Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Специальность 09.02.03 «Программирование в компьютерных системах»

**КВАЛИФИКАЦИОННЫЙ ЭКЗАМЕН**

Профессиональный модуль ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

*(наименование профессионального модуля)*

Выполнил:

обучающийся группы № 3ПКС-320

Бек-Назарова Анастасия Николаевна

*(И.О. Фамилия)*

Проверили:

Г.И. Киреева

*(И.О. Фамилия)*

И.В.Сибирев

*(И.О. Фамилия)*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

**2022**

Билет№1

Код:

using System;

using System.Collections.Generic;

using System.Linq;

using Binarysharp.Assemblers.Fasm;

using Process.NET;

using Process.NET.Native.Types;

using System.Runtime.InteropServices;

using System.Security;

using System.Text.RegularExpressions;

using System.IO;

namespace Билет\_1\_Задание\_1\_Бек\_Назарова\_\_ПКС\_420

{

public class Student

{

string Surame = " ";

List<int> Grade = new List<int>();

public Student(string surname, string grade)

{

this.Surame = surname;

this.Grade = grade.Split(',').Select(x => int.Parse(x)).ToList();

}

public List<int> AccessGrade()

{

return Grade;

}

public string surname()

{

return this.Surame;

}

public Student WriteThis()

{

Console.WriteLine(this.Surame + ":");

File.AppendAllText(Program.path, this.Surame + ":\n");

Console.Write("\tОценки: ");

File.AppendAllText(Program.path, "\tОценки: ");

foreach (int a in this.Grade) { Console.Write(a + ";"); File.AppendAllText(Program.path, a.ToString() + ';'); };

return this;

}

public int Summ() //Сумма оценок

{

int answer = 0;

if (!false)

{

FasmNet \_FasmNet = new FasmNet();

\_FasmNet.AddLine("use32");

\_FasmNet.AddLine("xor eax, eax");

\_FasmNet.AddLine("xor ebx, ebx");

\_FasmNet.AddLine("xor ecx, ecx");

\_FasmNet.AddLine("mov ebx, dword [ebp+12]");

\_FasmNet.AddLine("\_start:");

\_FasmNet.AddLine("add eax, dword [ebx]");

\_FasmNet.AddLine("add ebx, 4");

\_FasmNet.AddLine("add ecx, 1");

\_FasmNet.AddLine("cmp ecx, dword [ebp+8]");

\_FasmNet.AddLine("je \_end");

\_FasmNet.AddLine("jmp \_start");

\_FasmNet.AddLine("\_end:");

\_FasmNet.AddLine("ret");

byte[] \_ByteS = \_FasmNet.Assemble();

Process.NET.Memory.IAllocatedMemory \_IAllocatedMemory = new ProcessSharp(System.Diagnostics.Process.GetCurrentProcess(), Process.NET.Memory.MemoryType.Local).MemoryFactory.Allocate(

name: "SamName",

size: \_ByteS.Length,

protection: MemoryProtectionFlags.ExecuteReadWrite);

\_IAllocatedMemory.Write(0, \_ByteS);

delSumm \_delSumm = Marshal.GetDelegateForFunctionPointer<delSumm>(\_IAllocatedMemory.BaseAddress);

answer = \_delSumm(this.Grade.Count, this.Grade.ToArray());

\_IAllocatedMemory.Dispose();

}

return answer;

}

[SuppressUnmanagedCodeSecurity]

[UnmanagedFunctionPointer(CallingConvention.Cdecl)]

public delegate int delSumm(int \_count, params int[] \_ints);

}

public class Program

{

public static string path = @"C:\Users\bekna\OneDrive\Рабочий стол\Квалиф Бек-Назарова 3ПКС-420\Билет№1\_Бек-Назарова\_3ПКС420.txt";

[SuppressUnmanagedCodeSecurity]

[UnmanagedFunctionPointer(CallingConvention.Cdecl)]

public delegate int delegateDel(int \_A, int \_B);

public delegate int delSumm(int \_count, int \_ints);

public static int WholePart(int \_A, int \_B) // высчитать целую часть числа

{

int answer = 0;

FasmNet \_FasmNet = new FasmNet();

\_FasmNet.AddLine("use32");

\_FasmNet.AddLine("xor eax, eax");

\_FasmNet.AddLine("mov eax, dword [ebp+8]");

\_FasmNet.AddLine("cdq");

\_FasmNet.AddLine("idiv dword [ebp+12]");

\_FasmNet.AddLine("ret");

byte[] \_ByteS = \_FasmNet.Assemble();

Process.NET.Memory.IAllocatedMemory \_IAllocatedMemory = new ProcessSharp(System.Diagnostics.Process.GetCurrentProcess(), Process.NET.Memory.MemoryType.Local).MemoryFactory.Allocate(

name: "SamName1",

size: \_ByteS.Length,

protection: MemoryProtectionFlags.ExecuteReadWrite);

\_IAllocatedMemory.Write(0, \_ByteS);

delegateDel \_delSumm = Marshal.GetDelegateForFunctionPointer<delegateDel>(\_IAllocatedMemory.BaseAddress);

answer = \_delSumm(\_A, \_B);

\_IAllocatedMemory.Dispose();

return answer;

}

public static int DecimalPart(int \_A, int \_B) // высчитать десятичную часть числа

{

int answer = 0;

FasmNet \_FasmNet = new FasmNet();

\_FasmNet.AddLine("use32");

\_FasmNet.AddLine("xor eax, eax");

\_FasmNet.AddLine("mov eax, dword [ebp+8]");

\_FasmNet.AddLine("cdq");

\_FasmNet.AddLine("idiv dword [ebp+12]");

\_FasmNet.AddLine("mov eax, edx");

\_FasmNet.AddLine("ret");

byte[] \_ByteS = \_FasmNet.Assemble();

Process.NET.Memory.IAllocatedMemory \_IAllocatedMemory = new ProcessSharp(System.Diagnostics.Process.GetCurrentProcess(), Process.NET.Memory.MemoryType.Local).MemoryFactory.Allocate(

name: "SamName2",

size: \_ByteS.Length,

protection: MemoryProtectionFlags.ExecuteReadWrite);

\_IAllocatedMemory.Write(0, \_ByteS);

delegateDel \_delSumm = Marshal.GetDelegateForFunctionPointer<delegateDel>(\_IAllocatedMemory.BaseAddress);

answer = \_delSumm(\_A, \_B);

\_IAllocatedMemory.Dispose();

return answer;

}

public static int Achievement(int \_A, int \_B) // вычислить качественную успеваемость

{

int answer = 0;

FasmNet \_FasmNet = new FasmNet();

\_FasmNet.AddLine("use32");

\_FasmNet.AddLine("xor eax, eax");

\_FasmNet.AddLine("mov eax, dword [ebp+12]");

\_FasmNet.AddLine("cdq");

\_FasmNet.AddLine("idiv dword [ebp+8]");

\_FasmNet.AddLine("ret");

byte[] \_ByteS = \_FasmNet.Assemble();

Process.NET.Memory.IAllocatedMemory \_IAllocatedMemory = new ProcessSharp(System.Diagnostics.Process.GetCurrentProcess(), Process.NET.Memory.MemoryType.Local).MemoryFactory.Allocate(

name: "SamName3",

size: \_ByteS.Length,

protection: MemoryProtectionFlags.ExecuteReadWrite);

\_IAllocatedMemory.Write(0, \_ByteS);

delegateDel \_delSumm = Marshal.GetDelegateForFunctionPointer<delegateDel>(\_IAllocatedMemory.BaseAddress);

answer = \_delSumm(\_A, \_B);

\_IAllocatedMemory.Dispose();

return answer;

}

public static double Generalization(int \_A, int \_B)

{ return (WholePart(\_A, \_B) + DecimalPart(\_A, \_B) / (double)\_B); }

public static void Main(string[] args)

{

File.WriteAllText(path, "");

List<Student> Group = new List<Student>();

for (int i = 1; i <= 3; i++)

{

string Grd = "";

Console.Write($"Введите фамилию студента №{i}: ");

string sn = Convert.ToString(Console.ReadLine());

string surname = sn.Substring(0, 1).ToUpper() + sn.Substring(1);

if (Regex.IsMatch(surname, @"^[a-zA-Z]+$") || Regex.IsMatch(surname, @"^[а-яА-Я]+$"))

{

Random rnd = new Random();

int kolvo = rnd.Next(1, 11);

for (int j = 0; j < kolvo; j++)

{

int grades = rnd.Next(2, 6);

Grd += Convert.ToString(grades) + ",";

}

Grd = Grd.Remove(Grd.LastIndexOf(','));

Student student = new Student(surname, Grd);

Group.Add(student);

}

else

{

Console.WriteLine("Введен неверный формат данных, повторите ввод\n");

i--;

}

}

Console.WriteLine("\n\tИНФОРМАЦИЯ О СТУДЕНТАХ\n");

List<double> ArGrade = new List<double>();

int kolvoHorosh = 0;

for (int i = 0; i < 3; i++)

{

ArGrade.Add(Math.Round(Generalization(Group[i].Summ(), Group[i].AccessGrade().Count()), 2));

Group[i].WriteThis();

Console.WriteLine("\n\tСумма оценок: " + Group[i].Summ());

File.AppendAllText(path, "\n\tСумма оценок: " + Group[i].Summ() + "\n");

Console.WriteLine("\tСредняя оценка: " + ArGrade[i]);

File.AppendAllText(path, "\tСредняя оценка: " + ArGrade[i] + "\n");

}

for (int i = 0; i < 3; i++)

{

if (ArGrade[i] > 3.5)

{

kolvoHorosh++;

}

}

Console.WriteLine("\n\n\tКачественная успеваемость: " + Achievement(3, kolvoHorosh \* 100) + "%");

File.AppendAllText(path, "\n\n\tКачественная успеваемость: " + Achievement(3, kolvoHorosh \* 100) + "%\n\n\n");

Console.WriteLine("\n");

for (int i = 0; i < 3; i++)

{

List<int> fivemark = new List<int>();

int summ = 0;

File.AppendAllText(path, "Студент " + Group[i].surname() + ' ');

File.AppendAllText(path, "оценки за 5 работ : ");

for (int y = 0; y < 5; y++)

{

try

{

File.AppendAllText(path, Group[i].AccessGrade()[y] + " ");

fivemark.Add(Group[i].AccessGrade()[y]);

summ += Group[i].AccessGrade()[y];

}

catch

{

File.AppendAllText(path, "- ");

}

}

File.AppendAllText(path, ";\tCредняя оценка студента : ");

File.AppendAllText(path, Generalization(summ, fivemark.Count()) + "\n");

}

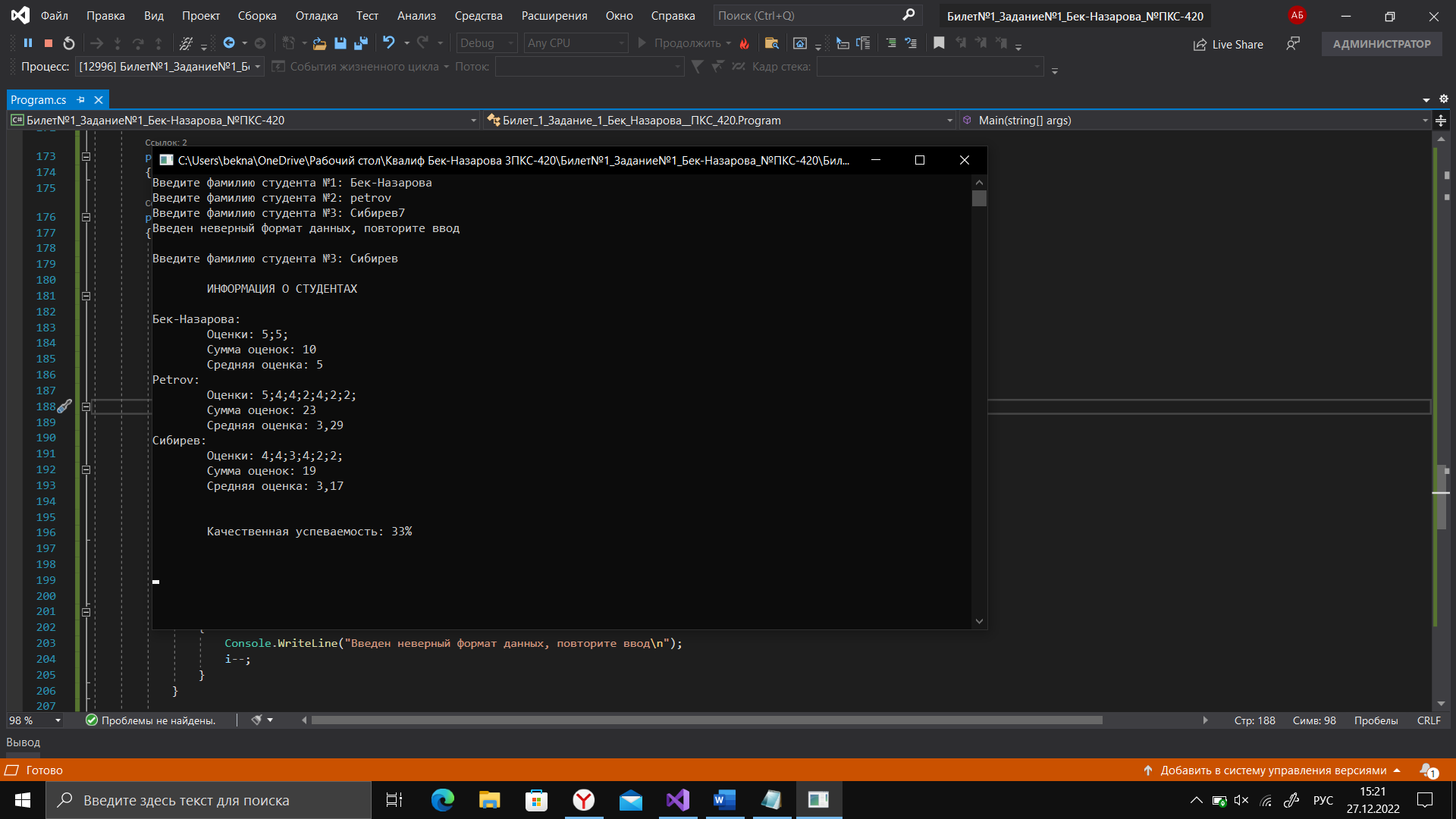
Console.ReadLine();

}

}

}

Вводимые и выводимые данные консоли:



1. Реализована проверка на вводимые символы. У пользователя есть возможность вводить фамилии на двух языках – русский и английский, но он не имеет возможности ввести какие-либо другие символы (кроме дефиса) или цифры.
2. При введении фамилии с «маленькой» буквы программа изменяет написание первой буквы на прописную («большую»)
3. Оценки за практические работы сформированы генератором случайных чисел.

Данные, сохраненные в документе:

Изображение выглядит как текст

Автоматически созданное описание

1. В файл сохраняется вся информация из консоли (кроме строк ввода), а также информация о пяти оценках за практические работы студента
2. При отсутствии пяти оценок за практические работы, программа при сохранении в файл дополняет недостающее количество оценок «-»