Учреждение образования Федерации профсоюзов Беларуси

«Международный университет «МИТСО»

Рег. № \_\_\_\_\_\_\_\_\_\_ Кафедра информационных систем и технологий

Дата \_\_\_\_\_\_\_\_\_2020

**КУРСОВАЯ РАБОТА**

на тему Учет расхода электроэнергии

по дисциплине Основы конструирования программ

|  |  |
| --- | --- |
| Основные замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Отметка о допуске курсовой работы к  защите:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_  Подпись научного руководителя:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Студент: Илья Русенко Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  (Ф.И.О. – полностью)  Курс \_\_1\_\_, группа \_\_\_2021\_\_\_\_  Факультет \_\_экономический\_\_\_  Специальность \_\_Информационные системы и технологии\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Научный руководитель:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность, ученая степень, ученое звание)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф.И.О. – полностью) |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc41387618)

[1.ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc41387619)

[2.ТРЕБОВАНИЕ К ПРОГРАММЕ 5](#_Toc41387620)

[3.КОНСТРУИРОВАНИЕ ПРОГРАММЫ 6](#_Toc41387621)

[4.ОПИСАНИЕ ИНТЕРФЕЙСА 9](#_Toc41387622)

[5.ВЫБОР СПОСОБА ОРГАНИЗАЦИИ ДАННЫХ 11](#_Toc41387623)

[6.РАЗРАБОТКА ПЕРЕЧНЯ ПОЛЬЗОВАТЕЛЬСКИХ МЕТОДОВ ПРОГРАММЫ 12](#_Toc41387624)

[7.ПРОГРАММНАЯ ЧАСТЬ 13](#_Toc41387625)

[ЗАКЛЮЧЕНИЕ 15](#_Toc41387626)

[ПРИЛОЖЕНИЕ 16](#_Toc41387627)

# **ВВЕДЕНИЕ**

Была поставлена задача разработать программный модуль «Учет расхода электроэнергии», предназначенный для использования работниками службы электроснабжения. В базе содержатся сведения о клиентах службы (Имя, фамилия, возраст, количество используемой энергии, денежный баланс на счету ). При возникновения нового клиента его данные заноситься в базу.

Программа создана для систематизации клиентов службы электроснабжения, быстрого просмотра/изменения и добавления новых клиентов службы.

Цель разработки обеспечить работы службы электроснабжения быстро и максимально эффективно, удовлетворять запросы службы, а также должен быть возможность пополнять базу, для оптимизации процессов службы электроснабжения. И неотъемлемой частью данной программы должна высокая оптимизация, чтобы пользователь мог запустить даже на самом слабом устройстве.

Задачи данного программного продукта: быстро и гибко редактироваться со стороны владельцев службы электроснабжения. С точки зрения использования пользователя программа должна иметь красивый, понятный и интуитивный интерфейс чтобы было владельцу приятно пользоваться программой. Так же требуется предусмотреть возможность добавления новых функций в программе.

Для разработки программы был использован текстовый файл и язык программирования C# разработанный в среде Visual Studio компаний Microsoft.

Visual Studio-линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой Windows Forms, а также веб-сайты, веб-приложения, веб-службы, как в родном, так и в управляемом кодах для всех платформ, поддерживающих Windows, Windows Mobile, Windows Ce, .Net Framework, Xbox, Windows Phone .Net Compact Framework и Silverling.

# **1.ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Была поставлена задача разработать программный модуль «Учет расхода электроэнергии», предназначенный для использования работниками службы электроснабжения. В базе содержатся сведения о клиентах службы (Имя, фамилия, возраст, количество используемой энергии, денежный баланс на счету). При возникновения нового клиента его данные заноситься в базу.

За последние десятилетия увеличилось как количество городов, как и их население. Из-за этого возникают сложности в снабжении большого количество населения, поэтому разрабатываемая программа должна выдержать весьма серьёзную конкуренцию.

Функции программного продукта: содержатся сведения о клиентах службы электроснабжения (Имя, фамилия, возраст, количество используемой энергии, денежный баланс на счету) должны иметь понятные данные и возможность обрабатывать их (добавлять/удалять, просматривать, изменять).

Данный программный продукт выгодно отличается от других подобных тем что, пользуясь им, пользователь может не только просматривать клиентов, но и производить различные взаимодействия с самими клиентами.

# **2.ТРЕБОВАНИЕ К ПРОГРАММЕ**

Исходя из задач, описанных выше главными требованиями к языку, на котором будет написана программы — это тонкая настройка и возможность использованием минимум вычислительных ресурсов добиваться максимум результата.

Для выполнения данных задач очень подходит такой язык программирования как с#. Так как его из-за того, что он является весьма низкоуровневым языком программирования, а значит он больше других приближен к машинному коду, из-за этого с# дает возможность оптимизировать проект так что даже на очень старых устройствах он будет работать.

Среда разработки Microsoft Visual Studio. Вид приложения – консольное. Программа построена на базе объектно-ориентированного программирования. Способ организации данных поля классов. Способ хранения данных – текстовый фалй/xml. Каждая логическая завершенная данных подзадача программы реализована в виде методов. К защите курсовой предоставляется: консольное приложение и пояснительная записка

# **3.КОНСТРУИРОВАНИЕ ПРОГРАММЫ**

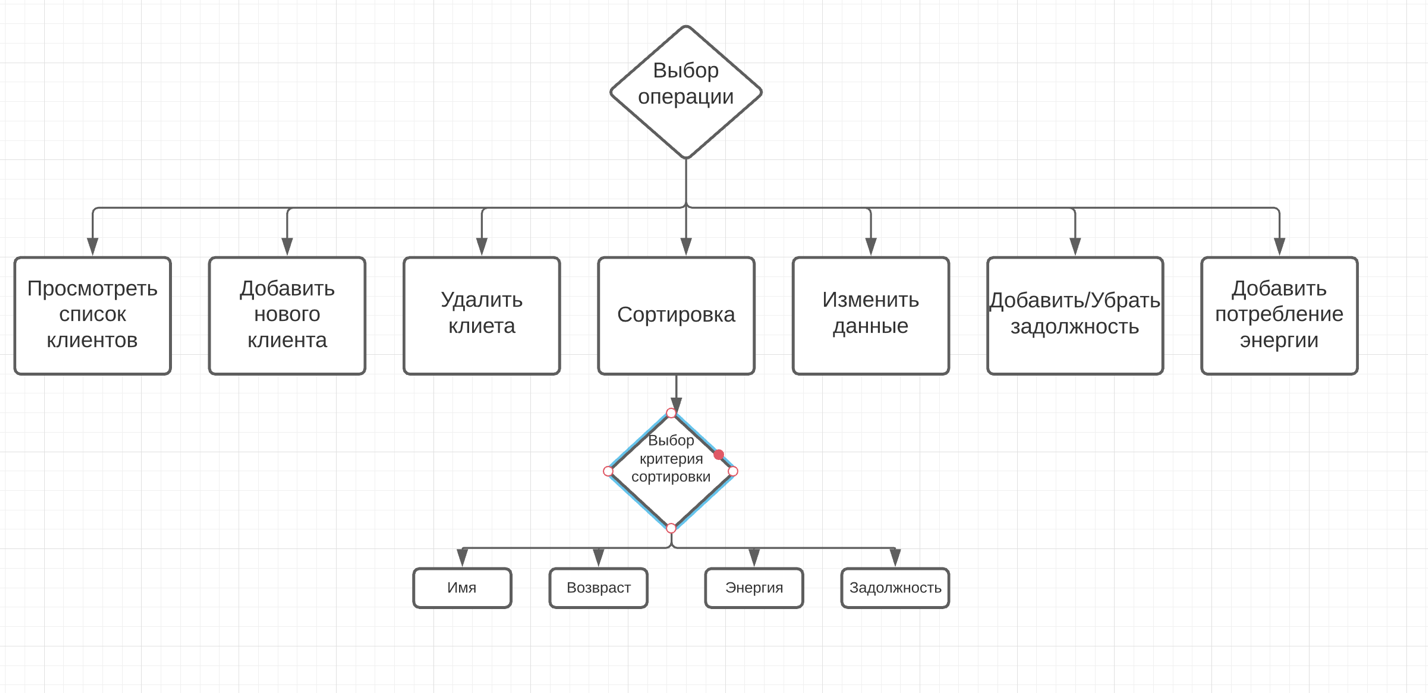


Рис 3

В текущей главе представлено описание алгоритма (рис 3), который имеет начало алгоритма и конец, внутри алгоритма имеются циклы и иные операций.

Первым пунктом идёт выбор операции, за счет введение цифры от 1 до 7 и последующей за вводом проверки.

1 - Просмотреть список клиентов

2 - Добавить нового клиента

3 - Удалить клиента

4 - Сортировка

5 - Изменить данные

6 - Добавить/Убрать задолженность

7 - Добавить потребление энергии

Следующий шаг — это работа непосредственно с программой в зависимости от выбранных данных. Добавление данных, вывод данных,удаление клиентов, сортировка, изменение данных.

# **4.ОПИСАНИЕ ИНТЕРФЕЙСА**

При старте программы происходит выбор желаемой операции (рис.4.1)

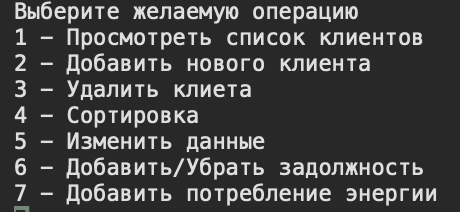


Рис. 4.1

При выборе первого пункта на консоль выводиться все данные из файла (рис. 4.2).



Рис.4.2

При выборе второго пункта пользователю необходимо поочерёдно вводить данные. А после ввода данных происходит сохранение данных в xml-файл (рис 4.3).

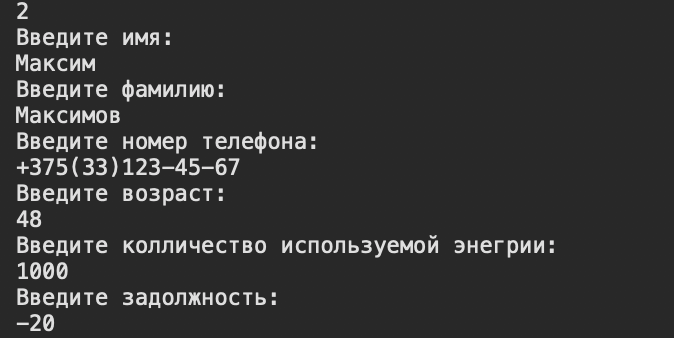


Рис 4.3

При выборе третьего пункта (рис 4.4) происходит удаление элемента. Вначале на консоль выводиться количество элементов. И необходимо выбрать элемент, входящий в область указанного значения.

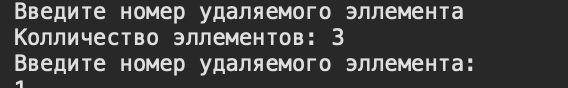


Рис.4.4

В случае же если значение указано неверно, то выводится следующая надпись (рис.4.5).



Рис.4.5

При выборе четвертого пункта происходит сортировка. Вначале происходит последующий выбор (имя, возраст, энергия, задолженность), по какому пункту будет происходить сортировка (рис.4.6).

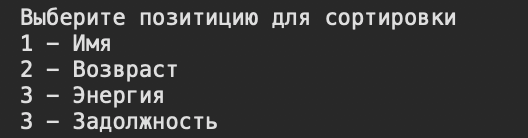


Рис.4.6

А потом отсортированные данные выводятся на консоль (рис 4.7).



Рис 4.7

Изменение данных происходит схожим образам с добавление и удалением (рис 4.4). Вначале надо выбрать необходимый элемент. После этого происходит последующая последовательная изменение данных(рис.4.8).

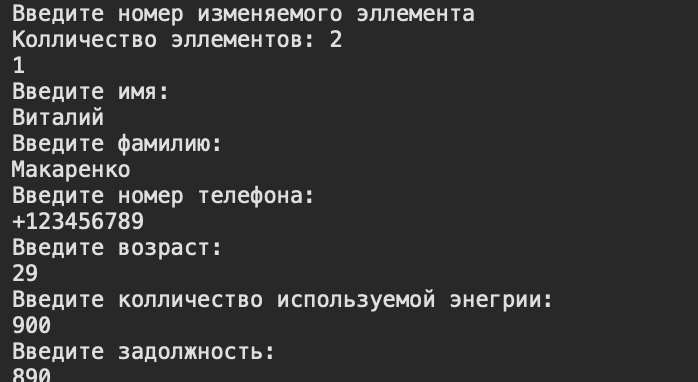


Рис.4.8

При выборе пункта номер 6 добавление или уменьшение задолжники (рис 4.9)

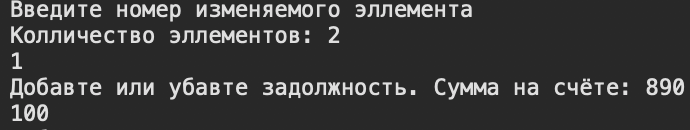


Рис 4.9

При выборе пункта номер 7 добавление потребления энергии (рис 4.10)

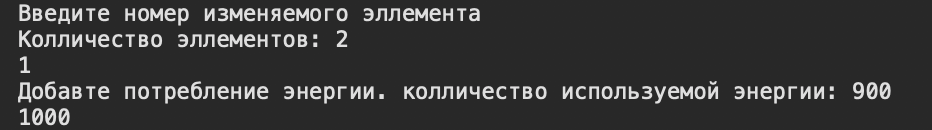


Рис 4.10

В случае же если выбран неверный пункт, то на консоль выводятся надпись (рис. 4.11).



Рис 4.11

## 

# **5.ВЫБОР СПОСОБА ОРГАНИЗАЦИИ ДАННЫХ**

Программа состоит одного класса.

Класс **Client** (рис 5.1):

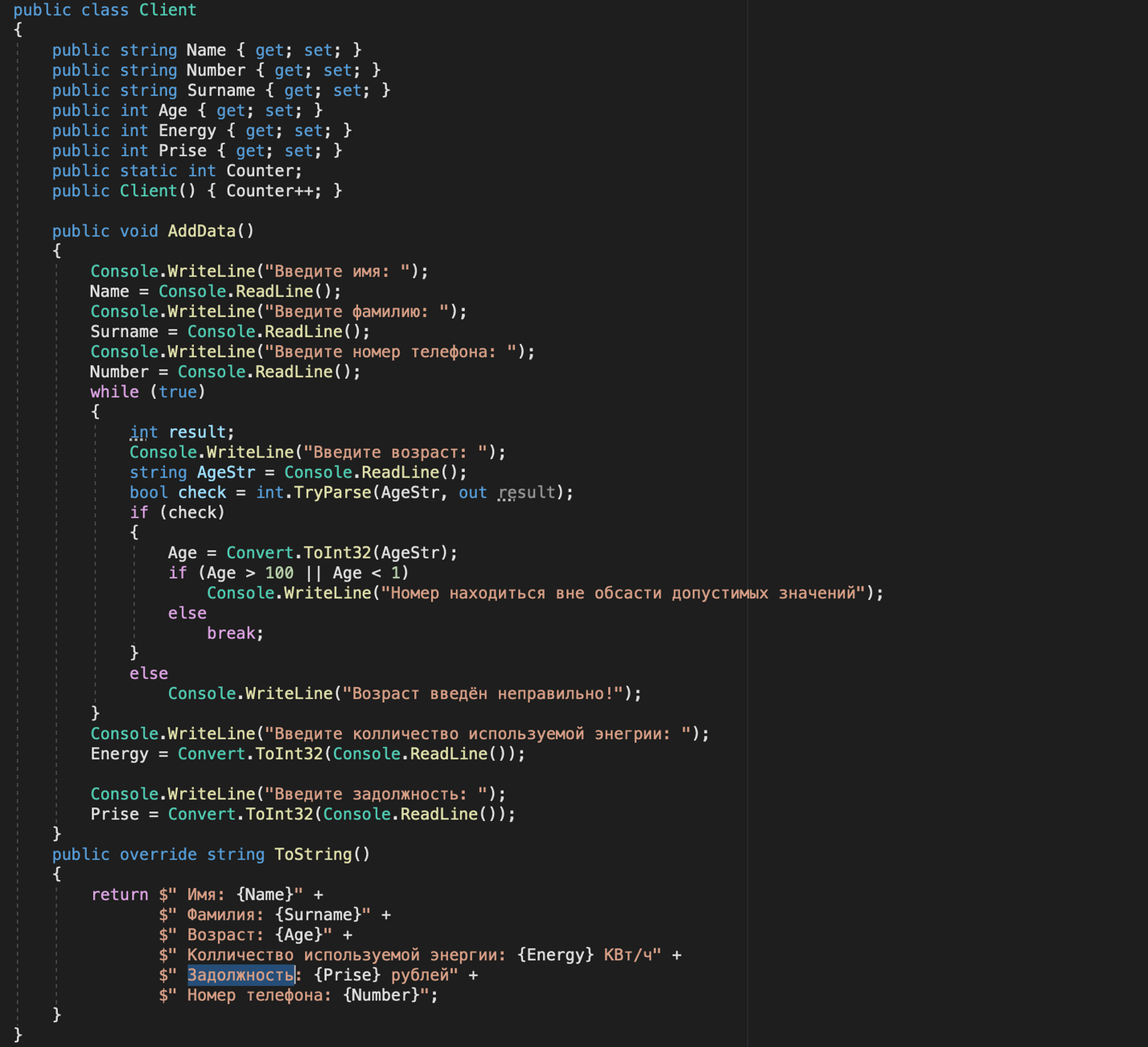


Рис 5.1

Переменные: **string** Name, **string** Surname, **int** Age, **int** Prise, **int** Energy, **string** Number – публичные переменные, которые служат в качестве информации об клиент.

Counter – статическая переменная служащая для определения общее количество клиентов.

Методы:

AddData – **данный** метод является частью класса Client. Он необходим для заполнения или редактирование данных клиента. Работа данного метода заключается в заполнении полей класса данными с консоли (Name, Surname, Age, Prise, Energy, Number). В методе используются механизмы ветвления и циклы, для реализации алгоритма вариации возраста. Данный метод не возвращает значений.

override ToString() – данный метод является перегруженный метод из стандартной библиотеки. Он необходим для представления пользовательских типов данных в строковом виде. В данной программе этот метод перегружен для вывода переменныx: Name, Surname, Age, Prise, Energy, Number. Данный метод возвращает строковое значение.

Так же в классе реализован конструктор без параметров. Его главная и единственная задача считать количество созданных экземпляров класса. Так как чтение из файла объекта класса происходит через конструктора без параметров.

# **6.РАЗРАБОТКА ПЕРЕЧНЯ ПОЛЬЗОВАТЕЛЬСКИХ МЕТОДОВ ПРОГРАММЫ**

**void** Print () — это функция, состоящая из одной переменной типа IEnumerable<Client> - это коллекцию которую метод будет выводить на консоль. Работа функции построена последовательно, в ней не предусмотрен циклический метод. Функция дает пользователю вывести список клиентов на консоль. Данная функция не возвращает значений.

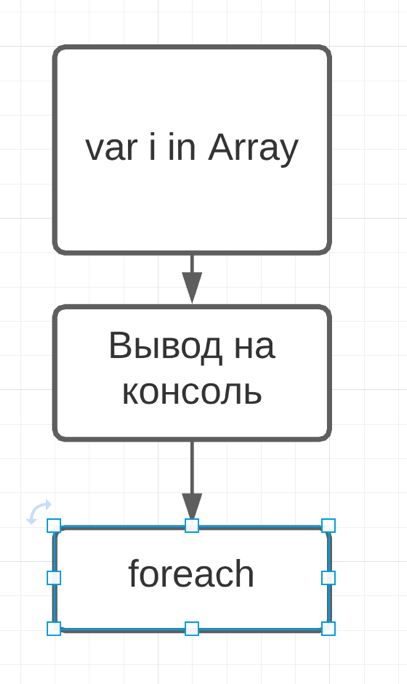
**void** Save () — это функция, состоящая из двух переменных XmlSerializer – предназначенный для записи данных в xml-файл и List<Client> - непосредственно коллекция для которая будет записываться в файл. Данные функция необходима для сохранения списка в файл. Для оптимальной работы потока ввода в файл была использована конструкция using. Так как при её использовании нет необходимости следить за закрытием потока. Работа функции построена последовательно. Данная функция не возвращает значений.

**void** AddData() **–** данный метод является частью класса Client. Он необходим для заполнения или редактирование данных клиента. Работа данного метода заключается в заполнении полей класса данными с консоли(Name, Surname, Age, Prise, Energy, Number). В методе используются механизмы ветвления и циклы, для реализации алгоритма вариации возраста. Данный метод не возвращает значений.

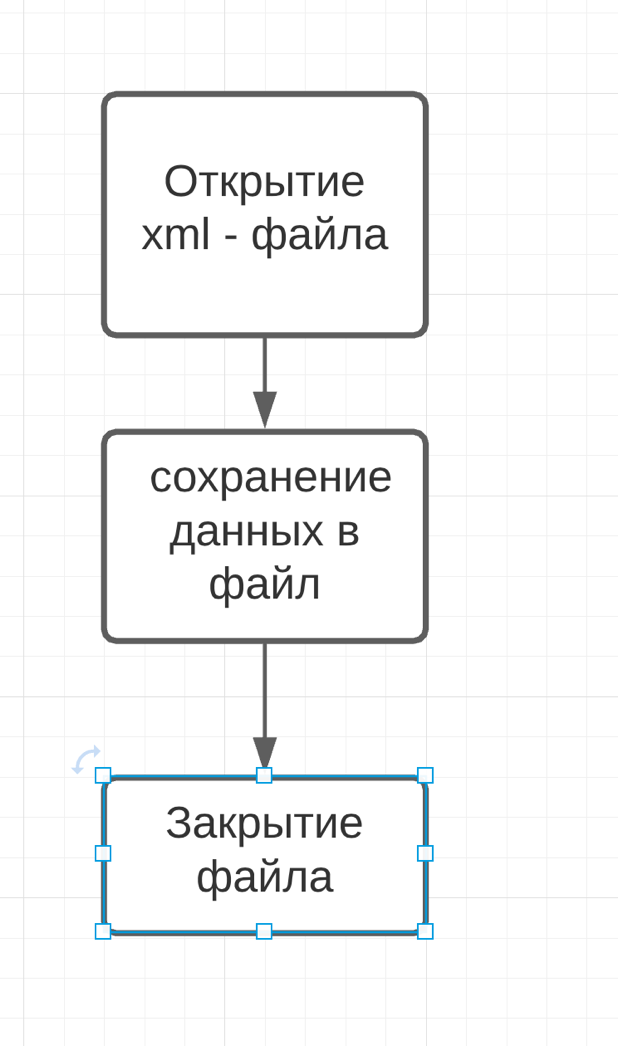
override **string** ToString() – данный метод является перегруженный метод из стандартной библиотеки. Он необходим для представления пользовательских типов данных в строковом виде. В данной программе этот метод перегружен для вывода переменныx: Name, Age, Number, Musical\_Instrument. Данный метод возвращает строковое значение.

# **7.ПРОГРАММНАЯ ЧАСТЬ**

void Print()



void Save()



Вначале происходит считывание данных из файла и заполнение данных в список. В случае же возникновение каких-либо ошибок они обрабатываются и выводиться на консоль.

Затем происходит выбор операции:

1 - Просмотреть список клиентов

2 - Добавить нового клиента

3 - Удалить клиента

4 - Сортировка

5 - Изменить данные

6 - Добавить/Убрать задолженность

7 - Добавить потребление энергии

Далее при помощи конструкции switch происходит переход к выбранной операции.

1: Циклом foreach происходит итерация по списку. При каждой итерации происходит вызов перегруженного метода ToString. Который выводит в консоль данные каждого клиента (имя, возраст, энергия, задолженность). Так же в цикле реализована присвоение Id при помощи одноименной переменной.

2: Вначале создаётся новый(пустой) клиент. Далее вызывается метод класса Client AddData.

Его суть заключается в том, что пользователь поочерёдно вводит данные с консоли. Поле Age имеет проверку на валидность вводимых данных. Т.е проверка являются ли вводимые данные числом и не выходит ли данное число из области допустимых значений.

Далее происходит добавление клиента в список, а затем происходит сохранение дополненного списка в файл при помощи функции Save – который принимает 2 параметра (Переменная для сохранения в файл и список клиентов).

3: Удаление происходит схожим образом с добавлением из-за сходства данных алгоритмов и появилась необходимость создание функции Save.

Удаление элементов происходит по номеру клиента в списке. Вначале на консоль выводится количество элементов чтобы пользователю было удобнее взаимодействовать в программе. А далее пользователь вводит желаемый элемент для удаления. Так же приходит проверка данных на валидность. Т.е проверка являются ли вводимые данные числом и не выходит ли данное число из области допустимых значений.

4: Сортировка данных происходит при помощи библиотеки Linq по следующим параметрам: имя, возраст, энергия, задолженность. После же сортировки элементов происходит вывод отсортированного списка на консоль.

5: Изменение происходит схожим образом с удалением из-за сходства данных алгоритмов и появилась необходимость создание функции Save. Так как изменение происходит так же с выбором изменяемого элемента и последующей проверкой введенного номера на валидность. Вот только не производит создание нового элемента, а из выбранного элемента списка происходит вызов метода AddData.

6: Взаимодействие с задолженностью происходит по номеру клиента в списке. Вначале на консоль выводится количество элементов чтобы пользователю было удобнее взаимодействовать в программе. А далее пользователь вводит желаемый элемент для изменения задолженности. Так же приходит проверка данных на валидность. Т.е проверка являются ли вводимые данные числом и не выходит ли данное число из области допустимых значений.

Далее происходи ввод числовое значение, которое будет добавляться к переменной Prise.

7: Добавление потребление энергии происходит по номеру клиента в списке. Вначале на консоль выводится количество элементов чтобы пользователю было удобнее взаимодействовать в программе. А далее пользователь вводит желаемый элемент для добавления счётчика энергопотребления. Так же приходит проверка данных на валидность. Т.е проверка являются ли вводимые данные числом и не выходит ли данное число из области допустимых значений.

Далее происходи ввод числовое значение, которое будет добавляться к переменной Energy.

# **ЗАКЛЮЧЕНИЕ**

Исходя из поставленных целей и задач было сконструировано, разработана и протестирован программный модуль «Учет расхода электроэнергии», предназначенный для использования работниками службы электроснабжения. Были реализованы задачи:

1)Быстро и гибко редактироваться со стороны владельцев службы электроснабжения.

2)Программный продукт имеет красивый, понятный и интуитивный интерфейс и владельцу приятно пользоваться программой.

3)Так же предусмотрено возможность добавления новых функций в программе.

Цель была выполнены, картотека имеет следующие преимущества:

1)Работа службы электроснабжения происходит быстро и максимально эффективно.

2)Удовлетворяет запросы службы электроснабжения.

2)Также реализована возможность пополнять базу, для оптимизации процессов службы электроснабжения.

3)Реализована высокая оптимизация, что-то привело к возможности пользователем запускать приложение даже на самом слабом устройстве.

Данный программный продукт выгодно отличается от других подобных тем что, пользуясь им, пользователь может не только просматривать клиентов, но и производить различные взаимодействия с самими клиентами.

Таким образом программа работает без сбоев и все задачи были выполнены.

# **ПРИЛОЖЕНИЕ**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Xml.Serialization;

namespace Pavel\_Kursach

{

class Program

{

static async System.Threading.Tasks.Task Main(string[] args)

{

while (true)

{

List<Client> Array = new List<Client>();

XmlSerializer formatter = new XmlSerializer(typeof(List<Client>));

try

{

using (FileStream fs = new FileStream("client.xml", FileMode.OpenOrCreate))

{

Array = (List<Client>)formatter.Deserialize(fs);

}

}

catch(InvalidOperationException ex)

{

Console.WriteLine(ex.Message);

}

Console.WriteLine("Выберите желаемую операцию");

string selection = Console.ReadLine();

switch (selection)

{

case "1":

int i = 0;

foreach (Client p in Array)

{

Console.WriteLine($"Id: {i}" + p.ToString());

i++;

}

break;

case "2":

Client client = new Client();

client.AddData();

Array.Add(client);

Save(formatter, Array);

break;

case "3":

Console.WriteLine($"Введите номер удаляемого эллемента\nКолличество эллементов: {Array.Count}");

int Number;

while (true)

{

int result;

Console.WriteLine("Введите номер удаляемого эллемента: ");

string NumberStr = Console.ReadLine();

bool check = int.TryParse(NumberStr, out result);

if (check)

{

Number = Convert.ToInt32(NumberStr);

if (Number > Array.Count || Number < 1)

Console.WriteLine("Номер находиться вне обсасти допустимых значений");

else

break;

}

else

Console.WriteLine("Номер введён неправильно!");

}

Array.RemoveAt(Number - 1);

File.Delete("client.xml");

Save(formatter, Array);

break;

case "4":

Console.WriteLine("Выберите позитицию для сортировки");

Console.WriteLine("1 - Имя");

Console.WriteLine("2 - Возвраст");

Console.WriteLine("3 - Энергия");

Console.WriteLine("3 - Задолжность");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

IEnumerable<Client> sortedAge = Array.OrderBy(i => i.Age);

Print(sortedAge);

break;

case "2":

IEnumerable<Client> sortedName = Array.OrderBy(i => i.Name);

Print(sortedName);

break;

case "3":

IEnumerable<Client> sortedEnergy = Array.OrderBy(i => i.Energy);

Print(sortedEnergy);

break;

case "4":

IEnumerable<Client> sortedPrise = Array.OrderBy(i => i.Prise);

Print(sortedPrise);

break;

default:

Console.WriteLine("Вы выбрали неверный пунтк");

break;

}

break;

case "5":

int number = 0;

while (true)

{

int result;

Console.WriteLine($"Введите номер изменяемого эллемента\nКолличество эллементов: {Array.Count}");

string NumberStr = Console.ReadLine();

bool check = int.TryParse(NumberStr, out result);

if (check)

{

number = Convert.ToInt32(NumberStr);

if (number > Array.Count || number < 1)

Console.WriteLine("Номер находиться вне обсасти допустимых значений");

else

break;

}

else

Console.WriteLine("Номер введён неправильно!");

}

Array[number - 1].AddData();

File.Delete("client.xml");

Save(formatter, Array);

break;

case "6":

int numbe = 0;

while (true)

{

int result;

Console.WriteLine($"Введите номер изменяемого эллемента\nКолличество эллементов: {Array.Count}");

string NumberStr = Console.ReadLine();

bool check = int.TryParse(NumberStr, out result);

if (check)

{

numbe = Convert.ToInt32(NumberStr);

if (numbe > Array.Count || numbe < 1)

Console.WriteLine("Номер находиться вне обсасти допустимых значений");

else

break;

}

else

Console.WriteLine("Номер введён неправильно!");

}

Console.WriteLine($"Добавте или убавте задолжность. Сумма на счёте: {Array[numbe - 1].Prise} ");

Array[numbe - 1].Prise += Convert.ToInt32(Console.ReadLine());

File.Delete("client.xml");

Save(formatter, Array);

break;

case "7":

int numb = 0;

while (true)

{

int result;

Console.WriteLine($"Введите номер изменяемого эллемента\nКолличество эллементов: {Array.Count}");

string NumberStr = Console.ReadLine();

bool check = int.TryParse(NumberStr, out result);

if (check)

{

numb = Convert.ToInt32(NumberStr);

if (numb > Array.Count || numb < 1)

Console.WriteLine("Номер находиться вне обсасти допустимых значений");

else

break;

}

else

Console.WriteLine("Номер введён неправильно!");

}

Console.WriteLine($"Добавте потребление энергии. колличество используемой энергии: {Array[numb - 1].Energy} ");

Array[numb - 1].Energy += Convert.ToInt32(Console.ReadLine());

File.Delete("client.xml");

Save(formatter, Array);

break;

default:

Console.WriteLine("Данные введены не верно");

break;

}

}

}

public static void Save(XmlSerializer formatter, List<Client> Array)

{

using (FileStream fs = new FileStream("client.xml", FileMode.OpenOrCreate))

{

formatter.Serialize(fs, Array);

}

}

public static void Print(IEnumerable<Client> Array)

{

foreach (var i in Array)

{

Console.WriteLine(i.ToString());

}

}

}