Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИКЕ**

Язык программирования C#

тема

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель от университета | |  |  |  |  |  | Д. А. Евдокимов |
|  | |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ19-17/1б 031939175 | | |  |  |  | А. Д. Непомнящий |
|  | номер группы, зачетной книжки | | |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2020

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc50324895)

[1 Общие сведения о языке С# 4](#_Toc50324896)

[1.1 История создания 4](#_Toc50324897)

[1.2 Область применения 5](#_Toc50324898)

[1.3 Текущее положение и перспективы 6](#_Toc50324899)

[2 Особенности и возможности языка C# 8](#_Toc50324900)

[3 Сравнение C# с некоторыми другими языками программирования 8](#_Toc50324901)

[3.1 Java 8](#_Toc50324902)

[3.2 F# 9](#_Toc50324903)

[3.3 Python 10](#_Toc50324904)

[4 Практическое задание 10](#_Toc50324905)

[4.1 Цель 10](#_Toc50324906)

[4.2 Задачи 10](#_Toc50324907)

[4.3 Описание задания 11](#_Toc50324908)

[4.4 Выполнение 11](#_Toc50324909)

[4.5 Демонстрация работы программы 30](#_Toc50324910)

[Заключение 35](#_Toc50324911)

[Список использованных источников 36](#_Toc50324912)

# ВВЕДЕНИЕ

Язык программирования – формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под её управлением.

На протяжении истории было создано огромное множество языков программирования для различных целей. Областью применения определяется сложность синтаксиса языка, его универсальность, баланс между производительностью и удобством. Какие-то языки получают широкое распространение и надолго остаются в ходу, какие-то становятся неактуальными, некоторые вовсе изначально создаются в качестве юмора или демонстрации границ возможностей разработки языков программирования.

Специалисту в области программной инженерии следует иметь общее представление о наиболее популярных языках программирования  
(JavaScript, Python, Java, C#, C++, C и т. д.), об их концепциях, основных областях применения, достоинствах и недостатках.

Для знакомства в рамках летней практики был выбран язык программирования C#. Были рассмотрены его история создания, назначение, ключевые особенности, перспективы. На выбранном языке была написана программа в соответствии с заданием 7 практической работы по дисциплине «Основы программирования».

# 1 Общие сведения о языке С#

### 1.1 История создания

История C# неразрывно связана с историей платформы .NET корпорации Microsoft, основным языком для работы с которой он является, и для работы с который и был разработан. Разработка платформы началась в 1999 году. Официально о разработке новой технологии было объявлено 13 января  
2000 года, в этот день руководство корпорации объявило о новой стратегии компании, получившей название Next Generation Windows Services (NGWS, «новое поколение служб Windows»).

Компания доминировала на рынке операционных систем и веб-браузеров, обладала множеством наработок в области ПО для Интернета, а также имела долю в компаниях, занимавшихся предоставлением ПО в прокат через Интернет. Кроме того, у корпорации имелось множество различных (и зачастую несовместимых между собой) сред и технологий программирования, поскольку разработка инструментов для программистов была языкоориентированной, то есть для Visual Basic существовал свой набор приложений, а для C++ – свой. Поэтому одной из целей разработки новой платформы было объединение всех наиболее удачных наработок в рамках единой платформы и их унификация. Кроме того, ставилась задача следования всем актуальным тенденциям в области программирования на тот момент. Так, например, новая платформа должна была напрямую поддерживать объектно-ориентированность, безопасность типов, сборку мусора и структурную обработку исключений. Кроме того, корпорации необходимо было предоставить свой ответ набиравшей популярность платформе Java от Sun Microsystems. [1]

К 2000 году компания Microsoft подготовила промышленные версии новых компонентных технологий и решений в области обмена сообщениями и данными, а также создания Интернет-приложений. В поддержку этих новшеств Microsoft выпустила инструментарий для разработки приложений – платформу .NET.

В ходе работы над ней, в 1999 Андерс Хейлсберг организовал команду для работы над новым языком Cool (C-like Object Oriented Language). К релизу .NET язык был переименован в C#, а ASP.NET полностью написана на нём. [2]

Стоит отметить, что .NET долгое время развивался преимущественно как платформа для Windows под названием .NET Framework. В 2019 вышла последняя версия этой платформы – .NET Framework 4.8. Она больше не развивается.

С 2014 Microsoft стал развивать альтернативную платформу – .NET Core, которая уже предназначалась для разных платформ и должна была вобрать в себя все возможности устаревшего .NET Framework и добавить новую функциональность. [3]

### **1.2 Область применения**

C# имеет достаточно широкую область применения.

C# – исключительное средство для разработки приложений для универсальной платформы Windows, будь то десктопные приложения, приложения для Windows mobile, Xbox или очков дополненной реальности HoloLens.

C# хорошо пригоден для веб-разработки, .NET Core совместима с Linux и MacOS.

Платформа Xamarin позволяет создавать мобильные приложения для iOS, Android и Windows Mobile, хотя здесь обычно отдаётся предпочтение JavaScript.

Кроссплатформенный фреймворк ML.NET позволяет применять C# в машинном обучении.

C# является преимущественным языком для работы с игровым движком Unity. [4]

### 1.3 Текущее положение и перспективы

На август 2020, согласно данным индекса TIOBE, C# занимает 5 место по популярности среди всех языков программирования. С 2015 года резких изменений в его популярности нет. [5] Согласно опросу разработчиков за 2020 год на Stack Overflow, C# занимает 7 место по популярности. [6] Статистика представлена на рисунках 1 и 2.

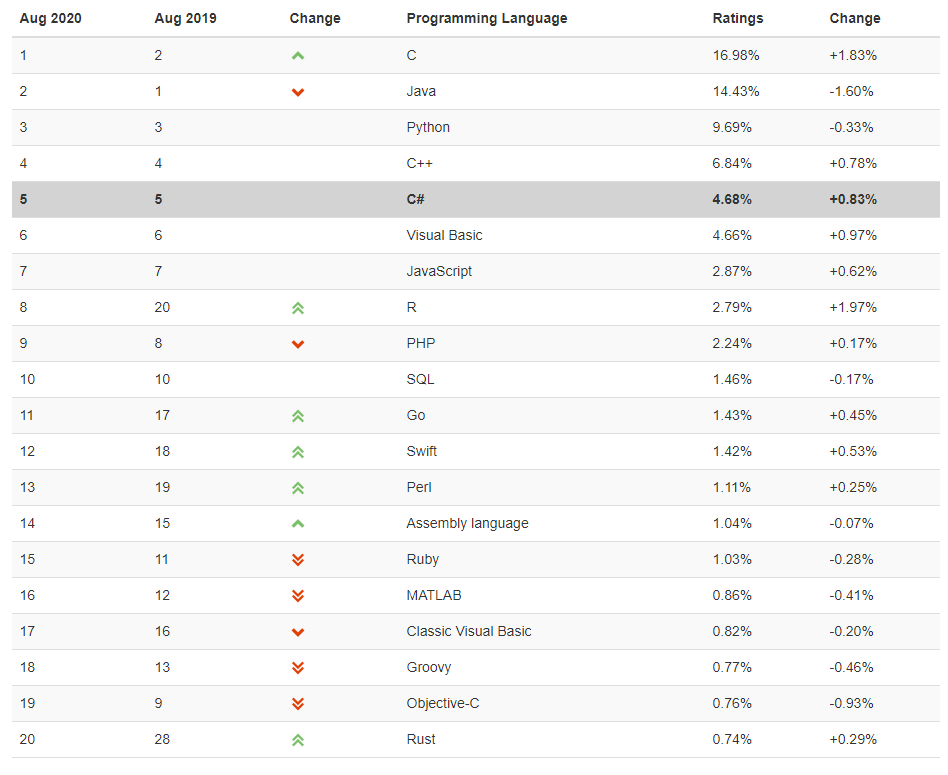


Рисунок 1 – Популярность языков программирования на август 2020 года согласно индексу TIOBE

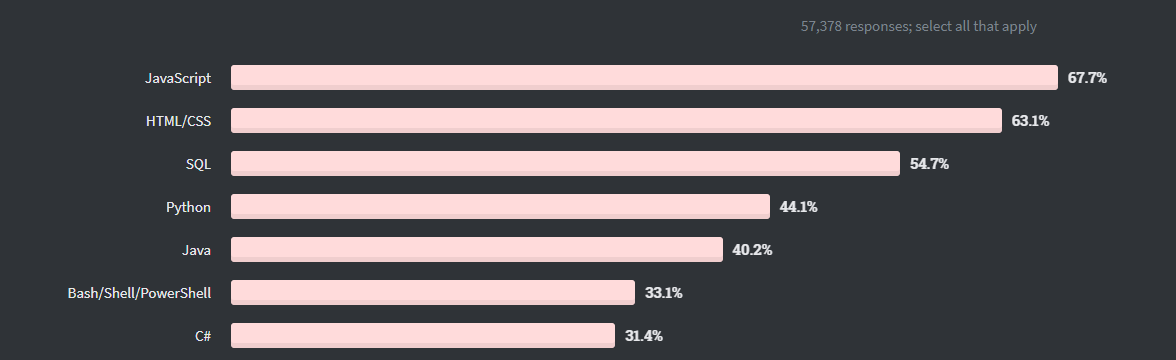


Рисунок 2 – Популярность языков программирования согласно опросу за 2020 год на Stack Overflow

6 мая было объявлено, что следующим после .NET Core 3.0 релизом будет .NET 5. Это будет следующий большой релиз в семействе .NET.

В будущем останется только один .NET, и его можно будет использовать для разработки под Windows, Linux, macOS, iOS, Android, tvOS, watchOS, WebAssembly и другие платформы.

.NET 5 – следующий шаг в .NET Core. Проект призван улучшить .NET в нескольких ключевых аспектах – создать единые исполняющую среду и фреймворк, которые можно использовать везде, с одинаковым поведением в runtime и опытом разработки, расширить возможности .NET за счёт лучших наработок из .NET Core, .NET Framework, Xamarin и Mono 3 и собрать продукт из единой кодовой базы, над которой разработчики (из Microsoft и сообщества) могут вместе работать и расширять её, что позволит улучшить все возможные сценарии. [7]

# 2 Особенности и возможности языка C#

C# – это объектно- и компонентно-ориентированный язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML, память автоматически освобождается сборщиком мусора. Программы, написанные на C# могут пользоваться классами и методами Base Class Library – стандартной библиотекой классов платформы .NET  
Framework. [8]

C# - один из языков высокого уровня платформы .NET. Компилятор транслирует код на C# в CIL. Далее при запуске на выполнение будет происходить JIT-компиляция в машинный код. [3]

# 3 Сравнение C# с некоторыми другими языками программирования

### 3.1 Java

Языки C# и Java появились в разное время. Язык Java был создан задолго до появления C#. Таким образом, если Java создавался, опираясь в большей степени на опыт языков Objective C и C, то для C# такой опорой являлись C++ и сам Java. И, несмотря на своё название, C# оказался ближе к Java, чем к C++. С точки зрения разработчика языки Java и C# очень похожи. Оба языка являются строго типизированными, объектными. Оба вобрали в себя многое из синтаксиса C++, но в отличие от C++, проще в освоении для начинающих. Оба позаимствовали из C набор основных ключевых слов и служебных символов, в том числе фигурные скобки для выделения блоков. Оба языка опираются на сборку мусора. Оба языка сопровождаются богатыми коллекциями библиотек. Но есть в языках также свои особенности и различия, сильные и слабые стороны. C# учёл многие недостатки Java, и исправил их в своей реализации. Но и Java не стоит на месте, развиваясь параллельно с C#.

Сравнение C# и Java становится объектом серьезных трудов, а выбор лучшего из двух языков – объектом многочисленных споров на разных форумах.

Среди преимуществ C# против Java, например, отмечают то, что Java несколько уступает обобщенными типами, отсутствием value-типов, меньшим количеством примитивных типов. C# обеспечивает более широкую поддержку событийно-ориентированного программирования. В C# включена перегрузка операторов. [9] Однако перечисленные преимущества C# вполне можно не считать таковыми, апеллируя выгодой подобного упрощения языка. Основные преимущества Java скорее не в синтаксисе, а в широчайшей распространенности языка и значительно большим количеством поддерживающих его устройств.

Так же существует позиция, что Java и C# два языка с разными задачами, решающий каждый свою задачу лучше друг друга. [10]

### 3.2 F#

Так же, как и C#, F# является языком высокого уровня платформы .NET.  
F# является функциональным языком программирования, но также поддерживает процедурное и объектно-ориентированное программирование. [11]. Из того, что F# задуман как функциональный язык, вытекают его преимущества и недостатки перед C#.

В частности выделяют: лаконичность синтаксиса, большую безопасность в отношении типов, относительную легкость тестирования и отладки, широкие возможности распараллеливания вычислений. [12]

F# нацелен на упрощение работы с Big Data и многопоточность, то есть востребован в науке, в создании искусственного интеллекта.

### 3.3 Python

Python – интерпретируемый высокоуровневый язык программирования общего назначения, поддерживает структурное, функциональное, обобщенное и объектно-ориентированное программирование. Дизайн языка выстроен вокруг объектно-ориентированной модели. Python работает почти на всех известных платформах.

Основные отличия Python от C# – значительно более простой синтаксис и динамическая типизация, интерпретируемость, большая кроссплатформенность, меньшее быстродействие, разделяют их сферы применения. Наиболее широко Python применяется как сценарный язык, в научных расчётах, в машинном обучении, также широкое применение находит в веб-разработке, в системном администрировании. [13].

# 4 Практическое задание

### 4.1 Цель

Ознакомиться с синтаксисом языка C#, особенностями создания программ и общепринятыми стандартами написания кода на этом языке.

### 4.2 Задачи

Сущности А и Б имеют отношение многие ко многим. Написать программу, моделирующую это отношение.

На оценку 3 балла:

1. написать программу в соответствии с заданием;
2. реализовать возможность добавления элементов А и Б;
3. реализовать возможность создания связи между элементами  
   типа А и Б.

На оценку 4 балла:

1. выполнить требования предыдущих пунктов;
2. добавить возможность сохранять и загружать данные из  
   файла (в т.ч. связи);
3. добавить возможность выводить на экран все элементы А или Б (на усмотрение пользователя);
4. добавить возможность выводить на экран все элементы А, связанные с выбранным элементом Б и наоборот.

На оценку 5 баллов:

1. выполнить требования предыдущих пунктов;
2. добавить возможность удаления выбранного элемента  
   типа А или типа Б;
3. добавить сортировку по одному из выбранных полей.

### 4.3 Описание задания

Товары и заказы.

А: товар (наименование, цена, масса).

Б: заказ (ФИО заказчика, срок доставки, дата оформления).

Один товар может входить в несколько заказов, один заказ может содержать несколько товаров.

### 4.4 Выполнение

Решение SummerPractice1 для выполнения задачи состоит из библиотек классов SummerPractice1.Core, включающей файлы OperationCode.cs, Order.sc, OrderSave.sc, Product.cs, SaveData.cs, и SummerPractice1.Main, включающей Program.cs с методом Main. На рисунке ниже представлена структура решения.

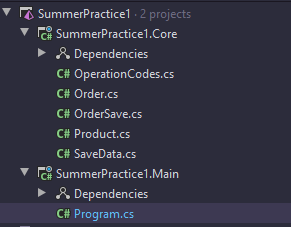


Рисунок 3 – Структура решения

Листинг 1 – Код в файле OperationCodes.cs

namespace SummerPractice1.Core

{

public enum OperationCodes

{

None = -1,

LoadFromFile = 1,

SaveToFile = 2,

Add = 3,

Print = 4,

Delete = 5,

SortOrders = 6,

SortProducts = 7,

Quit = 8

}

public enum AddSuboperationCodes

{

AddOrder = 1,

AddProduct = 2,

AddRelation,

AddBack

}

public enum PrintSuboperationCodes

{

PrintOrders = 1,

PrintProducts = 2,

PrintByOrder,

PrintByProduct,

Окончание листинга 1

PrintBack

}

public enum DeleteSuboperationCodes

{

DeleteOrder = 1,

DeleteProduct = 2,

DeleteBack

}

public enum SortOrdersSuboperationCodes

{

SortOrdersOrd = 1,

SortOrdersShip = 2,

SortOrdersBack = 3

}

public enum SortProductsSuboperationCodes

{

SortProductsPrice = 1,

SortProductsWeight = 2,

SortProductsBack = 3

}

}

Листинг 2 – Код в файле Order.cs

using System;

using System.Collections.Generic;

namespace SummerPractice1.Core

{

public class Order

{

public List<Product> Content { get; } = new List<Product>();

public DateTime OrderDate { get; set; }

public string Owner { get; set; }

public DateTime ShipmentDate { get; set; }

public Order(string owner, DateTime orderDate, DateTime shipmentDate)

{

Owner = owner;

OrderDate = orderDate;

ShipmentDate = shipmentDate;

}

public Order(OrderSave origin, List<Product> products)

{

Owner = origin.Owner;

OrderDate = origin.OrderDate;

ShipmentDate = origin.ShipmentDate;

foreach (var i in origin.Products)

{

Content.Add(products.Find(u => u.Name == i));

}

}

public override string ToString() =>

$"Owner: {Owner}\nDate of order: {OrderDate}\nDate of shipment: {ShipmentDate}\n";

public List<Product> AllProducts() => Content;

}

}

Листинг 3 – Код в файле OrderSave.cs

using System;

using System.Collections.Generic;

namespace SummerPractice1.Core

{

public class OrderSave

{

public DateTime OrderDate { get; set; }

public string Owner { get; set; }

public DateTime ShipmentDate { get; set; }

public List<string> Products { get; } = new List<string>();

public OrderSave()

{

}

public OrderSave(Order origin)

{

Owner = origin.Owner;

OrderDate = origin.OrderDate;

ShipmentDate = origin.ShipmentDate;

foreach (var i in origin.Content)

{

Products.Add(i.Name);

}

}

}

}

Листинг 4 – Код в файле Product.cs

using System.Collections.Generic;

using System.Linq;

namespace SummerPractice1.Core

{

public class Product

{

public Product(string name, int price, int weight)

{

Name = name;

Окончание листинга 4

Price = price;

Weight = weight;

}

public string Name { get; set; }

public int Price { get; set; }

public int Weight { get; set; }

public override string ToString() => $"Product: {Name}\nPrice: {Price}\nWeight: {Weight}\n";

public IEnumerable<Order> AllOrders(IEnumerable<Order> objList) =>

objList.Where(order => order.Content.Contains(this));

}

}

Листинг 5 – Код в файле SaveData.cs

using System.Collections.Generic;

namespace SummerPractice1.Core

{

public class SaveData

{

public List<Product> Products { get; }

public List<OrderSave> Orders { get; }

public SaveData(List<Product> products, List<OrderSave> orders)

{

Products = products;

Orders = orders;

}

}

}

Листинг 6 – Код в файле Program.cs

using System;

using System.Collections.Generic;

using System.IO;

using Newtonsoft.Json;

using SummerPractice1.Core;

namespace SummerPractice1.Main

Продолжение листинга 6

{

internal static class Program

{

private static int CustomIntInput(Func<int, bool> inputCheck)

{

while (true)

{

if (Int32.TryParse(Console.ReadLine(), out var result))

{

if (inputCheck(result))

{

return result;

}

}

Console.WriteLine("Wrong format!");

}

}

private static DateTime CustomDateInput()

{

while (true)

{

try

{

var result = DateTime.Parse(Console.ReadLine());

return result;

}

catch (FormatException)

{

Console.WriteLine("Wrong format!");

}

}

}

private static Product ProductByName(List<Product> products, string name)

{

return products.Find(u => u.Name == name);

}

Продолжение листинга 6

private static void AddProduct(List<Product> products, string name, int price, int weight)

{

if (ProductByName(products, name) != null)

{

Console.WriteLine("Such product already exists!");

return;

}

products.Add(new Product(name, price, weight));

Console.WriteLine("Added!");

}

private static void DeleteProduct(List<Product> products, List<Order> orders, string name)

{

if (products.RemoveAll(u => u.Name == name) == 0)

{

Console.WriteLine("No such product");

return;

}

foreach (var i in orders)

{

i.Content.RemoveAll(u => u.Name == name);

}

Console.WriteLine("Deleted!");

}

private static string AllProductsToString(List<Product> products)

{

var result = "";

foreach (var i in products)

{

result += i.ToString();

}

return result;

}

Продолжение листинга 6

private static Order OrderByOwner(List<Order> orders, string owner)

{

return orders.Find(u => u.Owner == owner);

}

private static void AddOrder(List<Order> orders, string owner, DateTime orderDate, DateTime shipmentDate)

{

if (OrderByOwner(orders, owner) != null)

{

Console.WriteLine("Order with such owner already exists!");

return;

}

orders.Add(new Order(owner, orderDate, shipmentDate));

Console.WriteLine("Added!");

}

private static void DeleteOrder(List<Order> orders, string owner)

{

orders.RemoveAll(u => u.Owner == owner);

Console.WriteLine("Deleted!");

}

private static string AllOrdersToString(List<Order> orders)

{

var result = "";

foreach (var i in orders)

{

result += i.ToString();

}

return result;

}

private static void AddRelation(List<Product> products, List<Order> orders, string productName,

string orderOwner)

{

if (ProductByName(products, productName) == null)

{

Console.WriteLine("No such product!");

Продолжение листинга 6

return;

}

if (OrderByOwner(orders, orderOwner) == null)

{

Console.WriteLine("No such order!");

return;

}

OrderByOwner(orders, orderOwner).Content.Add(ProductByName(products, productName));

Console.WriteLine("Added!");

}

private static string OrderContentToString(Order order, List<Product> products)

{

var result = "";

if (order.Content.Count == 0)

{

result = "No products in order!";

}

foreach (var i in products)

{

if (order.Content.Exists(u => u == i))

{

result += i + "\n";

}

}

return result;

}

private static string ProductReferencesToString(Product product, List<Order> orders)

{

var result = "";

foreach (var i in orders)

{

if (i.Content.Contains(product))

Продолжение листинга 6

{

result += i + "\n";

}

}

if (result == "")

{

result = "No references to product";

}

return result;

}

private static bool MainMenuInputChecker(int intToCheck) =>

intToCheck >= (int) OperationCodes.LoadFromFile

&& intToCheck <= (int) OperationCodes.Quit;

private static bool AddMenuInputChecker(int intToCheck) =>

intToCheck >= (int) AddSuboperationCodes.AddOrder

&& intToCheck <= (int) AddSuboperationCodes.AddBack;

private static bool PrintMenuInputChecker(int intToCheck) =>

intToCheck >= (int) PrintSuboperationCodes.PrintOrders

&& intToCheck <= (int) PrintSuboperationCodes.PrintBack;

private static bool DeleteMenuInputChecker(int intToCheck) =>

intToCheck >= (int) DeleteSuboperationCodes.DeleteOrder

&& intToCheck <= (int) DeleteSuboperationCodes.DeleteBack;

private static bool SortOrdersMenuInputChecker(int intToCheck) =>

intToCheck >= (int) SortOrdersSuboperationCodes.SortOrdersOrd

&& intToCheck <= (int) SortOrdersSuboperationCodes.SortOrdersBack;

private static bool SortProductsMenuInputChecker(int intToCheck) =>

intToCheck >= (int) SortProductsSuboperationCodes.SortProductsPrice

&& intToCheck <= (int) SortProductsSuboperationCodes.SortProductsBack;

private static bool PositiveIntInputChecker(int intToCheck) => intToCheck >= 0;

private static void Main(string[] args)

Продолжение листинга 6

{

var operationCode = OperationCodes.None;

var listOfProducts = new List<Product>();

var listOfOrders = new List<Order>();

while (operationCode != OperationCodes.Quit)

{

Console.WriteLine();

Console.WriteLine("1. Load data from savefile.");

Console.WriteLine("2. Save current data to savefile.");

Console.WriteLine("3. Add new element.");

Console.WriteLine("4. Print information.");

Console.WriteLine("5. Delete elements.");

Console.WriteLine("6. Sort orders.");

Console.WriteLine("7. Sort products.");

Console.WriteLine("8. Quit without saving.");

Console.WriteLine();

operationCode = (OperationCodes) CustomIntInput(MainMenuInputChecker);

switch (operationCode)

{

case OperationCodes.Add:

{

Console.WriteLine();

Console.WriteLine("1. Add order.");

Console.WriteLine("2. Add product.");

Console.WriteLine("3. Add relation.");

Console.WriteLine("4. Back.");

Console.WriteLine();

var suboperationCode = (AddSuboperationCodes) CustomIntInput(AddMenuInputChecker);

switch (suboperationCode)

{

case AddSuboperationCodes.AddProduct:

{

Console.WriteLine("Enter product's name");

var name = Console.ReadLine();

Console.WriteLine("Enter product's price");

Продолжение листинга 6

var price = CustomIntInput(PositiveIntInputChecker);

Console.WriteLine("Enter product's weight");

var weight = CustomIntInput(PositiveIntInputChecker);

AddProduct(listOfProducts, name, price, weight);

break;

}

case AddSuboperationCodes.AddOrder:

{

Console.WriteLine("Enter owner's name");

var owner = Console.ReadLine();

Console.WriteLine("Enter date of order");

var dateOfOrder = CustomDateInput();

Console.WriteLine("Enter date of shipment");

var dateOfShipment = CustomDateInput();

AddOrder(listOfOrders, owner, dateOfOrder, dateOfShipment);

break;

}

case AddSuboperationCodes.AddRelation:

{

Console.WriteLine("Enter product's name");

var name = Console.ReadLine();

Console.WriteLine("Enter owner's name");

var owner = Console.ReadLine();

AddRelation(listOfProducts, listOfOrders, name, owner);

break;

}

case AddSuboperationCodes.AddBack:

break;

default:

throw new ArgumentOutOfRangeException();

}

break;

}

case OperationCodes.Print:

{

Console.WriteLine();

Console.WriteLine("1. Print all orders.");

Console.WriteLine("2. Print all products.");

Продолжение листинга 6

Console.WriteLine("3. Print order's content.");

Console.WriteLine("4. Print product's related orders.");

Console.WriteLine("5. Back.");

Console.WriteLine();

var suboperationCode = (PrintSuboperationCodes) CustomIntInput(PrintMenuInputChecker);

switch (suboperationCode)

{

case PrintSuboperationCodes.PrintProducts:

{

Console.WriteLine(AllProductsToString(listOfProducts));

break;

}

case PrintSuboperationCodes.PrintOrders:

{

Console.WriteLine(AllOrdersToString(listOfOrders));

break;

}

case PrintSuboperationCodes.PrintByOrder:

{

Console.WriteLine("Enter owner's name");

var owner = Console.ReadLine();

Console.WriteLine(OrderContentToString(OrderByOwner(listOfOrders, owner),

listOfProducts));

break;

}

case PrintSuboperationCodes.PrintByProduct:

{

Console.WriteLine("Enter product's name");

var name = Console.ReadLine();

Console.WriteLine(ProductReferencesToString(ProductByName(listOfProducts, name),

listOfOrders));

break;

}

case PrintSuboperationCodes.PrintBack:

break;

default:

Продолжение листинга 6

throw new ArgumentOutOfRangeException();

}

break;

}

case OperationCodes.Delete:

{

Console.WriteLine();

Console.WriteLine("1. Delete order");

Console.WriteLine("2. Delete product.");

Console.WriteLine("3. Back");

Console.WriteLine();

var suboperationCode = (DeleteSuboperationCodes) CustomIntInput(DeleteMenuInputChecker);

switch (suboperationCode)

{

case DeleteSuboperationCodes.DeleteOrder:

{

Console.WriteLine("Enter owner's name");

var owner = Console.ReadLine();

DeleteOrder(listOfOrders, owner);

break;

}

case DeleteSuboperationCodes.DeleteProduct:

Console.WriteLine("Enter product's name");

var name = Console.ReadLine();

DeleteProduct(listOfProducts, listOfOrders, name);

break;

case DeleteSuboperationCodes.DeleteBack:

break;

default:

throw new ArgumentOutOfRangeException();

}

break;

}

case OperationCodes.LoadFromFile:

{

listOfProducts.Clear();

Продолжение листинга 6

listOfOrders.Clear();

Console.WriteLine("Enter file name");

var filename = Console.ReadLine();

try

{

using var sr = new StreamReader(filename);

var loaded = JsonConvert.DeserializeObject<SaveData>(sr.ReadToEnd());

listOfProducts = loaded.Products;

foreach (var i in loaded.Orders)

{

listOfOrders.Add(new Order(i, listOfProducts));

}

Console.WriteLine("Loaded!");

}

catch (Exception)

{

Console.WriteLine("Error occured!");

}

break;

}

case OperationCodes.SaveToFile:

{

Console.WriteLine("Enter file name");

var filename = Console.ReadLine();

try

{

using var sr = new StreamWriter(filename);

List<OrderSave> ordersSaveData = new List<OrderSave>();

foreach (var i in listOfOrders)

{

ordersSaveData.Add(new OrderSave(i));

}

var json = JsonConvert.SerializeObject(new SaveData(listOfProducts, ordersSaveData));

sr.WriteLine(json);

Console.WriteLine("Saved!");

}

Продолжение листинга 6

catch (Exception)

{

Console.WriteLine("Error occured!");

}

break;

}

case OperationCodes.SortOrders:

{

Console.WriteLine();

Console.WriteLine("1. Sort orders by order date");

Console.WriteLine("2. Sort orders by shipment date");

Console.WriteLine("3. Back");

Console.WriteLine();

var suboperationCode = (SortOrdersSuboperationCodes) CustomIntInput(SortOrdersMenuInputChecker);

switch (suboperationCode)

{

case SortOrdersSuboperationCodes.SortOrdersOrd:

{

listOfOrders.Sort((a, b) => DateTime.Compare(a.OrderDate, b.OrderDate));

Console.WriteLine("Sorted!");

break;

}

case SortOrdersSuboperationCodes.SortOrdersShip:

{

listOfOrders.Sort((a, b) => DateTime.Compare(a.ShipmentDate, b.ShipmentDate));

Console.WriteLine("Sorted!");

break;

}

case SortOrdersSuboperationCodes.SortOrdersBack:

{

break;

}

default:

throw new ArgumentOutOfRangeException();

}

break;

Продолжение листинга 6

}

case OperationCodes.SortProducts:

{

Console.WriteLine();

Console.WriteLine("1. Sort products by price");

Console.WriteLine("2. Sort products by weight");

Console.WriteLine("3. Back");

Console.WriteLine();

var suboperationCode =

(SortProductsSuboperationCodes) CustomIntInput(SortProductsMenuInputChecker);

switch (suboperationCode)

{

case SortProductsSuboperationCodes.SortProductsPrice:

{

listOfProducts.Sort((a, b) => b.Price - a.Price);

Console.WriteLine("Sorted!");

break;

}

case SortProductsSuboperationCodes.SortProductsWeight:

{

listOfProducts.Sort((a, b) => b.Weight - a.Weight);

Console.WriteLine("Sorted!");

break;

}

case SortProductsSuboperationCodes.SortProductsBack:

{

break;

}

default:

throw new ArgumentOutOfRangeException();

}

break;

}

case OperationCodes.Quit:

break;

case OperationCodes.None:

break;

Окончание листинга 6

default:

throw new ArgumentOutOfRangeException();

}

}

}

}

}

### 4.5 Демонстрация работы программы

Ниже на рисунках приведены результаты выполнения некоторых операций в программе.

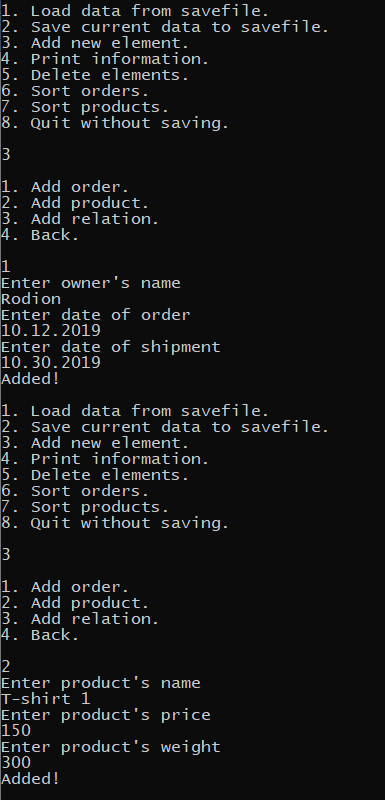


Рисунок 4 – Создание заказа и одного продукта

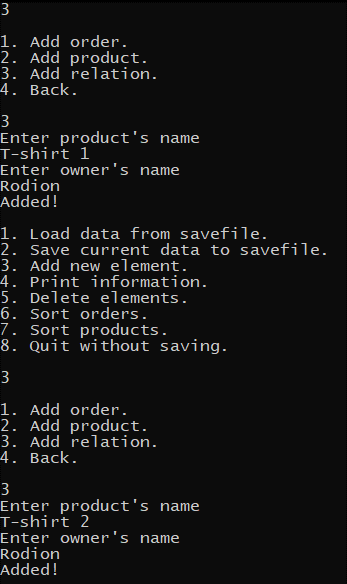


Рисунок 5 – Создание двух связей заказа с продуктами

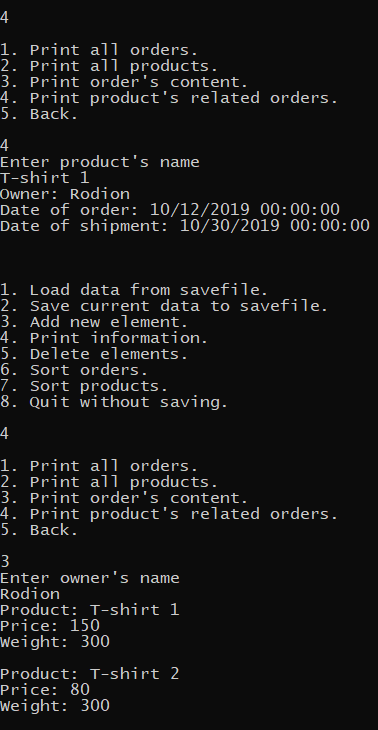


Рисунок 6 – Вывод связанных с продуктом заказов и всех продуктов в заказе

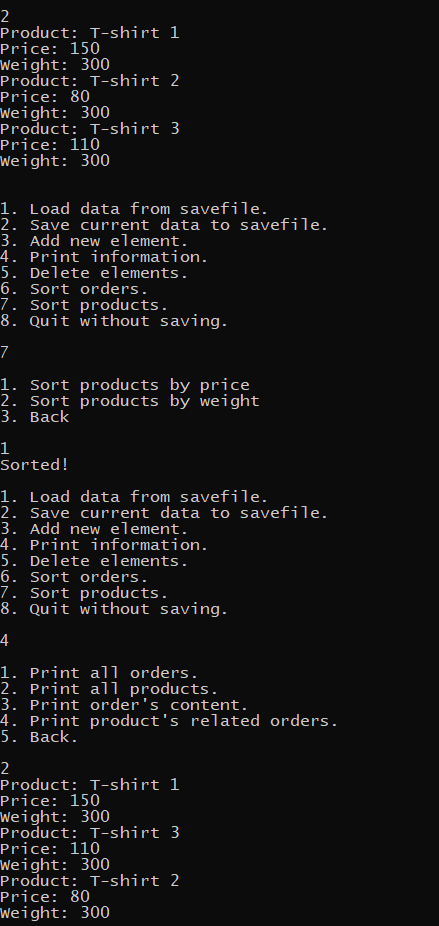


Рисунок 7 – Сортировка по выбранному полю

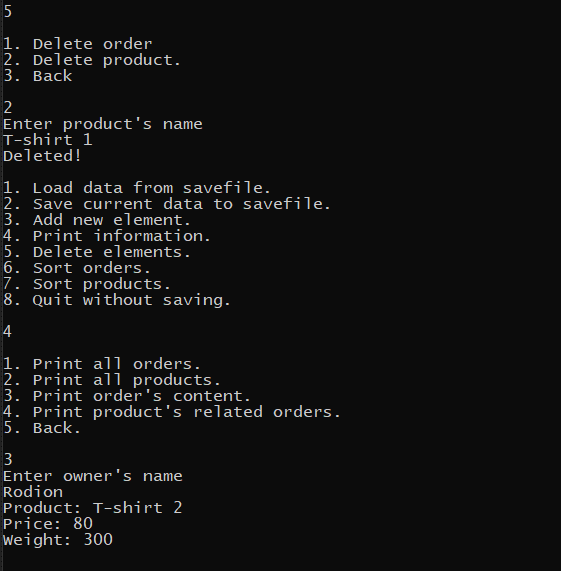


Рисунок 8 – Удаление продукта и проверка содержавшего его заказа

# ЗАКЛЮЧЕНИЕ

Язык программирования C# на август 2020 года является распространенным, поддерживаемым и развиваемым объектно-ориентированным языком программирования. C# является наиболее подходящим языком для создания приложений для универсальной платформы Windows, но так же может быть использован в разработке для многих других широко распространенных платформ и находит применение в различных сферах, таких как веб-разработка, мобильные приложения, машинное обучение, разработка видеоигр.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Redmond maps plans for life after the pc / Cnet [Электронный ресурс] – Режим доступа: [https://www.cnet.com/news/redmond-maps-plans-for-life-after-the-pc](https://www.cnet.com/news/redmond-maps-plans-for-life-after-the-pc/) (Дата обращения: 31.08.2020).
2. История языков программирования: C# – впереди планеты всей / Хабр [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/313694/> (Дата обращения: 31.08.2020).
3. C# и .NET введение / Metanit [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/tutorial/1.1.php> (Дата обращения: 31.08.2020).
4. Почему C# программисты скоро будут нарасхват / Proglib [Электронный ресурс] – Режим доступа: <https://proglib.io/p/c-sharp-popularity> (Дата обращения: 31.08.2020).
5. Index / TIOBE – The software quality company [Электронный ресурс] – Режим доступа: [https://www.tiobe.com/tiobe-index](https://www.tiobe.com/tiobe-index/) (Дата обращения: 31.08.2020).
6. Stack Overflow Developer Survey 2020 / Stack Overflow [Электронный ресурс] – Режим доступа: <https://insights.stackoverflow.com/survey/2020> (Дата обращения: 31.08.2020).
7. Introducing .NET 5 / Microsoft Developer Blog [Электронный ресурс] – Режим доступа: <https://devblogs.microsoft.com/dotnet/introducing-net-5> (Дата обращения: 31.08.2020).
8. Обзор языка C# / Microsoft Docs [Электронный ресурс] – Режим доступа:  [[https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp](https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/)](https://devblogs.microsoft.com/dotnet/introducing-net-5) (Дата обращения: 31.08.2020).
9. C# and Java: Compering Programming Languages / Microsoft Docs [Электронный ресурс] – Режим доступа: [<https://docs.microsoft.com/en-us/previous-versions/windows/embedded/ms836794(v=msdn.10)>](https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/) (Дата обращения: 31.08.2020).
10. Крадущийся тигр, затаившийся дракон / Хабр [Электронный ресурс] – Режим доступа:  [<https://habr.com/ru/post/145932/>](https://habr.com/ru/post/313694/) (Дата обращения: 31.08.2020).
11. Что такое F# / Microsoft Docs [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/fsharp/what-is-fsharp> (Дата обращения: 31.08.2020).
12. F# меня испортил, или почему я больше не хочу писать на C# / Хабр [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/428930/> (Дата обращения: 31.08.2020).
13. C# vs Python - detailed comparison / Boldare [Электронный ресурс] – Режим доступа: <https://www.boldare.com/blog/python-vs-c-sharp-a-comparison/> (Дата обращения: 31.08.2020).
14. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной  
    деятельности. – Введ. 09.01.2014. – Красноярск: ИПК СФУ, 2014. – 60 с.