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-420

А.В.Хардова

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(И.О. Фамилия)*

Проверили:

Г.И. Киреева

*(И.О. Фамилия)*

И.В.Сибирев

*(И.О. Фамилия)*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

**2022**

**Цель:** Ознакомиться с C# и ассемблерной вставкой.

Билет номер 2.

Содержание:

1. [Код программы(Листинг программы с комментариями кода):](#_Код_программы(Листинг_программы)
2. [Консоль программы (Результаты выполнения программы):](#_Консоль_программы_(Результаты)
3. [Текстовый файл программы (Содержимое):](#_Текстовый_файл_программы)

# Код программы(Листинг программы с комментариями кода):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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;

using System.Diagnostics;

namespace ConsoleApp

{

//////

///

public class Worker

{

string Surame = " ";

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

public Worker(string surname, string salary)

{

this.Surame = surname;

this.Salary = salary.Split(',').Select(x => int.Parse(x)).ToList(); //перевод строки в список

}

public List<int> AccessSalary()

{

return Salary; //возвращает

}

public string surname() //возвращает

{

return this.Surame;

}

public Worker WriteThis() //печатает инфу о работнике на экран и записывает определенные данные в файл

{

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

Console.Write("\tЗарплата: ");

foreach (int a in this.Salary)

{

Console.Write(a + ",");

};

// File.AppendAllText(path, "\n");

return this;

}

public int Summ() //Сумма зарплаты

{

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); //эта команда выделяет место в памяти для нашего ассемблер кода

\_IAllocatedMemory.Write(0, \_ByteS);

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

answer = \_delSumm(this.Salary.Count, this.Salary.ToArray()); //передаем управление ассемблерной функции с переданными аргументами

\_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:\exam.txt"; //статическое поле хранения нашего файла с инфой

[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: "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 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 double SalaryStCount(int \_A, int \_B) // средняя зарплата

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

public static void Main(string[] args)

{

File.WriteAllText(path, ""); //обнуляем наш файл при каждом новом запуске

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

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

{

string Grd = "";

Console.Write($"Введите фамилию работника {i}: ");

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

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

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

{

Random rnd = new Random(); //рандом

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

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

{

int grades = rnd.Next(40000, 60000);

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

}

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

Worker worker = new Worker(surname, Grd);

Group.Add(worker);

}

else

{

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

i--;

}

}

Console.WriteLine("\n\tДанные о работниках\n"); //здесь просто выводим информацию и записываем в файл

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

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

{

ArSalary.Add(Math.Round(SalaryStCount(Group[i].Summ(), Group[i].AccessSalary().Count()), 2));

Group[i].WriteThis();

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

Console.WriteLine("\tСредняя зарплата: " + ArSalary[i]);

}

for (int i = 0; i < 3; i++) //файл

{

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

int summ = 0;

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

File.AppendAllText(path, "Зарплата за 6 месяцев : ");

for (int y = 0; y < 6; y++)

{

try

{

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

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

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

}

catch

{

File.AppendAllText(path, " - ");

}

}

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

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

}

Console.ReadLine();

}

}

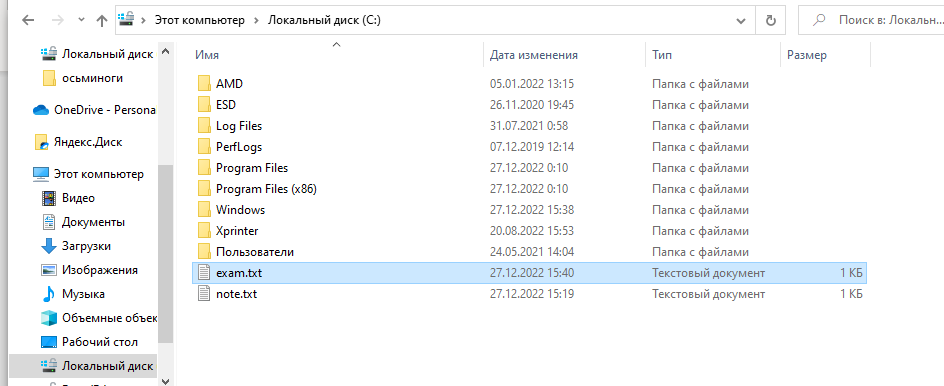
}

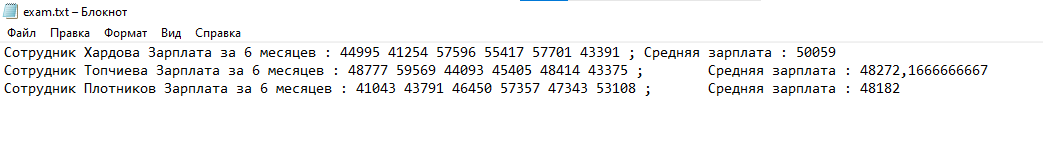
# Консоль программы (Результаты выполнения программы):

Изображение выглядит как текст

Автоматически созданное описание

# Текстовый файл программы (Содержимое):





Вывод **:** Я смогла ознакомиться с C# и ассемблерной вставкой.