*MaxW), временная сложность алгоритма так же будет порядка O(N\*MaxW).

# Интерфейс

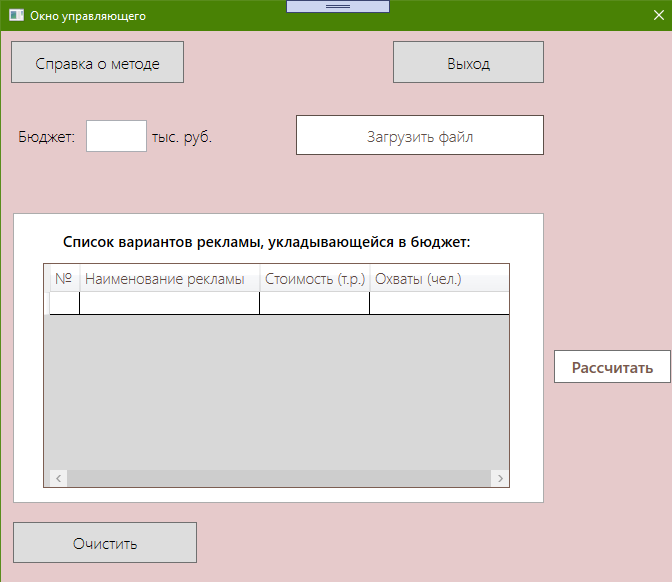
При запуске программы открывается окно «Вход в систему». Пользователю необходимо ввести логин и пароль, после чего нажать на кнопку «ВОЙТИ».



В этом же окне пользователь может ознакомиться с информацией о программе - для этого предназначена кнопка «справка».



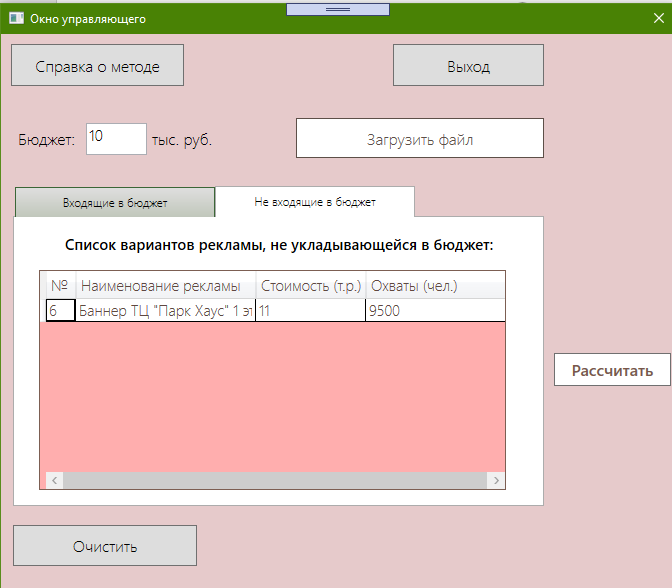
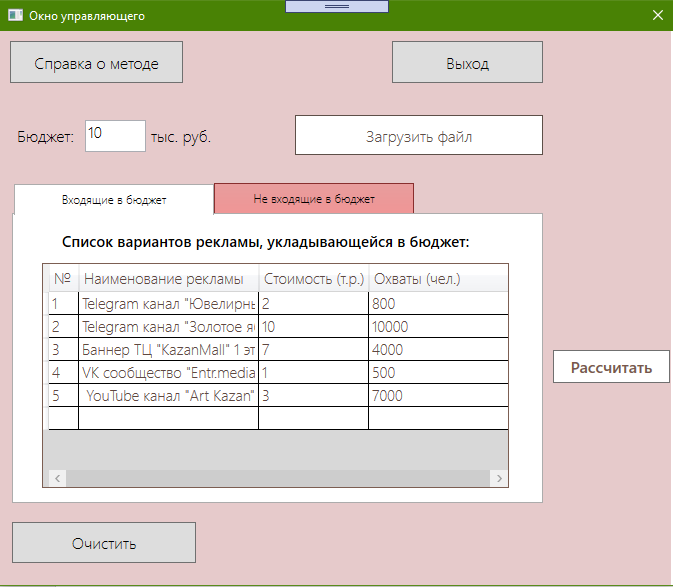
После того, как пользователь вошел в систему, открывается окно «Окно управляющего».



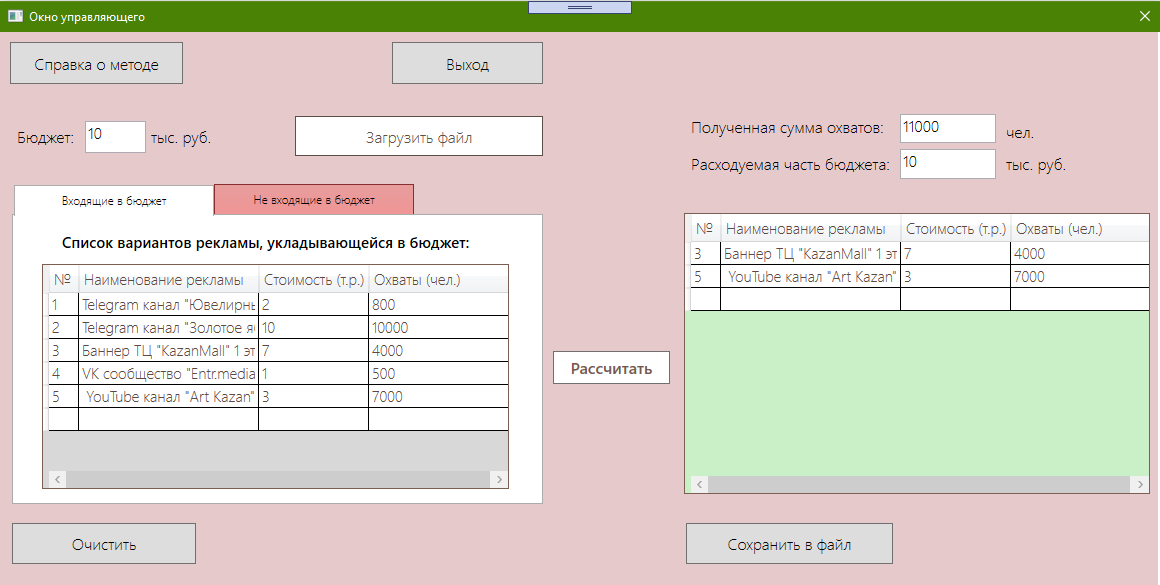
В этом окне пользователь может нажать на кнопку «Справка о методе», чтобы узнать, как именно реализован алгоритм.

Далее пользователь может самостоятельно заполнить поле бюджета и список всех вариантов рекламы с соответствующими данными прямо в таблице, либо загрузить имеющиеся данные из файла, кликнув на кнопку «Загрузить файл».

Если данные загружаются из файла, то при наличии варианта рекламы, стоимость которой больше заданного бюджета, эта реклама попадает на вкладку «Не входящие в бюджет».



После этого пользователь нажимает на кнопку «Рассчитать» и посчитанные алгоритмом данные (суммарные охваты, расходуемая часть бюджета и оптимальный набор рекламы) отображаются в том же окне. Найденные значения доступны только для чтения, их пользователь изменить не может.



Пользователь может нажать на кнопку «Очистить», чтобы удалить все заполненные данные.

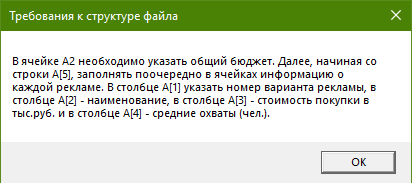
Чтобы вернуться в окно входа в систему, пользовать может нажать на кнопку «Выход».

Если все данные будут заполнены, отобразится кнопка «Сохранить в файл», позволяющая пользователю сохранить исходные и полученные данные в отчет в Excel.

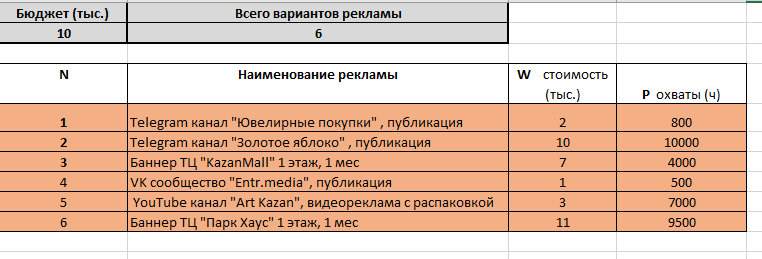
# Система управления базой данных

Данная программа предусматривает как самостоятельный ввод данных пользователем в окне программы, так и взаимодействие с базой данных. Для этого используется кнопка «Загрузить файл».

При нажатии на кнопку «Загрузить файл» пользователь будет уведомлен о структуре файла.



В качестве базы данных используется excel-файл, который выглядит следующим образом:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_начало кода\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Загрузка файла осуществляется методом dowload\_Button\_Click :

private void dowload\_Button\_Click(object sender, RoutedEventArgs e)

{

var reklama\_list = fromExcel().ToList();

var ok\_reklama\_list = reklama\_list.Where(x => x.w <= Event.maxx\_PP).ToList();

var non\_reklama\_list = reklama\_list.Where(x => x.w > Event.maxx\_PP).ToList();

reklama\_listDataGrid.ItemsSource = ok\_reklama\_list;

NONreklama\_listDataGrid.ItemsSource = non\_reklama\_list;

max.Text = Event.maxx\_PP.ToString();

tab1.Visibility = Visibility.Visible;

if (non\_reklama\_list.Count!=0)

tab2.Visibility = Visibility.Visible;

}

При клике на кнопку, в первую очередь, вызывается метод fromExcel(), выходным параметром которого является коллекция рекламы (объекты класса Reklama), которая реализуются интерфейсом IEnumerable<T>.

Метод выглядит следующим образом:

public static IEnumerable<Reklama> fromExcel()

{



MessageBox.Show("В ячейке A2 необходимо указать общий бюджет. Далее, начиная со строки А[5], заполнять поочередно в ячейках информацию о каждой рекламе. В столбце А[1] указать номер варианта рекламы, в столбце А[2] - наименование, в столбце А[3] - стоимость покупки в тыс.руб. и в столбце А[4] - средние охваты (чел.)."

, "Требования к структуре файла");

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "xcel Files(.xls)|\*.xls| xcel Files(.xlsx)| \*.xlsx | Excel Files(\*.xlsm) | \*.xlsm";

Nullable<bool> result = openFileDialog1.ShowDialog();

Microsoft.Office.Interop.Excel.Application ExcelApp;

Workbook excelappworkbook;

Microsoft.Office.Interop.Excel.Worksheet worksheet;

Range excelcells, excelcells1, excelcells2, excelcells3, excelcells4;

if (result == true)

{

string fileName = System.IO.Path.GetFileName(openFileDialog1.FileName);

ExcelApp = new Microsoft.Office.Interop.Excel.Application();

excelappworkbook = ExcelApp.Workbooks.Open(openFileDialog1.FileName);

worksheet = (Microsoft.Office.Interop.Excel.Worksheet)excelappworkbook.Sheets[1];

excelcells = worksheet.Cells[2, 1];

Reklama.maxx\_PP = Convert.ToInt32(excelcells.Value);

excelcells = worksheet.Cells[2, 2];

Reklama.maxx\_nn = Convert.ToInt32(excelcells.Value);

{

// Перебираем диапазон нужных строк

for (int row = 5; row < 5 + Reklama.maxx\_nn; ++row)

{

excelcells1 = worksheet.Cells[row, 1];

excelcells2 = worksheet.Cells[row, 2];

excelcells3 = worksheet.Cells[row, 3];

excelcells4 = worksheet.Cells[row, 4];

// По каждой строке формируем объект

Reklama reklama\_i = new Reklama

{

num = Convert.ToInt32(excelcells1.Value),

name = Convert.ToString(excelcells2.Value),

w = Convert.ToInt32(excelcells3.Value),

p = Convert.ToInt32(excelcells4.Value),

};

// И возвращаем его

yield return reklama\_i;

}

}

ExcelApp.Quit();

}

}

Именно этот метод считывает всю информацию из файла и возвращает её уже в виде некоторой коллекции данных.

конец кода

После загрузки файла данные отобразятся в окне программы. При этом, автоматически осуществится фильтрация загруженных из файла данных. Если в файле окажутся такие варианты рекламы, стоимость которых превышает заявленный бюджет магазина, то в окне программы появятся 2 вкладки: «Входящие в бюджет» и «Не входящие в бюджет».

начало кода

В коде фильтрация происходит сразу после получения общего списка рекламы. Создаются два списка:

1. Список ok\_reklama\_list включает рекламу, стоимость которой (поле w) меньше (или равна) общего бюджета (Reklama.maxx\_PP)

var ok\_reklama\_list = reklama\_list.Where(x => x.w <= Reklama.maxx\_PP).ToList();

1. Список non\_reklama\_list включает рекламу, стоимость которой (поле w) больше общего бюджета (Reklama.maxx\_PP)

var non\_reklama\_list = reklama\_list.Where(x => x.w > Reklama.maxx\_PP).ToList();

Затем каждый из созданных списков привязывается соответствующей таблице на форме и заполняется поле с общим бюджетом:

reklama\_listDataGrid.ItemsSource = ok\_reklama\_list;

NONreklama\_listDataGrid.ItemsSource = non\_reklama\_list;

max.Text = Reklama.maxx\_PP.ToString();

Далее меняется видимость вкладки с укладывающимися в бюджет вариантами рекламы и видимость с не укладывающимися в бюджет вариантами (но только в случае, если такие есть):

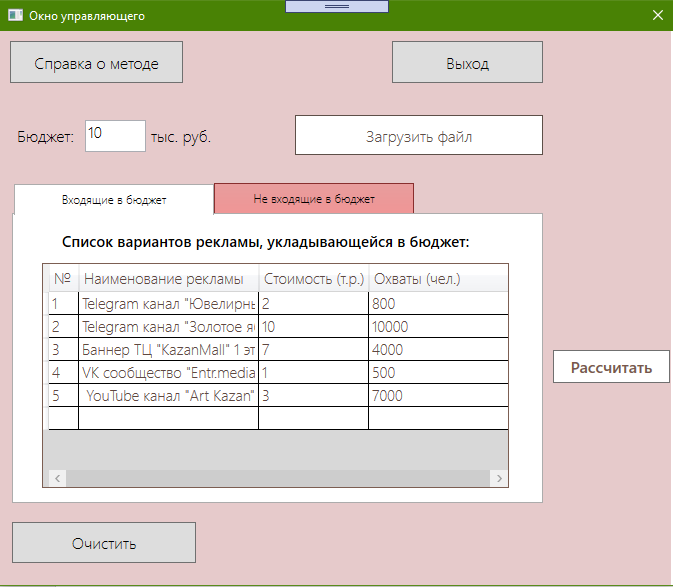
tab1.Visibility = Visibility.Visible;

if (non\_reklama\_list.Count!=0)

tab2.Visibility = Visibility.Visible;

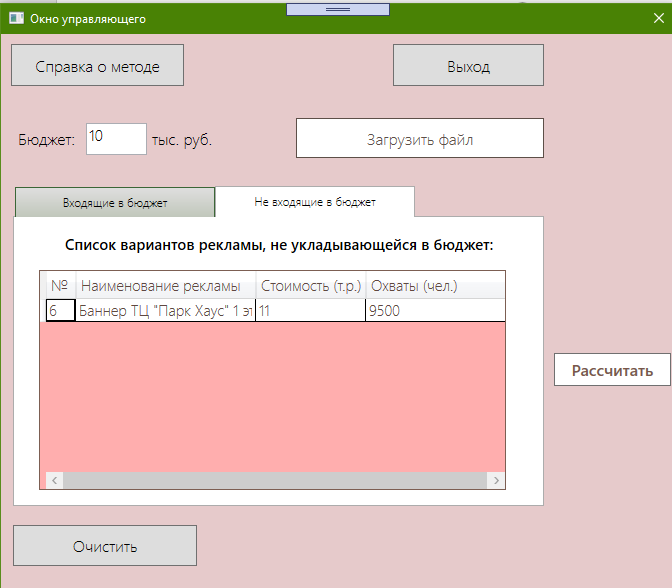
конец кода

Соответственно, на вкладке «Входящие в бюджет» отобразятся только те варианты рекламы, стоимость покупки которых будет меньше или равна общему бюджету ювелирного магазина:



Конечно, в дальнейшем алгоритм будет работать только с тем списком рекламы, которые очевидно укладываются в бюджет.

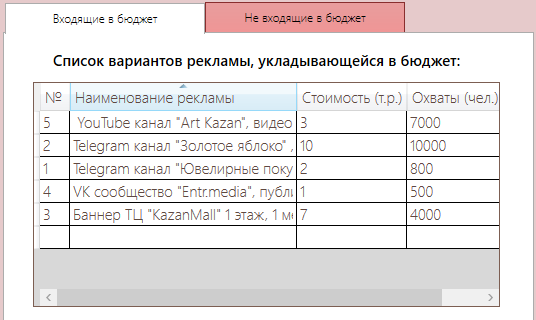
А на вкладке «Не входящие в бюджет» отобразятся только те варианты рекламы, стоимость покупки которых больше общего бюджета магазина:



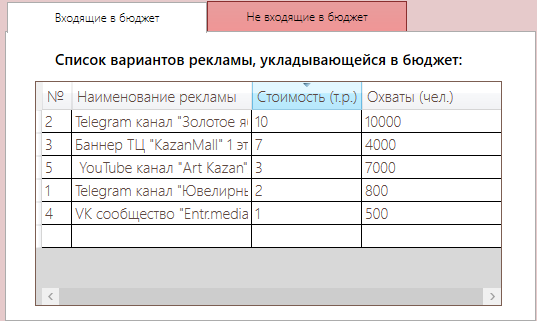
Такая фильтрация данных предоставит ЛПР наглядную информацию о рекламе, купить которую точно не получится из-за недостаточного количества выделенных бюджетных средств на маркетинг, в результате чего ЛПР может пересмотреть бюджет при необходимости.

При этом все данные можно отсортировать по любому параметру (по возрастанию или по убыванию), просто кликнув по заголовку столбца:

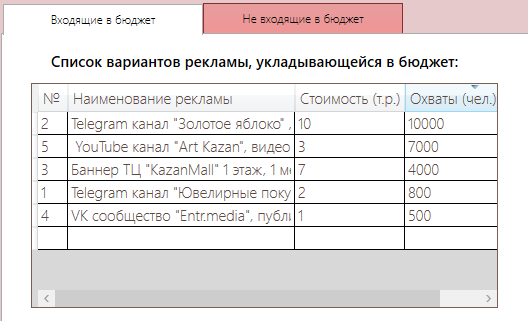
1.по названию



2. по стоимости

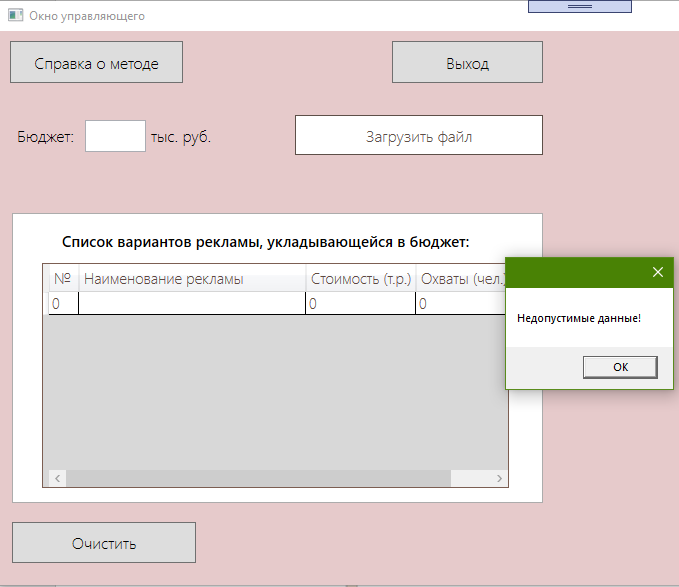


3. по охватам

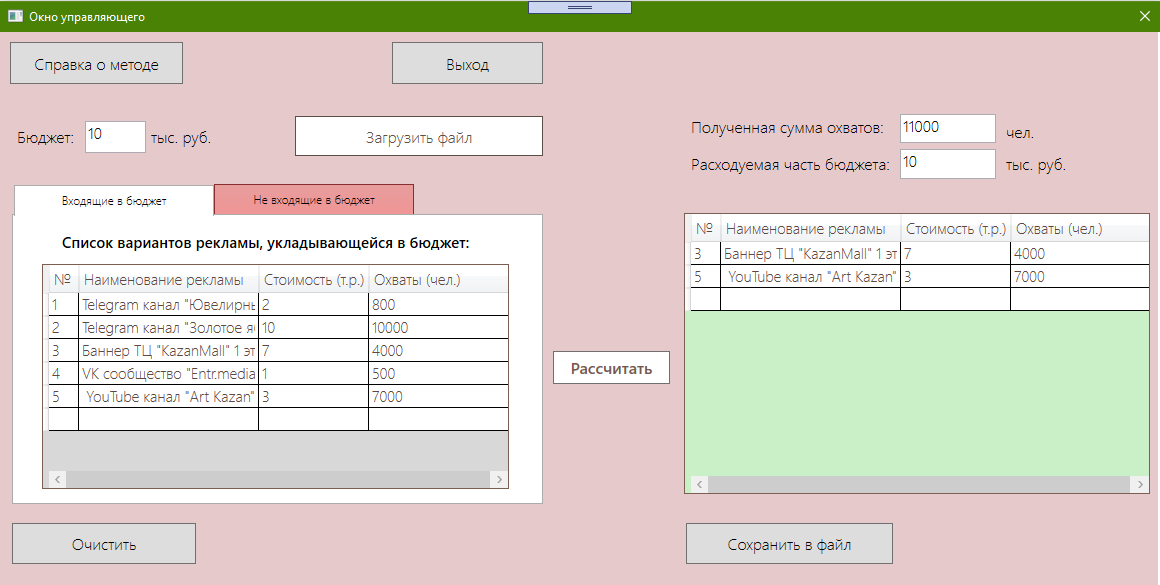


# Система управления базой моделей

После ввода, либо загрузки пользователем данных, в окне программы следует нажать кнопку «Рассчитать», в действие которой и реализуется сам алгоритм. В случае некорректного ввода (к примеру, в файле помимо цифр присутствуют буквы, или остались пустыми некоторые из полей), программа уведомит пользователя об ошибке ввода.



При клике на кнопку «Рассчитать» на окне появляются результирующие данные: таблица с оптимальным набором рекламы, полученный суммарный охват и расходуемая часть бюджета, а также кнопка «Сохранить в файл».



Код алгоритма:

Поскольку алгоритм используется для решения задачи о рюкзаке, все приведенные в коде комментарии связаны с решением этой задачи.

Для реализации алгоритма я создала класс Backpack{}, предназначенный для описания наборов рекламы, включающий в себя, помимо конструктора, следующие методы:

1. public static Backpack Max(Backpack[] b) {}– предназначен для поиска набора рекламы с максимальными суммарными охватами (предназначен для поиска рюкзака с максимальной суммарной ценностью)
2. public static Backpack Fill(Event[] a, Backpack[] bp, int max) – предназначен для заполнения набора рекламой (предназначен для заполнения рюкзака предметами)

class Backpack

{

bool content { get; set; } //в рюкзаке есть что-то или нет

public List<Event> events { get; set; } //список предметов в рюкзаке

public int maxP; //максимальная ценность рюкзака

public static int My\_w;

public static int My\_p;

public int max

{

get { return maxP; }

set { maxP = value; }

}

public Backpack()

{

content = false; //в рюкзаках ничего не лежит

events = new List<Event>();

maxP = 0; //изначально все рюкзаки пустые

}

public static Backpack Max(Backpack[] b) //ищем рюкзак с максимальной суммарной ценностью

{

Backpack res = b[0];

for (int i = 0; i < b.Length; i++)

{

if (res.maxP < b[i].maxP)

res = b[i];

}

return res;

}

public static Backpack Fill(Event[] a, Backpack[] bp, int max)

{

bp[0].content = true; //нулевой рюкзак считаем заполненным

for (int i = 0; i < a.Length; i++) //цикл по всем предметам

{

for (int j = max; j >= a[i].w; j--) //цикл начиная с последнего рюкзака с максимальным весом

//до минимального рюкзака в который можем положить i предмет

{

int k = j - a[i].w; // вычитаем вес предмета

if (bp[k].content == true) //если в полученном рюкзаке уже что-то лежит, то сравниваем ценность

{

if (bp[k].maxP + a[i].p >= bp[j].maxP) //если ценность с новым предметом больше с ценностью уже положенных

{

bp[j].maxP = bp[k].maxP + a[i].p; //меняем ценность

bp[j].events.Clear(); //очищаем предметы в рюкзаке

foreach (var p in bp[k].events) //добавляем все из рюкзака к и добавляем еще i предмет

{

bp[j].events.Add(p);

}

bp[j].events.Add(a[i]);

bp[j].content = true;

}

}

}

}

My\_w = 0;

My\_p = 0;

Backpack res = Backpack.Max(bp); //выбираем рюкзак с наибольшей ценностью

foreach (var p in res.events)

{

My\_w += p.w;

My\_p += p.p;

}

return res;

}

}

Также был создан класс Reklama{}, предназначенный для описания рекламы, которое имеет порядковый номер (num), имя (name), стоимость (w), охват (p).

public class Reklama

{

public Reklama()

{

}

public int num { get; set; }

public string name { get; set; }

public int w { get; set; }

public int p { get; set; }

public Reklama(int Num, string Name, int Weight, int P)

{

num = Num;

name = Name;

w = Weight;

p = P;

}

public static int maxx\_PP;

public static int maxx\_nn;

}

При клике на кнопку «Рассчитать» считывается набор рекламы и бюджет(максимальная суммарная стоимость), затем заполняется список наборов рекламы различных суммарных стоимостей, не превышающих бюджет и ищется наиболее оптимальный набор. Код приведен ниже:

private void calculate\_Button\_Click(object sender, RoutedEventArgs e)

{

try

{

Reklama[] reklama\_list = reklama\_listDataGrid.ItemsSource.Cast<Reklama>().ToArray();

int MaxW = Convert.ToInt32(max.Text);

Backpack[] backpacks = new Backpack[MaxW + 1];

for (int i = 0; i < backpacks.Length; i++)

{

backpacks[i] = new Backpack();

}

var res = Backpack.Fill(reklama\_list, backpacks, MaxW);

var optimized\_reklama\_list = res.reklama\_list;

optimized\_reklama\_listDataGrid.ItemsSource = optimized\_reklama\_list;

OptMax.Text = Backpack.My\_w.ToString();

OptPP.Text = Backpack.My\_p.ToString();

System.Windows.Application.Current.MainWindow = this;

System.Windows.Application.Current.MainWindow.Width = 1176;

if (!(MaxW > 0))

{

MessageBox.Show("Недопустимое значение указанного бюджета!");

return;

}

}

catch

{

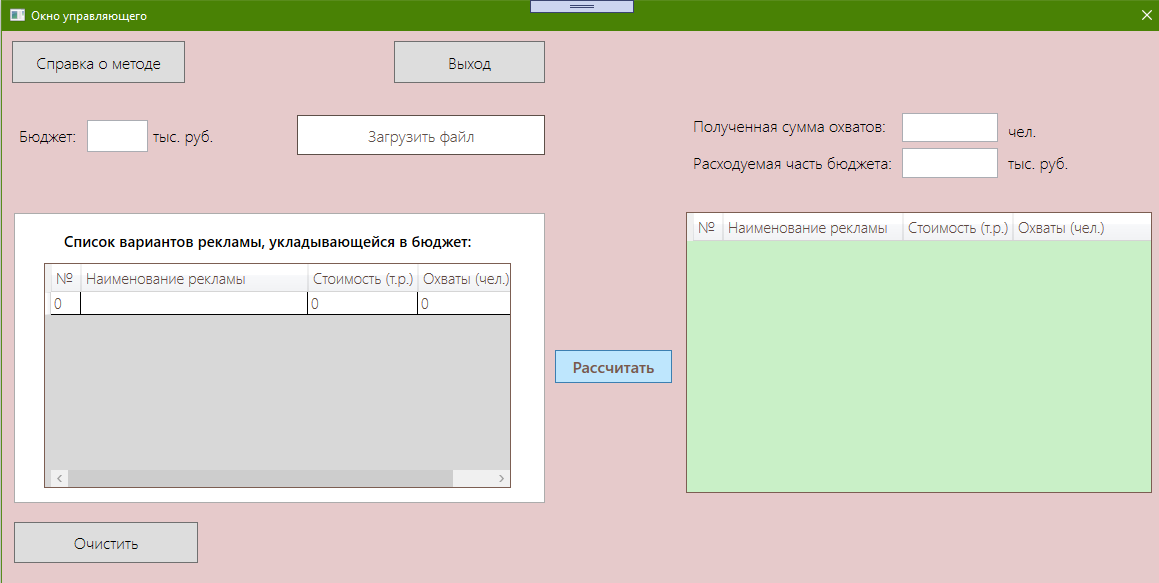
MessageBox.Show("Недопустимые данные!");

return;

}

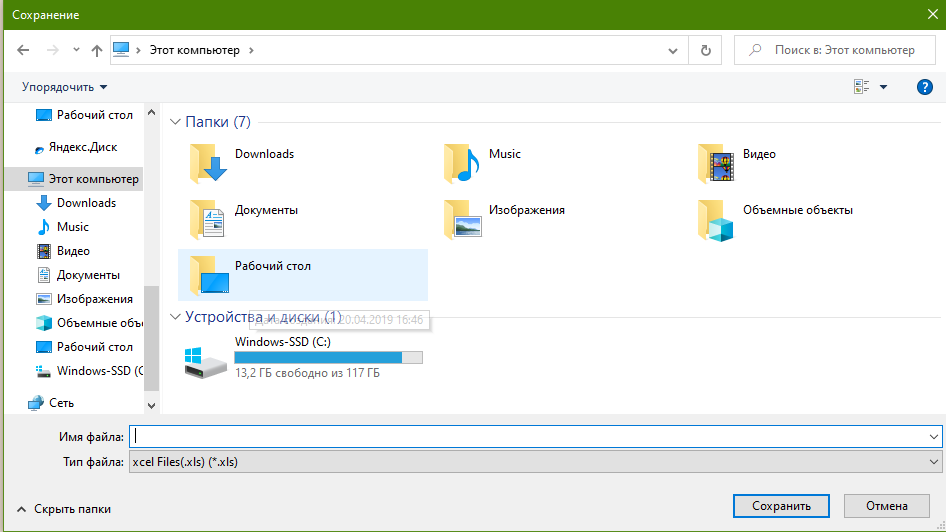
}

При клике на кнопку «Очистить» все заполненные данные удаляются и кнопка «Сохранить в файл» исчезает.

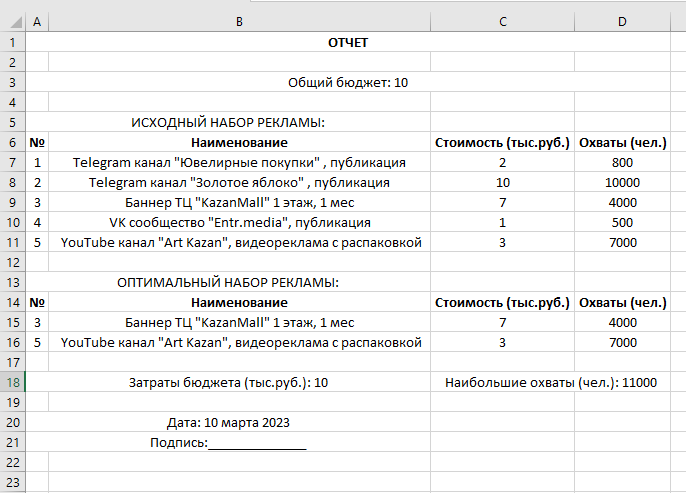


# Отчет и справка

Для создания отчета необходимо после расчета оптимального набора рекламы нажать на кнопку «Сохранить в файл»:



После чего будет сформирован отчет в файле с расширением .xlsx:



Код создания отчета содержится в функции клика на соответствующую кнопку:

private void save\_Button\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "xcel Files(.xls)|\*.xls| xcel Files(.xlsx)| \*.xlsx | Excel Files(\*.xlsm) | \*.xlsm";

Nullable<bool> result = saveFileDialog1.ShowDialog();

if (result == true)

{

Microsoft.Office.Interop.Excel.Application app = new Microsoft.Office.Interop.Excel.Application();

app.Visible = true;

app.WindowState = XlWindowState.xlMaximized;

Workbook wb = app.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);

Worksheet ws = wb.Worksheets[1];

Range excelcells;

DateTime currentDate = DateTime.Now;

Range \_excelCells0 = ws.get\_Range("A1", "D1").Cells;

\_excelCells0.Merge(Type.Missing);

ws.Range["A1"].Value = "ОТЧЕТ";

excelcells = ws.Range["A1"];

excelcells.Font.Bold = true;

ws.get\_Range("A1", "D1").Style.HorizontalAlignment = XlHAlign.xlHAlignCenter;

Range \_excelCells2 = ws.get\_Range("A3", "D3").Cells;

\_excelCells2.Merge(Type.Missing);

ws.Range["A3"].Value = "Общий бюджет: "+ max.Text;

ws.get\_Range("A3", "D3").Style.HorizontalAlignment = XlHAlign.xlHAlignCenter;

Range \_excelCells1 = ws.get\_Range("A5", "B5").Cells;

\_excelCells1.Merge(Type.Missing);

ws.Range["A5"].Value = "ИСХОДНЫЙ НАБОР РЕКЛАМЫ:";

\_excelCells1.Columns.AutoFit();

excelcells = ws.Range["A7"];

excelcells.Value = "№";

excelcells.Font.Bold = true;

excelcells.Columns.AutoFit();

excelcells = ws.Range["B7"];

excelcells.Value = "Наименование";

excelcells.Font.Bold = true;

excelcells = ws.Range["C7"];

excelcells.Value = "Стоимость (тыс.руб.)";

excelcells.Font.Bold = true;

excelcells.Columns.AutoFit();

excelcells = ws.Range["D7"];

excelcells.Value = "Охваты (чел.)";

excelcells.Font.Bold = true;

excelcells.Columns.AutoFit();

int k = 0;

int indStr = 8;

int indStlb = 1;

foreach (Reklama f in reklama\_listDataGrid.ItemsSource)

{

ws.Cells[indStr, indStlb] = f.num.ToString();

ws.Cells[indStr, indStlb+1] = f.name.ToString();

ws.Cells[indStr, indStlb+2] = f.w.ToString();

ws.Cells[indStr, indStlb+3] = f.p.ToString();

indStr++;

k++;

}

indStr++;

indStlb = 2;

Range \_excelCellss = ws.get\_Range("A"+ indStr, "B"+ indStr).Cells;

\_excelCellss.Merge(Type.Missing);

ws.Range["A" + indStr].Value = "ОПТИМАЛЬНЫЙ НАБОР РЕКЛАМЫ:";

\_excelCellss.Columns.AutoFit();

indStr++;

excelcells = ws.Range["A" + indStr];

excelcells.Value = "№";

excelcells.Font.Bold = true;

excelcells = ws.Range["B" + indStr];

excelcells.Value = "Наименование";

excelcells.Font.Bold = true;

excelcells = ws.Range["C" + indStr];

excelcells.Value = "Стоимость (тыс.руб.)";

excelcells.Font.Bold = true;

excelcells = ws.Range["D" + indStr];

excelcells.Value = "Охваты (чел.)";

excelcells.Font.Bold = true;

indStr++;

indStlb = 1;

foreach (Reklama f in optimized\_reklama\_listDataGrid.ItemsSource)

{

ws.Cells[indStr, indStlb] = f.num.ToString();

ws.Cells[indStr, indStlb + 1] = f.name.ToString();

ws.Cells[indStr, indStlb + 2] = f.w.ToString();

ws.Cells[indStr, indStlb + 3] = f.p.ToString();

indStr++;

}

indStr = indStr + 1;

Range \_excelCells22 = ws.get\_Range("A"+ indStr, "B" + indStr).Cells;

\_excelCells22.Merge(Type.Missing);

ws.Range["A" + indStr].Value = "Затраты бюджета (тыс.руб.): " + OptMax.Text;

//ws.Cells[indStr, 1] = "Затраты бюджета: " + OptMax.Text;

Range \_excelCells11 = ws.get\_Range("C" + indStr, "D" + indStr).Cells;

\_excelCells11.Merge(Type.Missing);

ws.Range["C" + indStr].Value = "Наибольшие охваты (чел.): " + OptPP.Text;

indStr = indStr + 2;

Range \_excelCells222 = ws.get\_Range("A" + indStr, "B" + indStr).Cells;

\_excelCells222.Merge(Type.Missing);

ws.Range["A" + indStr].Value = "Дата: " + DateTime.Now.ToString("dd MMMM yyyy");

indStr = indStr + 1;

//ws.Cells[indStr+1, 1] = "Дата:";

//ws.Cells[indStr+1, 2] = DateTime.Now.ToString("dd MMMM yyyy");

Range \_excelCells2222 = ws.get\_Range("A" + indStr, "B" + indStr ).Cells;

\_excelCells2222.Merge(Type.Missing);

ws.Range["A" + indStr ].Value = "Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

//ws.Cells[indStr+1, 5] = "Подпись";

//ws.Cells[indStr+1, 6] = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

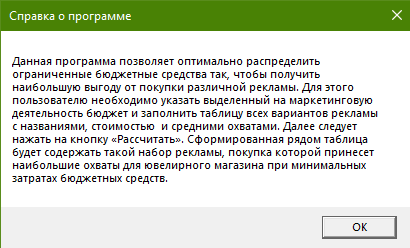
ws.Columns[2].ColumnWidth = 46;

app.Quit();

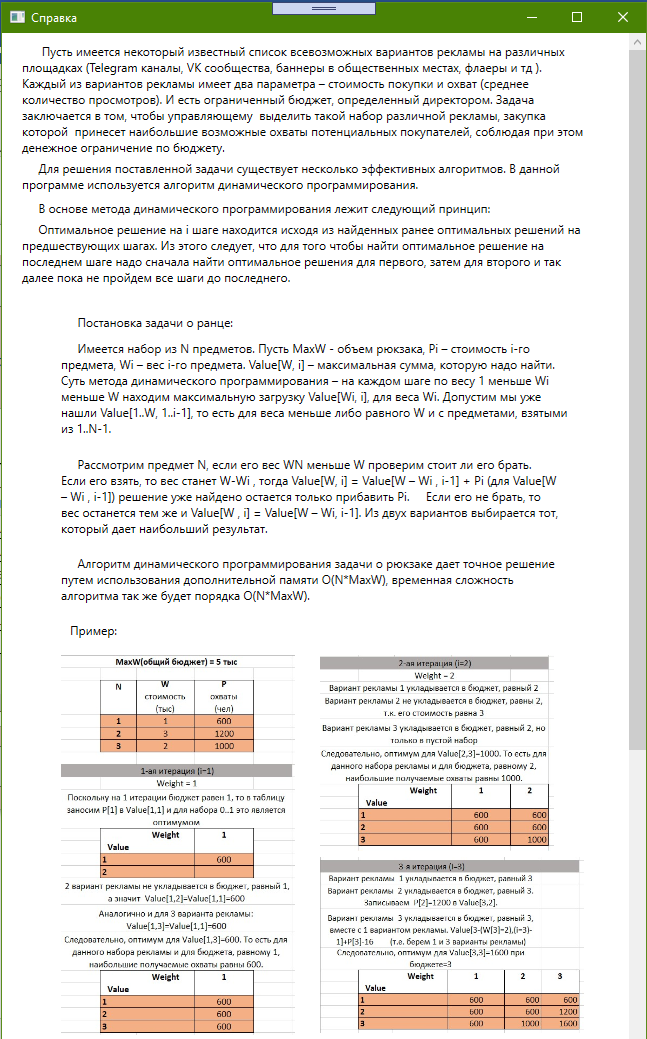
}

}

Информацию по работе с программой пользователь может получить, нажав на кнопку «Справка» в окне «Вход». В справке описывается постановка задачи, а также инструкция по заполнению данными.



Справку об алгоритме, который лежит в основе СППР, пользователь может получить, воспользовавшись кнопкой «Справка» в главном окне:



# Приложение

Окно входа:

<Window x:Class="WpfApp3.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp3"

mc:Ignorable="d"

Title="Вход в систему" Height="517" Width="400">

<Grid Margin="0,0,2,-4">

<Grid.Background>

<SolidColorBrush Color="#FFD28C8C"></SolidColorBrush>

</Grid.Background>

<Label Content="ювелирный магазин" HorizontalAlignment="Left" VerticalAlignment="Top" FontSize="20" Foreground="White" FontFamily="Segoe UI Light" Margin="107,17,0,0"/>

<Label Content="MOMENT jewelry" HorizontalAlignment="Left" VerticalAlignment="Top" FontSize="38" Foreground="White" FontFamily="Segoe UI Light" Margin="55,44,0,0" Width="294"/>

<Border BorderBrush="White" BorderThickness="1" HorizontalAlignment="Left" Height="2" VerticalAlignment="Top" Width="366" Margin="10,103,0,0" />

<Border BorderBrush="White" BorderThickness="1" HorizontalAlignment="Left" Height="2" VerticalAlignment="Top" Width="366" Margin="10,110,0,0"/>

<Button Content="ВОЙТИ" Background="Transparent" HorizontalAlignment="Left" Margin="71,403,0,0" VerticalAlignment="Top" Width="269" RenderTransformOrigin="-7.173,0.152" Height="36" FontSize="15

" BorderBrush="White" Foreground="White" FontWeight="Bold" Click="Button\_Click" >

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="buttonBorder" CornerRadius="0"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

Background="{TemplateBinding Background}" Margin="0,0,124,0">

<ContentControl x:Name="content" Margin="{TemplateBinding Padding}"

HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

VerticalAlignment="{TemplateBinding VerticalContentAlignment}"

Content="{TemplateBinding Content}" />

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="true">

<Setter Property="FontWeight" Value="Bold" />

<Setter TargetName="buttonBorder" Property="Background" Value="Transparent" />

<Setter TargetName="buttonBorder" Property="BorderBrush" Value="SaddleBrown" />

<Setter TargetName="buttonBorder" Property="BorderThickness" Value="1" />

<Setter TargetName="content" Property="Content" Value="ВОЙТИ >>" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

<PasswordBox FontFamily="Segoe UI Light" FontSize="16" Name="pas" HorizontalAlignment="Left" Margin="66,357,0,0" VerticalAlignment="Top" Width="274" Height="25"/>

<Label Height="37" Background="Transparent" FontFamily="Segoe UI Light" FontSize="16" Content="Пароль" HorizontalAlignment="Left" Margin="0,357,0,0" VerticalAlignment="Top" Width="79"/>

<TextBox FontFamily="Segoe UI Light" FontSize="16" Name="log" HorizontalAlignment="Left" Height="25" Margin="66,300,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="274" Background="White"/>

<Label Background="Transparent" Content="Логин" FontFamily="Segoe UI Light" FontSize="16" VerticalAlignment="Top" HorizontalAlignment="Left" Height="35" Margin="0,300,0,0" Width="79"/>

<Button Content="справка" Background="Transparent" HorizontalAlignment="Left" Margin="250,403,-73,0" VerticalAlignment="Top" Width="213" RenderTransformOrigin="-7.173,0.152" Height="36" FontSize="15" BorderBrush="#FF706262" Foreground="#FF706262" Click="Button\_Click\_1" >

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="buttonBorder" CornerRadius="0"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

Background="{TemplateBinding Background}" Margin="0,0,124,0">

<ContentControl x:Name="content" Margin="{TemplateBinding Padding}"

HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

VerticalAlignment="{TemplateBinding VerticalContentAlignment}"

Content="{TemplateBinding Content}" />

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="true">

<Setter Property="FontWeight" Value="Bold" />

<Setter TargetName="buttonBorder" Property="Background" Value="Transparent" />

<Setter TargetName="buttonBorder" Property="BorderBrush" Value="SaddleBrown" />

<Setter TargetName="buttonBorder" Property="BorderThickness" Value="1" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

<Image HorizontalAlignment="Left" Height="152" Margin="66,127,0,0" VerticalAlignment="Top" Width="274" Source="1.jpg" Stretch="UniformToFill"/>

</Grid>

</Window>

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO;

using Microsoft.Win32;

using System.Diagnostics;

using System.Runtime.InteropServices;

namespace WpfApp3

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (log.Text == "admin" && pas.Password == "admin")

{

Window11 win1 = new Window11();

this.Close();

win1.Show();

System.Windows.Application.Current.MainWindow = win1;

System.Windows.Application.Current.MainWindow.Width = 689;

}

else

{

MessageBox.Show("Неправильный логин или пароль!");

pas.Password = "";

log.Text = "";

}

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

MessageBox.Show("Данная программа позволяет оптимально распределить ограниченные бюджетные средства так, чтобы получить наибольшую выгоду от покупки различной рекламы. Для этого пользователю необходимо указать выделенный на маркетинговую деятельность бюджет и заполнить таблицу всех вариантов рекламы с названиями, стоимостью и средними охватами. Далее следует нажать на кнопку «Рассчитать». Сформированная рядом таблица будет содержать такой набор рекламы, покупка которой принесет наибольшие охваты для ювелирного магазина при минимальных затратах бюджетных средств.", "Справка о программе");

}

}

}

Окно управляющего:

<Window x:Class="WpfApp3.Window11"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp3"

mc:Ignorable="d"

Title="Окно управляющего" Width="1176" Height="593" WindowStartupLocation="CenterScreen" ResizeMode="NoResize">

<Grid Background="#FFE6CACB" Name="graf" Margin="0,0,2,-3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="553"/>

<ColumnDefinition Width="597"/>

<ColumnDefinition Width="0"/>

</Grid.ColumnDefinitions>

<Button FontFamily="Segoe UI Light" FontSize="16" Content="Очистить" Click="delete\_Button\_Click" Margin="12,491,357,25"/>

<Label FontSize="16" FontFamily="Segoe UI Light" Content="Расходуемая часть бюджета:" Margin="133,116,121,386" Grid.Column="1"/>

<Button Height="40" FontFamily="Segoe UI Light" VerticalAlignment="Top" Click="dowload\_Button\_Click" Margin="295,84,10,0" FontSize="16" Content="Загрузить файл" Background="White" BorderBrush="#FF5D4F48" Foreground="#FF5D4F48"/>

<Button FontSize="16" VerticalAlignment="Top" FontFamily="Segoe UI Semibold" Click="calculate\_Button\_Click" Margin="0,319,480,0" Height="33" Content="Рассчитать" Background="White" Foreground="#FF78574A" FontWeight="Bold" Grid.Column="1"/>

<TextBox FontFamily="Segoe UI Light" FontSize="16" Name="max" HorizontalAlignment="Left" Height="32" Margin="85,89,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="61"/>

<Button FontFamily="Segoe UI Light" FontSize="16" Content="Справка о методе" Click="info\_Button\_Click" Margin="10,10,370,505"/>

<Label FontFamily="Segoe UI Light" FontSize="16" Content="Бюджет:" Margin="12,89,421,417" />

<Label FontFamily="Segoe UI Light" FontSize="16" Content="тыс. руб." Margin="146,89,267,417" />

<TextBox IsReadOnly="true" FontFamily="Segoe UI Light" FontSize="16" Name="OptMax" HorizontalAlignment="Left" Height="30" Margin="347,117,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="96" Grid.Column="1"/>

<Label FontSize="16" FontFamily="Segoe UI Light" Content="Полученная сумма охватов:" Margin="133,79,106,420" Grid.Column="1"/>

<TextBox IsReadOnly="true" FontFamily="Segoe UI Light" FontSize="16" x:Name="OptPP" HorizontalAlignment="Left" Height="29" Margin="347,82,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="96" Grid.Column="1"/>

<Label FontFamily="Segoe UI Light" FontSize="16" Content="тыс. руб." Margin="448,116,32,392" Grid.Column="1" />

<DataGrid AutoGenerateColumns="False" FontSize="15" FontFamily="Segoe UI Light" Name="optimized\_reklama\_listDataGrid" Margin="131,181,0,95" BorderBrush="#FF7C5E52" Foreground="#FF5F4840" Background="#FFC9F0C7" Grid.Column="1" Grid.ColumnSpan="2">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding num}" CanUserResize="False" Header="№" Width="30"/>

<DataGridTextColumn Binding="{Binding name}" CanUserResize="True" Header="Наименование рекламы" Width="180"/>

<DataGridTextColumn Binding="{Binding w}" CanUserResize="False" Header="Стоимость (т.р.)" Width="110"/>

<DataGridTextColumn Binding="{Binding p}" CanUserResize="False" Header="Охваты (чел.)" Width="140"/>

</DataGrid.Columns>

</DataGrid>

<Button FontFamily="Segoe UI Light" FontSize="16" Content="Выход" Click="exit\_Button\_Click" Margin="392,10,10,505"/>

<Button Name="save" FontFamily="Segoe UI Light" FontSize="16" Content="Сохранить в файл" Click="save\_Button\_Click" Margin="133,491,257,25" Grid.Column="1"/>

<TabControl HorizontalAlignment="Left" Height="322" Margin="12,150,0,0" VerticalAlignment="Top" Width="531" SelectionChanged="TabControl\_SelectionChanged">

<TabItem Name="tab1" Header="Входящие в бюджет" Margin="0,0,0,-2" FontWeight="Bold" FontFamily="Segoe UI Light" Width="200" HorizontalAlignment="Center" Height="30" BorderBrush="#FF3B6637" Cursor="Hand" Visibility="Hidden" AutomationProperties.Name="tabs">

<TabItem.Background>

<LinearGradientBrush EndPoint="0,1" StartPoint="0,0">

<GradientStop Color="#FFDEDEDE" Offset="0"/>

<GradientStop Color="#FFC1C7BB" Offset="1"/>

</LinearGradientBrush>

</TabItem.Background>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label FontFamily="Segoe UI Semibold" FontSize="15" Content="Список вариантов рекламы, укладывающейся в бюджет:" FontStyle="Italic" FontWeight="Bold" Margin="42,10,59,237" />

<DataGrid MinWidth="467" AutoGenerateColumns="False" CanUserAddRows="true" FontSize="15" FontFamily="Segoe UI Light" ItemsSource="{Binding GridCollection}" Name="reklama\_listDataGrid" BorderBrush="#FF7C5E52" Foreground="#FF5F4840" Background="#FFD8D8D8" VerticalAlignment="Top" Height="225" HorizontalAlignment="Left" MaxWidth="463" Margin="27,47,0,0" Width="476">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding num}" CanUserResize="False" x:Name="num" Header="№" Width="30"/>

<DataGridTextColumn Binding="{Binding name}" CanUserResize="True" x:Name="name" Header="Наименование рекламы" Width="180"/>

<DataGridTextColumn Binding="{Binding w}" CanUserResize="False" x:Name="w" Header="Стоимость (т.р.)" Width="110"/>

<DataGridTextColumn Binding="{Binding p}" CanUserResize="False" x:Name="p" Header="Охваты (чел.)" Width="140"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</TabItem>

<TabItem Name="tab2" FontWeight="Bold" FontFamily="Segoe UI Light" Header="Не входящие в бюджет" HorizontalAlignment="Center" Height="30" Width="200" Margin="0" BorderBrush="#FF892E2E" Cursor="Hand" Visibility="Hidden">

<TabItem.Background>

<LinearGradientBrush EndPoint="0,1" StartPoint="0,0">

<GradientStop Color="#FFE89D9D" Offset="0"/>

<GradientStop Color="#FFF09696" Offset="1"/>

<GradientStop Color="#FFEC9898" Offset="0.612"/>

</LinearGradientBrush>

</TabItem.Background>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label FontFamily="Segoe UI Semibold" FontSize="15" Content="Список вариантов рекламы, не укладывающейся в бюджет:" Margin="44,10,10,212" FontStyle="Italic" FontWeight="Bold" RenderTransformOrigin="0.5,0.5" />

<DataGrid IsReadOnly="true" AutoGenerateColumns="False" CanUserAddRows="true" FontSize="15" FontFamily="Segoe UI Light" MinWidth="467" x:Name="NONreklama\_listDataGrid" BorderBrush="#FF7C5E52" Foreground="#FF5F4840" Background="#FFFFAEAE" VerticalAlignment="Top" Height="220" HorizontalAlignment="Left" MaxWidth="463" Margin="23,51,0,0" Width="481">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding num}" CanUserResize="False" x:Name="num1" Header="№" Width="30"/>

<DataGridTextColumn Binding="{Binding name}" CanUserResize="True" x:Name="name1" Header="Наименование рекламы" Width="180"/>

<DataGridTextColumn Binding="{Binding w}" CanUserResize="False" x:Name="w1" Header="Стоимость (т.р.)" Width="110"/>

<DataGridTextColumn Binding="{Binding p}" CanUserResize="False" x:Name="p1" Header="Охваты (чел.)" Width="140"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</TabItem>

</TabControl>

<Label FontFamily="Segoe UI Light" FontSize="16" Content="чел." Margin="448,84,32,424" Grid.Column="1" />

</Grid>

</Window>

using System;

using Microsoft.Office.Interop.Excel;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using Microsoft.Win32;

using System.Diagnostics;

using ClosedXML.Excel;

using System.Collections.ObjectModel;

namespace WpfApp3

{

/// <summary>

/// Логика взаимодействия для Window11.xaml

/// </summary>

public class Reklama

{

public Reklama()

{

}

public int num { get; set; }

public string name { get; set; }

public int w { get; set; }

public int p { get; set; }

public Reklama(int Num, string Name, int Weight, int P)

{

num = Num;

name = Name;

w = Weight;

p = P;

}

public static int maxx\_PP;

public static int maxx\_nn;

}

class Backpack

{

bool content { get; set; } //в рюкзаке есть что-то или нет

public List<Reklama> reklama\_list { get; set; } //список предметов в рюкзаке

public int maxP; //максимальная ценность рюкзака

public static int My\_w;

public static int My\_p;

public int max

{

get { return maxP; }

set { maxP = value; }

}

public Backpack()

{

content = false; //в рюкзаках ничего не лежит

reklama\_list = new List<Reklama>();

maxP = 0; //изначально все рюкзаки пустые

}

public static Backpack Max(Backpack[] b) //ищем рюкзак с максимальной суммарной ценностью

{

Backpack res = b[0];

for (int i = 0; i < b.Length; i++)

{

if (res.maxP < b[i].maxP)

res = b[i];

}

return res;

}

public static Backpack Fill(Reklama[] a, Backpack[] bp, int max)

{

bp[0].content = true; //нулевой рюкзак считаем заполненным

for (int i = 0; i < a.Length; i++) //цикл по всем предметам

{

for (int j = max; j >= a[i].w; j--) //цикл начиная с последнего рюкзака с максимальным весом

//до минимального рюкзака в который можем положить i предмет

{

int k = j - a[i].w; // вычитаем вес предмета

if (bp[k].content == true) //если в полученном рюкзаке уже что-то лежит, то сравниваем ценность

{

if (bp[k].maxP + a[i].p >= bp[j].maxP) //если ценность с новым предметом больше с ценностью уже положенных

{

bp[j].maxP = bp[k].maxP + a[i].p; //меняем ценность

bp[j].reklama\_list.Clear(); //очищаем предметы в рюкзаке

foreach (var p in bp[k].reklama\_list) //добавляем все из рюкзака к и добавляем еще i предмет

{

bp[j].reklama\_list.Add(p);

}

bp[j].reklama\_list.Add(a[i]);

bp[j].content = true;

}

}

}

}

My\_w = 0;

My\_p = 0;

Backpack res = Backpack.Max(bp); //выбираем рюкзак с наибольшей ценностью

foreach (var p in res.reklama\_list)

{

My\_w += p.w;

My\_p += p.p;

}

return res;

}

}

public partial class Window11

{

public Window11()

{

InitializeComponent();

DataContext = this;

GridCollection = new ObservableCollection<Reklama>();

}

public ObservableCollection<Reklama> GridCollection { get; set; }

public static IEnumerable<Reklama> fromExcel()

{

MessageBox.Show("В ячейке A2 необходимо указать общий бюджет. Далее, начиная со строки А[5], заполнять поочередно в ячейках информацию о каждой рекламе. В столбце А[1] указать номер варианта рекламы, в столбце А[2] - наименование, в столбце А[3] - стоимость покупки в тыс.руб. и в столбце А[4] - средние охваты (чел.)."

, "Требования к структуре файла");

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "xcel Files(.xls)|\*.xls| xcel Files(.xlsx)| \*.xlsx | Excel Files(\*.xlsm) | \*.xlsm";

Nullable<bool> result = openFileDialog1.ShowDialog();

Microsoft.Office.Interop.Excel.Application ExcelApp;

Workbook excelappworkbook;

Microsoft.Office.Interop.Excel.Worksheet worksheet;

Range excelcells, excelcells1, excelcells2, excelcells3, excelcells4;

if (result == true)

{

string fileName = System.IO.Path.GetFileName(openFileDialog1.FileName);

ExcelApp = new Microsoft.Office.Interop.Excel.Application();

excelappworkbook = ExcelApp.Workbooks.Open(openFileDialog1.FileName);

worksheet = (Microsoft.Office.Interop.Excel.Worksheet)excelappworkbook.Sheets[1];

excelcells = worksheet.Cells[2, 1];

Reklama.maxx\_PP = Convert.ToInt32(excelcells.Value);

excelcells = worksheet.Cells[2, 2];

Reklama.maxx\_nn = Convert.ToInt32(excelcells.Value);

//maxx\_PP = worksheet.Cell(2, 1).GetValue<int>();

//maxx\_PP = worksheet.Cell(2, 1).GetValue<int>();

{

// Перебираем диапазон нужных строк

for (int row = 5; row < 5 + Reklama.maxx\_nn; ++row)

{

excelcells1 = worksheet.Cells[row, 1];

excelcells2 = worksheet.Cells[row, 2];

excelcells3 = worksheet.Cells[row, 3];

excelcells4 = worksheet.Cells[row, 4];

// По каждой строке формируем объект

Reklama reklama\_i = new Reklama

{

num = Convert.ToInt32(excelcells1.Value),

name = Convert.ToString(excelcells2.Value),

w = Convert.ToInt32(excelcells3.Value),

p = Convert.ToInt32(excelcells4.Value),

};

// И возвращаем его

//workbook.Close();

yield return reklama\_i;

}

}

ExcelApp.Quit();

}

}

private void dowload\_Button\_Click(object sender, RoutedEventArgs e)

{

var reklama\_list = fromExcel().ToList();

var ok\_reklama\_list = reklama\_list.Where(x => x.w <= Reklama.maxx\_PP).ToList();

var non\_reklama\_list = reklama\_list.Where(x => x.w > Reklama.maxx\_PP).ToList();

reklama\_listDataGrid.ItemsSource = ok\_reklama\_list;

NONreklama\_listDataGrid.ItemsSource = non\_reklama\_list;

max.Text = Reklama.maxx\_PP.ToString();

tab1.Visibility = Visibility.Visible;

if (non\_reklama\_list.Count!=0)

tab2.Visibility = Visibility.Visible;

}

private void calculate\_Button\_Click(object sender, RoutedEventArgs e)

{

try

{

Reklama[] reklama\_list = reklama\_listDataGrid.ItemsSource.Cast<Reklama>().ToArray();

int MaxW = Convert.ToInt32(max.Text);

Backpack[] backpacks = new Backpack[MaxW + 1];

for (int i = 0; i < backpacks.Length; i++)

{

backpacks[i] = new Backpack();

}

var res = Backpack.Fill(reklama\_list, backpacks, MaxW);

var optimized\_reklama\_list = res.reklama\_list;

optimized\_reklama\_listDataGrid.ItemsSource = optimized\_reklama\_list;

OptMax.Text = Backpack.My\_w.ToString();

OptPP.Text = Backpack.My\_p.ToString();

System.Windows.Application.Current.MainWindow = this;

System.Windows.Application.Current.MainWindow.Width = 1176;

if (!(MaxW > 0))

{

MessageBox.Show("Недопустимое значение указанного бюджета!");

return;

}

}

catch

{

MessageBox.Show("Недопустимые данные!");

return;

}

}

//очистить

private void delete\_Button\_Click(object sender, RoutedEventArgs e)

{

//делаем кнопку "Сохранить в файл" невидимой

save.Visibility = Visibility.Hidden;

tab1.Visibility = Visibility.Hidden;

tab2.Visibility = Visibility.Hidden;

max.Text = "";

reklama\_listDataGrid.ItemsSource = null;

reklama\_listDataGrid.Items.Refresh();

NONreklama\_listDataGrid.ItemsSource = null;

NONreklama\_listDataGrid.Items.Refresh();

OptMax.Text = "";

OptPP.Text = "";

optimized\_reklama\_listDataGrid.ItemsSource = null;

optimized\_reklama\_listDataGrid.Items.Refresh();

Reklama[] del\_event = new Reklama[] { new Reklama()};

//GridCollection = new ObservableCollection<Reklama>();

reklama\_listDataGrid.Items.Add(del\_event);

reklama\_listDataGrid.Items.Refresh();

}

private void info\_Button\_Click(object sender, RoutedEventArgs e)

{

win2 win22 = new win2();

win22.Show();

}

private void exit\_Button\_Click(object sender, RoutedEventArgs e)

{

MainWindow win1 = new MainWindow();

this.Close();

win1.Show();

}

//сохранить в файл exel

private void save\_Button\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "xcel Files(.xls)|\*.xls| xcel Files(.xlsx)| \*.xlsx | Excel Files(\*.xlsm) | \*.xlsm";

Nullable<bool> result = saveFileDialog1.ShowDialog();

if (result == true)

{

Microsoft.Office.Interop.Excel.Application app = new Microsoft.Office.Interop.Excel.Application();

app.Visible = true;

app.WindowState = XlWindowState.xlMaximized;

Workbook wb = app.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);

Worksheet ws = wb.Worksheets[1];

Range excelcells;

DateTime currentDate = DateTime.Now;

Range \_excelCells0 = ws.get\_Range("A1", "D1").Cells;

\_excelCells0.Merge(Type.Missing);

ws.Range["A1"].Value = "ОТЧЕТ";

excelcells = ws.Range["A1"];

excelcells.Font.Bold = true;

ws.get\_Range("A1", "D1").Style.HorizontalAlignment = XlHAlign.xlHAlignCenter;

Range \_excelCells2 = ws.get\_Range("A3", "D3").Cells;

\_excelCells2.Merge(Type.Missing);

ws.Range["A3"].Value = "Общий бюджет: "+ max.Text;

ws.get\_Range("A3", "D3").Style.HorizontalAlignment = XlHAlign.xlHAlignCenter;

Range \_excelCells1 = ws.get\_Range("A5", "B5").Cells;

\_excelCells1.Merge(Type.Missing);

ws.Range["A5"].Value = "ИСХОДНЫЙ НАБОР РЕКЛАМЫ:";

\_excelCells1.Columns.AutoFit();

excelcells = ws.Range["A7"];

excelcells.Value = "№";

excelcells.Font.Bold = true;

excelcells.Columns.AutoFit();

excelcells = ws.Range["B7"];

excelcells.Value = "Наименование";

excelcells.Font.Bold = true;

excelcells = ws.Range["C7"];

excelcells.Value = "Стоимость (тыс.руб.)";

excelcells.Font.Bold = true;

excelcells.Columns.AutoFit();

excelcells = ws.Range["D7"];

excelcells.Value = "Охваты (чел.)";

excelcells.Font.Bold = true;

excelcells.Columns.AutoFit();

int k = 0;

int indStr = 8;

int indStlb = 1;

foreach (Reklama f in reklama\_listDataGrid.ItemsSource)

{

ws.Cells[indStr, indStlb] = f.num.ToString();

ws.Cells[indStr, indStlb+1] = f.name.ToString();

ws.Cells[indStr, indStlb+2] = f.w.ToString();

ws.Cells[indStr, indStlb+3] = f.p.ToString();

indStr++;

k++;

}

indStr++;

indStlb = 2;

Range \_excelCellss = ws.get\_Range("A"+ indStr, "B"+ indStr).Cells;

\_excelCellss.Merge(Type.Missing);

ws.Range["A" + indStr].Value = "ОПТИМАЛЬНЫЙ НАБОР РЕКЛАМЫ:";

\_excelCellss.Columns.AutoFit();

indStr++;

excelcells = ws.Range["A" + indStr];

excelcells.Value = "№";

excelcells.Font.Bold = true;

excelcells = ws.Range["B" + indStr];

excelcells.Value = "Наименование";

excelcells.Font.Bold = true;

excelcells = ws.Range["C" + indStr];

excelcells.Value = "Стоимость (тыс.руб.)";

excelcells.Font.Bold = true;

excelcells = ws.Range["D" + indStr];

excelcells.Value = "Охваты (чел.)";

excelcells.Font.Bold = true;

indStr++;

indStlb = 1;

foreach (Reklama f in optimized\_reklama\_listDataGrid.ItemsSource)

{

ws.Cells[indStr, indStlb] = f.num.ToString();

ws.Cells[indStr, indStlb + 1] = f.name.ToString();

ws.Cells[indStr, indStlb + 2] = f.w.ToString();

ws.Cells[indStr, indStlb + 3] = f.p.ToString();

indStr++;

}

indStr = indStr + 1;

Range \_excelCells22 = ws.get\_Range("A"+ indStr, "B" + indStr).Cells;

\_excelCells22.Merge(Type.Missing);

ws.Range["A" + indStr].Value = "Затраты бюджета (тыс.руб.): " + OptMax.Text;

//ws.Cells[indStr, 1] = "Затраты бюджета: " + OptMax.Text;

Range \_excelCells11 = ws.get\_Range("C" + indStr, "D" + indStr).Cells;

\_excelCells11.Merge(Type.Missing);

ws.Range["C" + indStr].Value = "Наибольшие охваты (чел.): " + OptPP.Text;

indStr = indStr + 2;

Range \_excelCells222 = ws.get\_Range("A" + indStr, "B" + indStr).Cells;

\_excelCells222.Merge(Type.Missing);

ws.Range["A" + indStr].Value = "Дата: " + DateTime.Now.ToString("dd MMMM yyyy");

indStr = indStr + 1;

//ws.Cells[indStr+1, 1] = "Дата:";

//ws.Cells[indStr+1, 2] = DateTime.Now.ToString("dd MMMM yyyy");

Range \_excelCells2222 = ws.get\_Range("A" + indStr, "B" + indStr ).Cells;

\_excelCells2222.Merge(Type.Missing);

ws.Range["A" + indStr ].Value = "Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

//ws.Cells[indStr+1, 5] = "Подпись";

//ws.Cells[indStr+1, 6] = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

ws.Columns[2].ColumnWidth = 46;

app.Quit();

}

}

Окно справки:

<Window x:Class="WpfApp3.win2"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp3"

mc:Ignorable="d"

Title="Справка" Height="1599.901" Width="612.745">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<Grid Margin="0,0,2,-3">

<TextBlock HorizontalAlignment="Left" Margin="20,10,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="562"><Run Text=" Пусть имеется некоторый известный список всевозможных "/><Run Text="вариантов рекламы на различных площадках ("/><Run Text="Telegram"/><Run Text=" каналы"/><Run Text=", "/><Run Text="VK"/><Run Text=" сообщества, баннеры в общественных местах, флаеры и тд"/><Run Text=" ). Каждый из вариантов рекламы "/><Run Text="имеет два параметра – стоимость "/><Run Text="покупки"/><Run Text=" и "/><Run Text="охват ("/><Run Text="среднее "/><Run Text="количеств"/><Run Text="о"/><Run Text=" п"/><Run Text="росмотров"/><Run Text="). И есть ограниченный бюджет, определенный директором. Задача заключается в том, чтобы управляющему выделить такой набор "/><Run Text="различной рекламы"/><Run Text=", "/><Run Text="закупка которой "/><Run Text=" принесет "/><Run Text="наибольшие возможные охваты потенциальных покупателей"/><Run Text=", соблюдая при этом денежное ограничение по бюджету."/></TextBlock>

<TextBlock HorizontalAlignment="Left" Margin="20,127,0,0" TextWrapping="Wrap" Text=" Для решения поставленной задачи существует несколько эффективных алгоритмов. В данной программе используется алгоритм динамического программирования." VerticalAlignment="Top" Width="562"/>

<TextBlock HorizontalAlignment="Left" Margin="20,167,0,0" TextWrapping="Wrap" Text=" В основе метода динамического программирования лежит следующий принцип:" VerticalAlignment="Top"/>

<TextBlock HorizontalAlignment="Left" Margin="20,188,0,0" TextWrapping="Wrap" Text=" Оптимальное решение на i шаге находится исходя из найденных ранее оптимальных решений на предшествующих шагах. Из этого следует, что для того чтобы найти оптимальное решение на последнем шаге надо сначала найти оптимальное решения для первого, затем для второго и так далее пока не пройдем все шаги до последнего." VerticalAlignment="Top" Width="562"/>

<TextBlock HorizontalAlignment="Left" Margin="59,281,0,0" TextWrapping="Wrap" Text=" Постановка задачи о ранце:" VerticalAlignment="Top" RenderTransformOrigin="0.498,0.398"/>

<TextBlock HorizontalAlignment="Left" Margin="59,307,0,0" TextWrapping="Wrap" Text=" Имеется набор из N предметов. Пусть MaxW - объем рюкзака, Pi – стоимость i-го предмета, Wi – вес i-го предмета. Value[W, i] – максимальная сумма, которую надо найти. Суть метода динамического программирования – на каждом шаге по весу 1 меньше Wi меньше W находим максимальную загрузку Value[Wi, i], для веса Wi. Допустим мы уже нашли Value[1..W, 1..i-1], то есть для веса меньше либо равного W и с предметами, взятыми из 1..N-1." VerticalAlignment="Top" Width="510"/>

<TextBlock HorizontalAlignment="Left" Margin="59,423,0,0" TextWrapping="Wrap" Text=" Рассмотрим предмет N, если его вес WN меньше W проверим стоит ли его брать.

Если его взять, то вес станет W-Wi , тогда Value[W, i] = Value[W – Wi , i-1] + Pi (для Value[W – Wi , i-1]) решение уже найдено остается только прибавить Pi.

Если его не брать, то вес останется тем же и Value[W , i] = Value[W – Wi, i-1]. Из двух вариантов выбирается тот, который дает наибольший результат.

" VerticalAlignment="Top" Width="501"/>

<TextBlock HorizontalAlignment="Left" Margin="59,522,0,0" TextWrapping="Wrap" Text=" Алгоритм динамического программирования задачи о рюкзаке дает точное решение путем использования дополнительной памяти O(N\*MaxW), временная сложность алгоритма так же будет порядка O(N\*MaxW).

" VerticalAlignment="Top" Width="501"/>

<Button FontFamily="Segoe UI Light" FontSize="16" Content="Закрыть" Click="exit\_Button\_Click" Margin="466,1514,34,21"/>

<TextBlock HorizontalAlignment="Left" Margin="68,589,0,0" TextWrapping="Wrap" Text="Пример:" VerticalAlignment="Top"/>

<Image HorizontalAlignment="Left" Height="387" Margin="59,622,0,0" VerticalAlignment="Top" Width="234" Source="1ит.jpg"/>

<Image HorizontalAlignment="Left" Height="202" Margin="318,622,0,0" VerticalAlignment="Top" Width="234" Source="2ит.jpg"/>

<Image HorizontalAlignment="Left" Height="197" Margin="318,824,0,0" VerticalAlignment="Top" Width="264" Source="3ит.jpg"/>

<Image HorizontalAlignment="Left" Height="216" Margin="135,1026,0,0" VerticalAlignment="Top" Width="317" Source="4ит.jpg"/>

<Image HorizontalAlignment="Left" Height="216" Margin="135,1270,0,0" VerticalAlignment="Top" Width="317" Source="5ит.jpg"/>

</Grid>

</ScrollViewer>

</Window>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfApp3

{

/// <summary>

/// Логика взаимодействия для win2.xaml

/// </summary>

public partial class win2 : Window

{

public win2()

{

InitializeComponent();

}

private void exit\_Button\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

}