МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ

ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)

Кафедра

«Автоматизированные системы управления»

Курсовая работа по дисциплине

«Объектно-ориентированное программирование»

Вариант № 21

Тема:

«Разработка информационной системы «Хлебозавод»

Выполнил: студент группы \_\_\_\_\_\_

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2019

Оглавление

[ЗАДАНИЕ 3](#_Toc11942932)

[ВВЕДЕНИЕ 5](#_Toc11942933)

[1 ОСНОВНОЙ РАЗДЕЛ 6](#_Toc11942934)

[1.1. Классы программы 6](#_Toc11942935)

[1.2. Коллекции программы 7](#_Toc11942936)

[2 ТЕКСТ ПРОГРАММЫ 7](#_Toc11942937)

[3 ФАЙЛЫ ДЛЯ ИНИЦИАЛИЗАЦИИ СПИСКОВ 13](#_Toc11942938)

[ЗАКЛЮЧЕНИЕ 15](#_Toc11942939)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc11942940)

# ЗАДАНИЕ

Требуется разработать программную систему, предназначенную для обработки данных о хлебозаводах. Такая система должна обеспечивать хранение сведений о производимых изделиях и их стоимости.

Сведения о производимых изделиях включают в себя: наименование изделия, объем производства, дату производства, срок годности, ингредиенты, отпускную цену. Каждый состав характеризуется наименованием, количеством, датой поставки, сроком годности. Состав изделия содержит список возможных ингредиентов: вода, мука пшеничная, мука ржаная, отруби, сахар, масло подсолнечное, соль, дрожжи, добавки. В составе могут отсутствовать некоторые ингредиенты (сахар, масло).

**Хлебозавод Изделие Состав**

Ключ Ключ Ключ

Название Название Ингредиент 1

Дата производства Вес изделия Ингредиент 2

Текущая дата Срок годности Ингредиент 3

Объем производства Ингредиент 4

Цена Ингредиент 5

Ингредиент 6

Вес

Дата поставки

Срок годности

Должны быть созданы обобщенные списки:

• Сведения о хлебозаводе.

• Сведения об изделии.

• Сведения о складе.

Руководству комбината могут понадобиться следующие сведения:

• Какие изделия выпускает заданный хлебозавод.

• Какова суммарная стоимость всех изделий, выпускаемых заданным хлебозаводом.

• Не было ли изделий, производство которых было выполнено после истечения срока какого-либо ингредиента.

• Отсортировать хлебозаводы в порядке убывания объема производства.

• В каком изделии больше всего ингредиентов.

• Какое изделие обеспечит при реализации максимальную прибыль.

Должна быть предусмотрена возможность добавления и удаления изделия для заданного хлебозавода.

# ВВЕДЕНИЕ

Целью курсовой работы является разработка в среде Microsoft Visual Studio на языке программирования C# консольного приложения Windows для работы с локальными обобщенными списками. Информация для списков должна сохраняться в файлах, реализованных с помощью приложения «Блокнот». Каждый из списков должен содержать не менее 10 инициализированных объектов класса.

Задачей курсовой работы является закрепление изученного материала и освоение методов обработки информации помощью запросов LINQ.

Программа, реализованная в результате работы, предназначена для управления хлебопекарным бизнесом и позволяет анализировать информацию о деятельности нескольких предприятий.

Основные действия, доступные пользователю программной системы:

1. Просмотр списка изделий, выпускаемых заданным хлебозаводом.
2. Подсчёт суммарной стоимости изделий, выпускаемых хлебозаводом.
3. Просмотр списка изделий с просроченными ингредиентами.
4. Сортировка хлебозаводов в порядке убывания объёма производства.
5. Выявление изделий с наибольшим количеством ингредиентов.
6. Выявление изделий с наибольшей прибылью при реализации.
7. Добавление изделий в список производимой продукции.
8. Удаление изделий из списка производимой продукции.

# 1 ОСНОВНОЙ РАЗДЕЛ

## Классы программы

Класс Factory содержит информацию об одном хлебозаводе. Члены класса приведены в Таблице\_1.

Таблица\_1. Поля класса Factory

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Key | int | Уникальный ключ записи |
| Name | string | Название хлебозавода |
| ProdDate | DateTime | Дата начала производства |

Класс Product содержит данные об одном выпуске конкретной продукции – иначе говоря, речь идёт о целой партии, наименование может совпадать, а дата, вес и другие поля могут отличаться. Члены класса приведены в Таблице\_2.

Таблица\_2. Поля класса Product

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Key | int | Уникальный ключ записи |
| FactoryKey | int | Ключ записи хлебозавода |
| Name | string | Наименование продукции |
| Weight | double | Вес выпущенной продукции |
| ExpiryDate | DateTime | Дата истечения срока годности |
| Count | int | Объём выпущенной партии |
| Price | double | Цена единицы продукции |

Класс Ingredient содержит данные об одной поставке на завод какого-либо ингредиента, используемого в производстве. Члены класса приведены в Таблице\_3.

Таблица\_3. Поля класса Ingredient

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Key | Int | Уникальный ключ записи |
| Name | String | Наименование ингредиента |
| Weight | double | Вес закупленной партии |
| DeliveryDate | DateTime | Дата поступления |
| ExpiryDate | DateTime | Дата истечения срока годности |

Класс Program – стандартный класс приложения на C#, содержащий точку входа программы – статическую функцию Main, с которой начинается выполнение приложения. Этот класс инициализирует обобщённые списки, управляет взаимодействием с пользователем и формирует запросы LINQ.

## Коллекции программы

Все коллекции в программе являются статическими членами класса Program и представлены обобщённым списочным контейнером List, параметризируемым соответствующим классом:

* List<Factory> factories – список хлебозаводов;
* List<Product> products – список изделий;
* List<Ingredient> ingredients – список ингредиентов.

# 2 ТЕКСТ ПРОГРАММЫ

Factory.cs

using System;

namespace bakery

{

class Factory

{

public int Key { set; get; } = 0;

public string Name { set; get; }

public DateTime ProdDate { set; get; }

}

}

Product.cs

using System;

namespace bakery

{

class Product

{

public int Key { set; get; } = 0;

public int FactoryKey { set; get; } = 0;

public string Name { set; get; }

public double Weight { set; get; }

public DateTime ExpiryDate { set; get; }

public int Count { set; get; } = 0;

public double Price { set; get; } = 0.0;

}

}

Ingredient.cs

using System;

namespace bakery

{

class Ingredient

{

public int Key { set; get; } = 0;

public string Name { set; get; }

public double Weight { set; get; }

public DateTime DeliveryDate { set; get; }

public DateTime ExpiryDate { set; get; }

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace bakery

{

class Program

{

static List<Factory> factories = new List<Factory>();

static List<Product> products = new List<Product>();

static List<Ingredient> ingredients = new List<Ingredient>();

static string[] paths = new string[]{

"factories",

"products",

"ingredients"

};

static void Main(string[] args)

{

// Загрузить список хлебзаводов

ReadList(0, 2);

// Загрузить список изделий

ReadList(1, 6);

// Загрузить список ингредиентов

ReadList(2, 5);

while (true)

{

char choice = UserChoice();

Console.WriteLine("Вы выбрали действие: " + choice);

if (choice == '0')

{

break;

}

switch(choice)

{

case '1': // Перечень изделий, выпускаемых хлебозаводом

FactoryProductsList();

break;

case '2': // Суммарная стоимость изделий, выпускаемых хлебозаводом

TotalCost();

break;

case '3': // Изделия с просроченными ингредиентами

ExpiredProductsList();

break;

case '4': // Список хлебозаводов в порядке убывания объёма производства

DescendingFactoriesList();

break;

case '5': // Изделие с наибольшим количеством ингредиентов

MaxIngredientsProduct();

break;

case '6': // Изделие с наибольшей прибылью при реализации

MaxRevenueProduct();

break;

case '7': // Добавить изделие, выпускаемое хлебозаводом

AddProduct();

break;

case '8': // Удалить изделие, выпускаемое хлебозаводом

RemoveProduct();

break;

default:

Console.WriteLine("Такого действия нет в списке");

break;

}

Console.Write("Нажмите любую клавишу для продолжения работы ");

Console.ReadKey(true);

}

}

// Вывод пользовательского меню и запрос выбора

static char UserChoice()

{

Console.Clear();

Console.WriteLine("0 - Выход");

Console.WriteLine("1 - Перечень изделий, выпускаемых хлебозаводом");

Console.WriteLine("2 - Суммарная стоимость изделий, выпускаемых хлебозаводом");

Console.WriteLine("3 - Изделия с просроченными ингредиентами");

Console.WriteLine("4 - Список хлебозаводов в порядке убывания объёма производства");

Console.WriteLine("5 - Изделие с наибольшим количеством ингредиентов");

Console.WriteLine("6 - Изделие с наибольшей прибылью при реализации");

Console.WriteLine("7 - Добавить изделие, выпускаемое хлебозаводом");

Console.WriteLine("8 - Удалить изделие, выпускаемое хлебозаводом");

return Console.ReadKey(true).KeyChar;

}

// Запрос у пользователя номера интересующего его хлебозавода

static int RequestFactoryKey()

{

Console.WriteLine("Список заводов:");

Console.WriteLine("№\tНазвание");

foreach (var item in factories)

{

Console.WriteLine("{0}\t{1}", item.Key, item.Name);

}

Console.Write("Введите № завода: ");

return Convert.ToInt32(Console.ReadLine().Trim());

}

// Перечень изделий, выпускаемых хлебозаводом

static int FactoryProductsList(bool choose = false)

{

int id = RequestFactoryKey();

var result = products.Where(p => p.FactoryKey == id);

// аналогичный запрос:

//var result = from p in products where p.FactoryKey == id select p;

Console.WriteLine("№\tВес\tЦена\tВыпуск\tГоден до\t\tНазвание");

foreach (var item in result)

{

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}\t{5}",

p.Key, p.Weight, p.Price, p.Count, p.ExpiryDate, p.Name);

}

if(choose)

{

Console.Write("Введите № партии: ");

return Convert.ToInt32(Console.ReadLine().Trim());

}

return 0;

}

// Суммарная стоимость изделий, выпускаемых хлебозаводом

static void TotalCost()

{

int id = RequestFactoryKey();

var result = products.Where(p => p.FactoryKey == id)

.GroupBy(p => p.FactoryKey)

.Select(g => new {

Id = g.Key,

Total = g.Sum(p => p.Price \* p.Count)

});

/\* тот же запрос через операторы LINQ:

var result = from p in products where p.FactoryKey == id

group p by p.FactoryKey into g

select new {

Id = g.Key,

Total = g.Sum(p => p.Price \* p.Count)

};

\*/

foreach(var group in result)

{

Console.WriteLine("Суммарная стоимость выпущенных изделий: {0} руб.", group.Total);

}

}

// Изделия с просроченными ингредиентами

static void ExpiredProductsList()

{

}

// Список хлебозаводов в порядке убывания объёма производства

static void DescendingFactoriesList()

{

var result = factories.GroupJoin(

products,

f => f.Key,

p => p.FactoryKey,

(fact, prods) => new

{

fact.Key,

fact.Name,

Count = prods.Sum(prod => prod.Count)

}

).OrderByDescending(f => f.Count);

Console.WriteLine("№\tНазвание\tОбъём производства");

foreach (var item in result)

{

Console.WriteLine("{0}\t{1}\t\t{2}", f.Key, f.Name, f.Count);

}

}

// Изделие с наибольшим количеством ингредиентов

static void MaxIngredientsProduct()

{

}

// Изделие с наибольшей прибылью при реализации

static void MaxRevenueProduct()

{

var view = products.Select(p => new {

p.Name,

Revenue = p.Price \* p.Count

});

var maxVal = view.Max(v => v.Revenue);

var result = view.Where(v => v.Revenue == maxVal);

Console.WriteLine("Изделия с наибольшей прибылью:");

foreach (var item in result)

{

Console.WriteLine("{0} ({1} руб).", item.Name, item.Revenue);

}

}

// Добавить изделие, выпускаемое хлебозаводом

static void AddProduct()

{

int factoryKey = RequestFactoryKey();

int key = products.Count + 1;

Console.Write("Введите наименование продукции: ");

string name = Console.ReadLine().Trim();

Console.Write("Введите вес единицы продукции (кг): ");

double weight = Convert.ToDouble(Console.ReadLine().Trim());

Console.Write("Введите дату окончания срока годности (ГГГГ-ММ-ДД): ");

DateTime date = DateTime.Parse(Console.ReadLine().Trim());

Console.Write("Введите объём партии (шт): ");

int count = Convert.ToInt32(Console.ReadLine().Trim());

Console.Write("Введите цену единицы продукции: ");

double price = Convert.ToDouble(Console.ReadLine().Trim());

Product product = new Product();

product.Key = key;

product.FactoryKey = factoryKey;

product.Name = name;

product.Weight = weight;

product.ExpiryDate =date ;

product.Count = count;

product.Price = price;

products.Add(product);

}

// Удалить изделие, выпускаемое хлебозаводом

static void RemoveProduct()

{

int key = FactoryProductsList(true);

var item = products.SingleOrDefault(p => p.Key == key);

if (item != null)

{

products.Remove(item);

}

}

// Загрузка списка из файла

static void ReadList(int index, int fieldCount)

{

if(index >= paths.Length)

return;

StreamReader stream = new StreamReader(paths[index], Encoding.Default);

string line;

int key = 1;

switch(index)

{

case 0:

while((line = stream.ReadLine()) != null)

{

string[] fields = line.Split(';');

if(fields.Length == fieldCount)

{

Factory item = new Factory();

item.Key = key++;

item.Name = fields[0];

item.ProdDate = DateTime.Parse(fields[1]);

factories.Add(item);

}

}

break;

case 1:

while ((line = stream.ReadLine()) != null)

{

string[] fields = line.Split(';');

if (fields.Length == fieldCount)

{

Product item = new Product();

item.Key = key++;

item.FactoryKey = Convert.ToInt32(fields[0]);

item.Name = fields[1];

item.Weight = Convert.ToDouble(fields[2]);

item.ExpiryDate = DateTime.Parse(fields[3]);

item.Count = Convert.ToInt32(fields[4]);

item.Price = Convert.ToDouble(fields[5]);

products.Add(item);

}

}

break;

case 2:

while ((line = stream.ReadLine()) != null)

{

string[] fields = line.Split(';');

if (fields.Length == fieldCount)

{

Ingredient item = new Ingredient();

item.Key = key++;

item.Name = fields[0];

item.Weight = Convert.ToDouble(fields[1]);

item.DeliveryDate = DateTime.Parse(fields[2]);

item.ExpiryDate = DateTime.Parse(fields[3]);

ingredients.Add(item);

}

}

break;

} // switch

Console.WriteLine(paths[index]);

}

}

}

# 3 ФАЙЛЫ ДЛЯ ИНИЦИАЛИЗАЦИИ СПИСКОВ

Сразу после запуска приложения происходит инициализация списков. Для этого используются обычные текстовые файлы, которые можно редактировать с помощью обычного «Блокнота» Windows. Каждой записи – элементу списка – соответствует одна строка. Поля в пределах одной строки отделяются символом точки с запятой – «;». Дробная часть действительных чисел отделена точкой.

Список хлебозаводов содержится в файле factories:

Каравай;2005-05-08

Завод №2;2001-08-20

Завод №3;2003-05-21

Завод №4;2004-03-05

Завод №5;2001-02-22

Завод №6;2015-09-02

Завод №7;2010-04-07

Завод №8;2012-08-08

Завод №9;2008-08-08

Завод №10;2003-11-12

Список хлебобулочных и кондитерских изделий содержится в файле products:

1;Батон нарезной;0.2;2019-06-23;1000;38.0

1;Батон нарезной;0.2;2019-06-25;1500;38.0

2;Рогалик;0.1;2019-06-19;1500;22.5

1;Ржаной хлеб;0.22;2019-06-21;1200;35.2

1;Фермерский хлеб;0.18;2019-06-20;1100;38.0

2;Пирожное;0.12;2019-06-18;1000;55.8

2;Пирожное;0.12;2019-06-24;1100;55.8

1;Булочка;0.15;2019-06-20;1300;40.0

1;Лаваш;0.25;2019-06-20;1000;45.4

3;Фруктовая корзинка;0.2;2019-06-25;1000;73.0

Список ингредиентов, используемых при изготовлении продукции, содержится в файле products:

Вода;100;2019-06-15;2019-07-15;

Вода;200;2019-06-18;2019-07-18;

Вода;300;2019-06-19;2019-07-19;

Мука пшеничная;250;2019-06-01;2019-10-01;

Мука пшеничная;100;2019-03-10;2019-06-10;

Мука пшеничная;100;2019-06-15;2019-09-15;

Отруби;80;2019-06-05;2019-11-05;

Сахар;200;2019-05-20;2020-05-20;

Сахар;100;2019-06-07;2020-06-07;

Масло подсолнечное;20;2019-05-20;2020-08-20;

Масло подсолнечное;100;2019-05-25;2020-08-25;

Масло подсолнечное;100;2019-06-01;2020-09-01;

Масло подсолнечное;50;2019-06-20;2020-09-20;

Соль;10;2019-04-10;2022-04-10;

Соль;50;2019-05-20;2022-05-20;

Дрожжи;50;2019-05-30;2019-06-30;

# ЗАКЛЮЧЕНИЕ

В результате работы была спроектирована и разработана программа, позволяющая анализировать хозяйственную деятельность нескольких хлебозаводов – просматривать выпускаемую продукцию, фильтровать, сортировать и группировать данные в соответствии с различными критериями.

В ходе работы над проектом получены практические навыки разработки программ на языке высокого уровня C#. Изучены особенности создания приложений с помощью этого языка программирования и среды разработки Microsoft Visual Studio. Кроме того, усвоены методы обработки данных с использованием механизма запросов LINQ.

В целом, на базе практического применения, закреплены базовые знания, полученные ранее, при изучении теории.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. – М.: ЛОРИ, 2013. – 320 c.

2. Котов, О.М. Язык C#. Краткое описание и введение в технологии программирования / О.М. Котов. – Екатеринбург: Издательствово Урал. унта, 2014. – 208 с.

3. Чернов, Э.А. Основы программирования на языке C# / Э.А. Чернов. – Дюссельдорф, Германия: LAP Lambert Academic Publishing, 2015. – 310 с.

4. Абрамян, Михаил Visual C# на примерах / Михаил Абрамян. – М.: БХВ-Петербург, 2015. – 572 c.

5. Культин, Н. С# в задачах и примерах / Н. Культин. – М.: БХВ-Петербург, 2016. – 952 c.

6. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. – М.: Наука и техника, 2017. – 496 c