Алгоритмизация и Програмиране C#

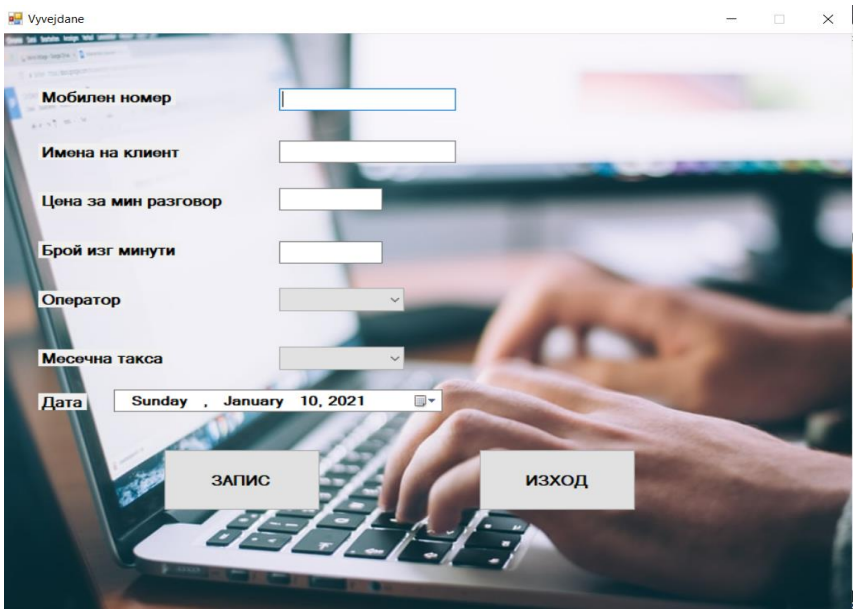
Курсова работа Ф.№114713.

Задание : създаване на информационна система и обработка на двоични файлове . По-конкретно за обработка на данни (въвеждане , извеждане и търсене на информация). Конкретният файл съм го разделил на подточки , където всяка отделна подточка описва стъпките , които съм извършил по задачата.

➤1. Създаване на Главна форма “MainForm.[Design]” и “MainForm.cs”.

В “MainForm.[Design]” съм използвал “Menu Strip Bar ” , където съм въвел три хедъра “Данни”, ”Търсене” и накрая ”Изход”, съобразно вашите изисквания .

➤2. В хедъра “Данни” съм направил три отделни формата , които са за “Въвеждане на данни” , ”Извеждане на данни ” и “ Извеждане на клиенти и такси (според оператора)”.



Във “Въвеждане на данни” > “Vyvejdane.Design” съм поставил 6 лейбъла : “Мобиленномер “ , ”Имена” , ”Цена” , ”Бр. Изг мин.” , ”Оператор”, и “Месечна такса”.

.Използвал съм 4 пъти textBox за първите четири отдясно . И 2 пъти ComboBox за “Оператор” и “ М.такса”.

В ComboBox на “Оператор” съм използвал следния метод...

```
1 reference
private void Vyvejdane_Load(object sender, EventArgs e)
{
    StreamReader txt = new StreamReader("operator.txt");
    string ivan;
    while ((ivan = txt.ReadLine()) != null) comboBox1.Items.Add(ivan);
    txt.Close();

    StreamReader txt1 = new StreamReader("Mtaksa.txt");
    string ivan1;
    while ((ivan1 = txt1.ReadLine()) != null) comboBox2.Items.Add(ivan1);
    txt.Close();
}
```

Във “Vyvejdane Load” съм използвал системния метод най-горе > using System IO;

Който метод ни позволява използването на StreamReader & Writer с цел да записване и обработване на двоични файлове. Създал съм 2 ноутпада , в които един съдържа операторите “А1”,”Теленор” и ”Виваком” , а във втория съм въвел посочените от вас стойности (5 , 7 , 10 и 12 лв.). Също така съм използвал и цикъла {while} ,благодарение , на него той използва ComboBoxa с цел въвеждане на данни от конкретния ноутпад ..

“operator.txt” , ”Mtaksa.txt” .

```
private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
{
if ((e.KeyChar < '0' || e.KeyChar > '9')&& e.KeyChar!=8 && e.KeyChar!='.' && e.Key
}

1 reference
private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
{
if ((e.KeyChar < '0' || e.KeyChar > '9')&& e.KeyChar!=8) e.KeyChar = '\0';
}

1 reference
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
if ((e.KeyChar < '0' || e.KeyChar > '9')&& e.KeyChar!=8) e.KeyChar = '\0';
}
```

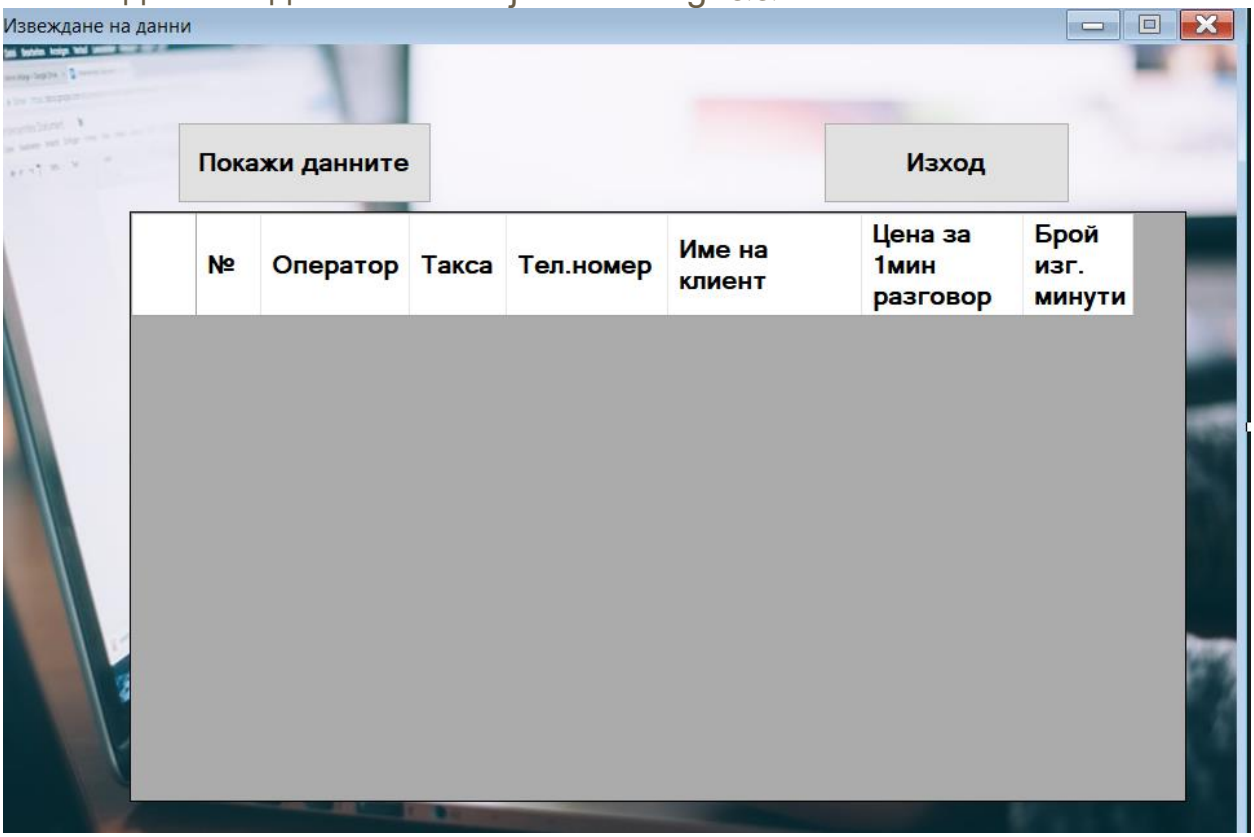
В 4-рите textBoxa съм ползвал разширението “Key_press” , което служи като “Warning” , че в определеното поле може да се въвеждат само стойности или букви.. Също така съм направил “MessageBox” , който предупреждава , че в някое от полетата не е въведена стойност и извежда “Reminder” , който показва тест “Не сте въвели стойност” и също така пренасочва курсора към определеното поле за въвеждане на данни .

```
int opt = comboBox1.SelectedIndex;
int taksa = comboBox2.SelectedIndex;
string nomer =textBox1.Text;
string ime = textBox2.Text;
double minrazg = double.Parse(textBox3.Text);
int brizg = int.Parse(textBox4.Text);
/ string data = dateTimePicker1.Value.Date.ToString();


FileStream f =new FileStream("Kursova1.dat", FileMode.OpenOrCreate,FileAccess.Write);
f.Seek(0, SeekOrigin.End);
BinaryWriter b = new BinaryWriter(f);
b.Write(opt);b.Write(taksa); b.Write(nomer); b.Write(ime); b.Write(minrazg); b.Write(brizg); //b.W
f.Close();
MessageBox.Show("Данните са запазени !");
 Close();
```

В тази снимка е описано , всяка една дейност как е въведена : според оператор , такса , номер и така нататък . Всяко едно действие има определена стойност : {int , string , double} . int служи за въвеждане на стойности без десетична запетая (с други думи точни стойности) , докато при (double) стойностите са с десетична запетая .string от своя страна служи за въвеждане на текст(текстов формат) така и стойности. По-надолу е създаден файлът “Kursova 1.dat ” , който служи за въвеждане , триене и четене на файла . {f.Seek(0, SeekOrigin.End); служи за запаметяване на текущи и минали данни

(стойности) ..... (BinaryWriter(f)); е стойност , която записва изредените по-долу стойности в конкретната последователност.. И като за завършек съм използвал отново структура “MessageBox.” и метода ”Show” . С цел , когато всички данни са въведени да изскочи текста , че данните са въведени успешно.

➤3. В хедъра “Данни” има още две форми , една от които е “Извеждане на данни” > “Izvejdane.Design&&.cs”



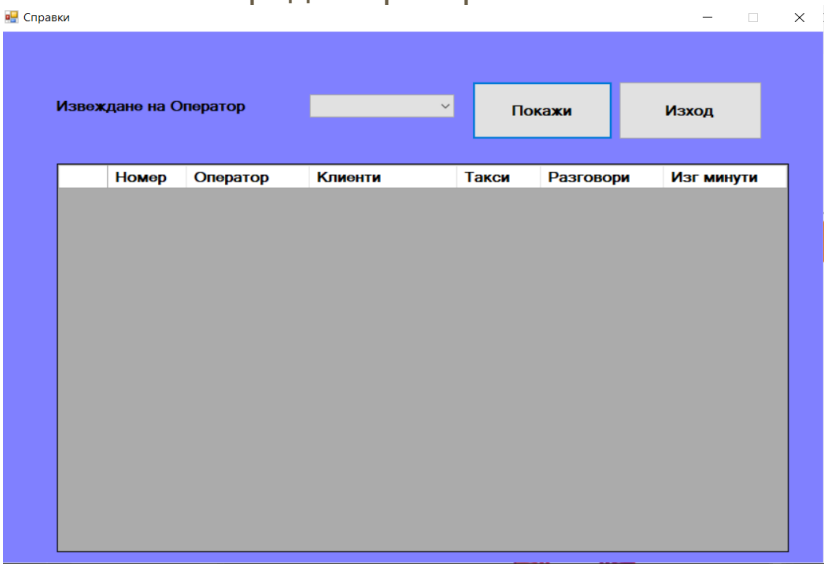
В формата “Izvejdane.Design” има 2 Buttons : За показване на данните и Изход , също така имам (data.Grid.View) , който представя данните в табличен вид в посочената от мен последователност. №,Оператор,Такса,Номер,Име,Цена,Бр. Мин.. В снимката по-долу съм използвал същите методи , както при формата за “Въвеждане на данни” по-точно относно : ноутпад и данните съдържащи се в него за (Оператор и Месечна Такса).Има и друга прилика ,а тя е че съм използвал същите стойности като {int , string , double} във всеки един файл формат , защото ако ги няма данните (не съответстват) или ги няма , програмата ще спре и няма да изпълни оператора няма да изпълни посоченият от нас цикъл или операция . Същото се отнася и за FileStream & BinaryReader ....

```
string[] ivan = new string[6];
StreamReader iv = new StreamReader("operator.txt");
string a; int x = 0;
while ((a = iv.ReadLine()) != null) ivan[x++] = a;
iv.Close();

string[] ivan1 = new string[8];
StreamReader iv1 = new StreamReader("Mtaksa.txt");
string a1; int b = 0;
while ((a1 = iv1.ReadLine()) != null) ivan1[b++] = a1;
iv1.Close();

FileStream fd = new FileStream("Kursova1.dat", FileMode.Open, FileAccess.Read);

BinaryReader rd = new BinaryReader(fd);
```

Единствената разлика между тази форма и предишната “Въвеждане на данни” , е че тука се използва {Data.Grid.View} с цел улеснение възпроизвеждането на данните в табличен вид с този код :

```
dataGridView1.Rows.Add(n++,ivan[opt],ivan1[taksa], nomer, ime, minrazg + "ст", brizg + "мин");
```

➤4. Трета форма от главния хедър “Данни” е “Извеждане на стойности според оператора”.

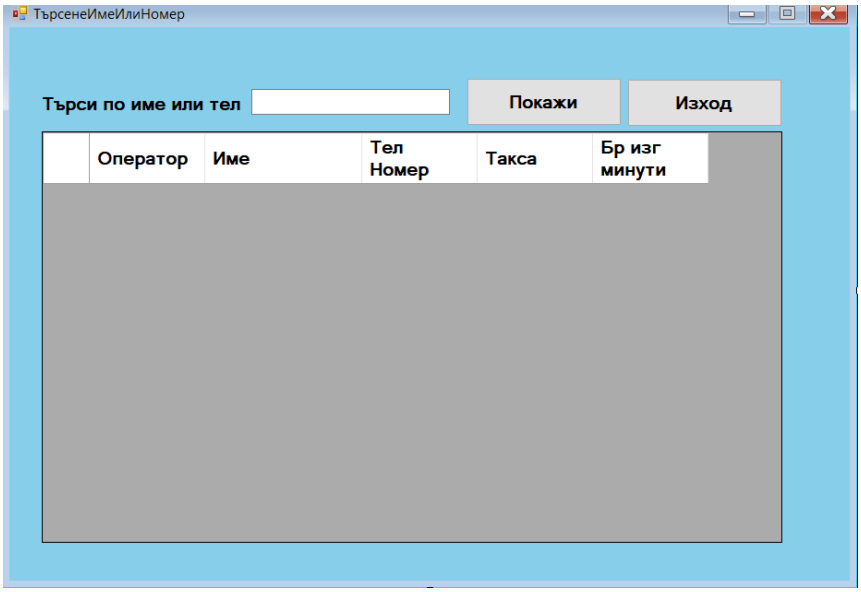


Целта на тази програма е да изведем горе описаните данни , чрез избран от нас Оператор (А1,Теленор,Виваком или др.) , отново с 2 бутона “Покажи ” и “ Изход ”.Също така с един ComboBox , който ни представя гореспоменатите оператори в падащо меню и не на последно място отново използваме {Data.Grid.View} , но този път с

разменени места на конкретните стойности .

```
dataGridView1.Rows.Clear();
string[] ivan = new string[6];
StreamReader iv = new StreamReader("operator.txt");
string a; int x = 0;
while ((a = iv.ReadLine()) != null) ivan[x++] = a;
iv.Close();

string[] ivan1 = new string[8];
StreamReader iv1 = new StreamReader("Mtaksa.txt");
string a1; int b = 0;
while ((a1 = iv1.ReadLine()) != null) ivan1[b++] = a1;
iv1.Close();
FileStream f = new FileStream("Kursova1.dat",FileMode.Open);
BinaryReader br = new BinaryReader(f);
int nom = 0;
while (f.Position < f.Length)
{
   int opt; int taksa;string nomer; string ime; double minrazg; int brizg; //string data;
    opt = br.ReadInt32();
    taksa = br.ReadInt32();
    nomer = br.ReadString();
    ime = br.ReadString();
    minrazg = br.ReadDouble();
    brizg = br.ReadInt32();
    //data = br.ReadString();

    if(opt == comboBox1.SelectedIndex)
    dataGridView1.Rows.Add(++nom,ivan[opt],ime,ivan1[taksa],minrazg+"ст",brizg+"мин");
}
```

В тази формата , както предишните сме използвали свойствата на ноутпад за добавяне на написани от нас стойности като оператор и такси.. Също така сме използвали абсолютно същата последователност на стойностите от тип {int , string , double} и единствената {Data.Grid.View1.Rows.Add} разлика има тука и тя ,е че извеждането на стойностите в колони е под различна структура на възпроизвеждане . ➤ Сега преминаваме към втория основен хедър “ Търсене ” при , който извеждането на данните се избира от потребителя дали да търси по “ Име , Номер” или нещо конкретно …



В тази форма [FormaPoImeTel.Design] отново има добавен {Data.Grid.View} 2 бутона за “ Покажи ” и “ Изход ” , разлика между този файл форм и другите , е че тука използвам textBox , чрез който се проверява търсеният от нас елемент име или тел.номер..

В самия код повечето стойности и типове данни се повтарят , както при преходните форми като : ноутпад за извеждането на оператор и месечна такса .. , създаване , редактиране или съхраняването на файл от типа “Kursova1.dat”. Единственото ново тука е начинът на търсене .

```
if (ime.ToUpper().StartsWith(find) || nomer.ToString().StartsWith(find))
    dataGridView1.Rows.Add(ivan[opt],ime,nomer, ivan1[taksa], brizg + "мин" );
```

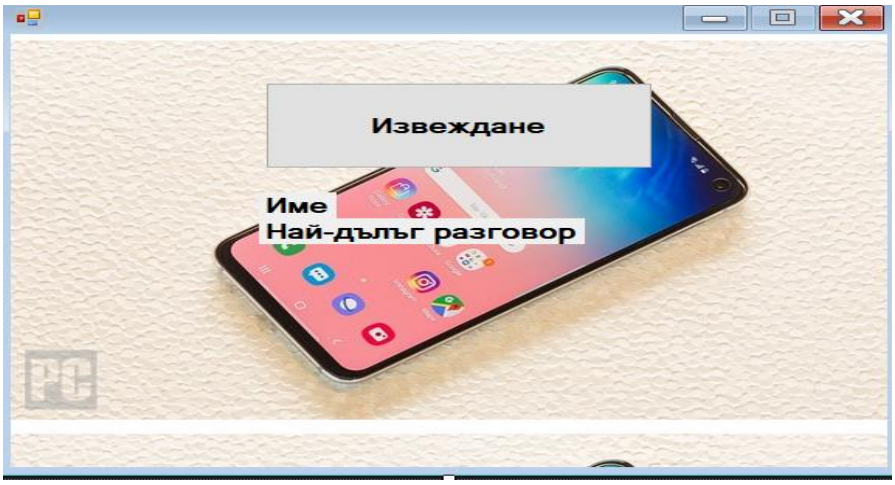
В този ред използвам цикъла “if” за да може потребителя да изведе данните по тип име или номер на клиента с метода “find” . Двете стойности имат прилика и тя , е че са от тип стрингова променлива , но разликата между двете , е че при “Име” се търси с начало на буква , с която да се ориентира програмата коя стойност да изведе и дали изобщо има така стойност , докато при номер разликата , е че тук се търси с начална стойност като например тел.номер започващ > 0894 или 0878 и т.н. ..

И разбира се в {Data.Grid.View1} възпроизвеждането на стойности е под друг ред в сравнение с предишните форми.

Също така съм създал “MessageBox” , който да информира потребителя ако въведе грешна стойност , че системата няма да изпълни избраният от него процес.

```
if (dataGridView1.RowCount == 0)
{
    MessageBox.Show("Няма намерени данни !");
}
```

➤Втора форма от хедъра “Търсене ” е “Извеждане на клиент с най-дълъг разговор”



В тази форма дизайнът се състои от един главен бутон , който извежда контакта с най-дълъг разговор в стойност и името на клиента с лейбъл 1 и 2.

```
FileStream fd = new FileStream("Kursova1.dat", FileMode.Open, FileAccess.Read);
BinaryReader rd = new BinaryReader(fd);
int min = -1;

while (fd.Position < fd.Length)

{


    int opt = rd.ReadInt32();
    int taksa = rd.ReadInt32();
    string nomer = rd.ReadString();
    string ime = rd.ReadString();
    double minrazg = rd.ReadDouble();
    int brizg = rd.ReadInt32();
    //string data = rd.ReadString();

   if(min < brizg)
    {
        min = brizg;


    }
     label1.Text = "Клиентът с най-дълъг разговор: "+ min;

     label2.Text = "" + ime;
```

За реализирането на този кодов формат съм въвел стойностите както преди така и тука в точно определена последователност и използването на FileStream,BinaryReader

От тип (using.System.IO;) . И съм използвал цикъла “ if ” за сравнение , ако минутите са по-малко от броя изг.мин. Дали минутите са равни или по-малки от бр изг.мин и ако да в label1.Text да изведе “ Клиентът с най-дълъг разговор ” + минутите за разговор , а в label2.Text извежда конкретния потребител , който е с най-дълъг разговор.

Изготвил:Иван Саръбеев

Специалност:БИС